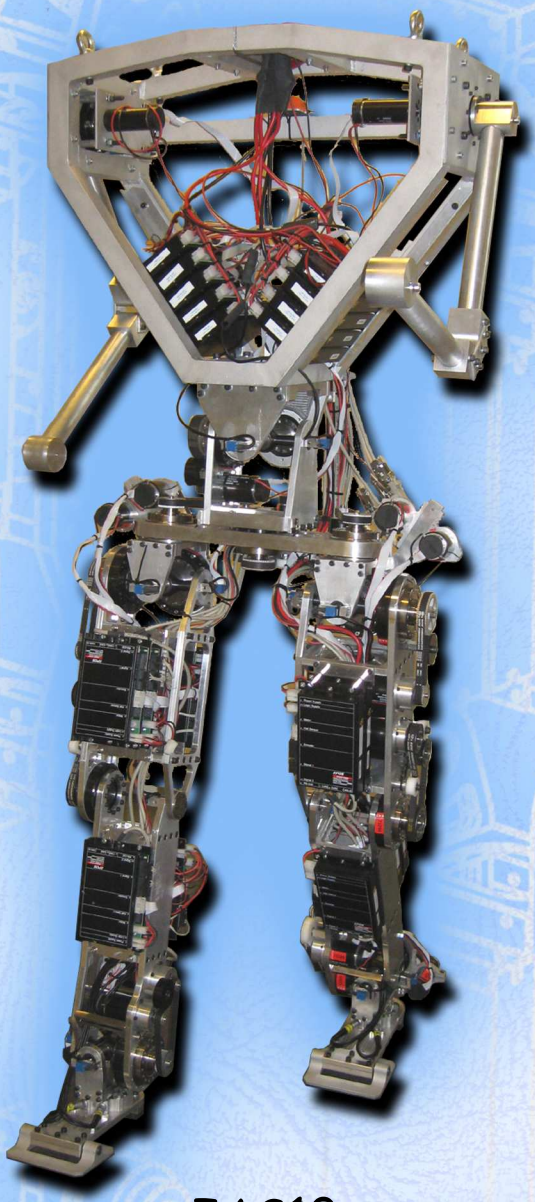
Modeling, Simulation, and Control of Biped Robot AAU-BOT1



IAS10
Fall 2008 - Spring 2009
09gr1031

AALBORG UNIVERSITY

AALBORG UNIVERSITY
Faculty of Engineering, Science and Medicine
Department of Electronic Systems
Section for Automation and Control
Fredrik Bajers Vej 7 C3
9220 Aalborg Ø
Denmark
http://www.control.aau.dk/

Title:
Modeling, Simulation, and Control of Biped Robot AAU-BOT1

Theme:
Master Thesis

Project period:
IAS9-IAS10
September 2nd 2008 - June 3rd 2009

Project group:
09gr1031

Group members:
Brian Thorarins Jensen
Michael Odgaard Kuch Niss

Supervisors:
Professor, Jakob Stoustrup
Associate Professor, Jan Helbo
Ph.d. Student, Mads Sølver Svendsen

Number printed:
6

Number of pages:
Report: 114
Total: 140

Abstract:
The field of robotics evolves rapidly, motivated by the thoughts of making robots perform tasks that humans do. This master's thesis concerns the further development of **AAU-BOT1**, which is a bipedal robot with human proportions, designed and built at Aalborg University. The robot is designed to combine the fields of robotics and health research in the study of gait patterns including dysfunctional limbs.
To provide a platform for this project and further development on **AAU-BOT1**, a complete hardware and software platform is set up to allow for easy implementation of new control systems.
A complete model of the robot is set up in order to simulate the robot realistically and to be used in the control system design.
To make the robot walk static balanced gait a trajectory is generated which meets the physical limitations of **AAU-BOT1**. To follow the trajectory a control system is created based on an unscented Kalman estimator, a posture controller, and a balance controller. The estimator is designed to obtain an estimate of unmeasured system states needed in the posture controller, which consists of feedback linearization and decoupling based on the computed-torque control method, combined with linear controllers.
The control system is tested with changes in the parameters of the system model to reflect model uncertainties compared to the real robot.
The final result is a complete development platform for **AAU-BOT1** and a control system which can make **AAU-BOT1** perform static balanced gait in simulations.

The contents of this report is freely available, but can only be published (with statement of source) as per agreement of the authors.

Preface

This report documents the results of the work of group 09gr1031 on 9^{th} and 10^{th} semester at Section of Automation and Control, Department of Electronic Systems, Aalborg University.

References in the Report

- Literature references of source material are referred to like [Siciliano and Khatib, 2008], where the name of the author and year of publication are in square brackets.
- Figures and tables are referred to by the number of the object. In a reference like Figure 1.1 the first number refers to the chapter and the second number refers to the consecutive figure number of the chapter.
- References to other parts of the report are done by referring to the number of the chapter or section where the content is located. These references are supplemented by page references.
- Equations are referred to much like figures and tables, however, with the number enclosed in brackets; i.e. Eq. (2.1).
- The attached DVD-ROM contains `MATLAB` scripts and `MATLAB Simulink` models developed for the **AAU-BOT1**. Furthermore, additional material concerning the project is also located on the DVD. References to the DVD are made in the following way, [`System_Test/`] where the text inside the brackets is the path on the DVD where the information is located.

The nomenclature, which contains the symbols used throughout the report, can be found on page viii.

A DVD is attached on page 140 containing additional documentation, and is referred to throughout the report.

Brian Thorarins Jensen

Michael Odgaard Kuch Niss

Contents

Nomenclature

Modeling

${}^{i-1}\vec{a}_i$	The link vector from joint $i-1$ to i	[m]
${}^{i-1}\vec{b}_i$	The CoM vector for link i seen in frame $i-1$	[m]
E_{kin}	The kinetic energy of the system	[J]
E_{pot}	The potential energy of the system	[J]
E_{grf}	is the potential energy contribution from the ground reaction forces	[J]
$\mathcal{F}$	The external forces for **AAU-BOT1**	
F_{g}	The load force acting vertical downwards	[N]
$\vec{F}_{\text{N},i}$	The normal force acting on the ith link	[N]
F_{x}, F_{y}, F_{z}	The force in the x-, y-, and z-axis of the FTS	[N]
g	The gravitational acceleration constant	$\left[\text{m/s}^2\right]$
$G_{\min}$ and $G_{\max}$	The minimum and maximum gear ratio respectively, in any joint	$[\cdot]$
G_{belt}	The gear ratio of the belt gears	$[\cdot]$
$\vec{G}(\vec{\theta})$	The vector containing all gravity terms	
$\boldsymbol{i}_{\text{m},i}$	The motor current of joint i	[A]
$\boldsymbol{I}_i$	The inertia tensor of link i around its CoM	$\left[\text{kgm}^2\right]$
I_{belt}	The moment of inertia of the belt gears	$\left[\text{kgm}^2\right]$
I_{CPU}	The moment of inertia of the Harmonic Drive gears	$\left[\text{kgm}^2\right]$
I_{m}	The moment of inertia of the high-speed motor shaft	$\left[\text{kgm}^2\right]$
I_{motor}	The moment of inertia of the motor	$\left[\text{kgm}^2\right]$
$\boldsymbol{J}(\vec{\theta})$	The Jacobian of **AAU-BOT1**	
K_{g}	The stiffness/torsion coefficient of the gear	[Nm/rad]
$\mathcal{L}$	The Lagrangian of **AAU-BOT1**	
m_i	The mass of the ith part of the robot	[kg]
$\vec{M}$	The sum of all moments affecting the robot	[Nm]
M_{x}, M_{y}, and M_{y}	The torques around the x-, y-, and z-axis, respectively	[Nm]
$\boldsymbol{M}(\vec{\theta})$	The mass matrix of the robot	
$\boldsymbol{M}_{\text{sim}}(\vec{\theta})$	The mass matrix utilized in the dynamic simulation model	
$\boldsymbol{M}_{\text{r}}(\vec{\theta})$ and $\boldsymbol{M}_{\text{l}}(\vec{\theta})$	The mass matrices calculated from the right and left toe, respectively	
$\vec{O}_{\text{ref}}$	The orientations used in the inverse kinematic	[rad]
$\vec{p}_{\text{CoM}}$	The position of the Center of Mass (CoM) of the robot	[m]
$\vec{p}_{\text{CoP}}$	The position of the Center of Pressure (CoP)	[m]
$\vec{p}_{\text{GCoM}}$	The position of the ground projection of the CoM	[m]
$\vec{p}_i$	The position of CoM of link i	[m]
$\vec{p}_{\text{o}}$	The point of origin	[m]
$\vec{p}_{\text{g}}$	The points with ground reaction forces	[m]
${}^0\vec{p}_i$	The CoM point for link i seen in the body frame	[m]
${}^0\vec{p}_{\text{j},i\text{-}1}$	The origin of the frame in joint $i-1$ seen in the body frame	[m]
$\vec{p}_{\text{ref}}$	The positions used in the inverse kinematic	[m]
p_{ZMP}	The point of the ZMP	[m]
$\vec{q}$	The generalized coordinate vector	

${}^{i-1}_{i}\boldsymbol{R}$	The rotation matrix from frame i to frame $i-1$	
${}^{i-1}_{i}\boldsymbol{T}$	The transformation matrix from frame i to $i-1$	
$\vec{V}(\vec{\theta}, \dot{\vec{\theta}})$	The velocity vector which contains centrifugal and Coriolis terms	
w_{sim}	The weight factor between the two feet	$[\cdot]$
$\vec{x}$, $\vec{y}$, and $\vec{z}$	The global coordinates for each link CoM	[m]
x_{ZMP} and y_{ZMP}	The x- and y-coordinate of the ZMP, respectively	[m]
x_{CoP} and y_{CoP}	The x- and y-coordinate of the CoP, respectively	[m]
z_i	The z-coordinate of the CoM of link i	[m]
$z_{\text{g},j}$	The z-coordinate of the jth spring attachment point	[m]
$\vec{\omega}_i$	The angular velocity vector of link i	[rad]
$\vec{\tau}$	The sum of torques acting on the joints	[Nm]
τ_{c}	The Coulomb friction in the joints	[Nm]
$\vec{\tau}_{\text{f}}$	The frictions torques in the joints	[Nm]
$\vec{\tau}_{\text{l}}$	The load torques in the joints	[Nm]
$\vec{\tau}_{\text{m}}$	The torques excerted by the DC motors	[Nm]
τ_{s}	The stiction in the joints	[Nm]
$\vec{\tau}_{\text{j}}$	The torques acting in the joints	[Nm]
$\vec{\tau}_{\text{gh}}$	The torque acting on the high-speed shaft of the gear	[Nm]
$\vec{\tau}_{\text{gl}}$	The torque acting on the low-speed shaft of the gear	[Nm]
$\vec{\theta}$	The angle vector containing the joint angles	[rad]
$\vec{\theta}_{\text{m}}$	The angle vector containing the motor angles	[rad]
$\theta_{\text{abs},i}$	The measured absolute joint angle of the ith joint	[rad]
$\vec{\theta}_{\text{ref}}$	The joint reference angle	[rad]
$\theta_{\text{rel},i}$	The measured relative joint angle of the ith joint	[rad]
θ_{Δ}	The torsion angle of the gear	[rad]
$\vec{\zeta}_i$	The vector that denotes the rotational axis of $\dot{\theta}_i$	$[\cdot]$
μ	The viscous friction in the joints	[Nms]
μ_{m}	The viscous friction of the high-speed motor shaft	[Nms]
μ_{g}	The gear slack damp coefficient	[Nms]
η_{g}	The efficiency of the gear	$[\cdot]$

Control

d_{sm}	The stability margin	[m]
$f\left(\vec{x}(k), \vec{u}(k)\right)$	The discrete non-linear process model	
$\vec{F}(\vec{\theta}, \dot{\vec{\theta}})$	The friction forces, including viscous friction, Coulomb friction, and stiction	[Nm]
$F_{\text{z,r}}$ and $F_{\text{z,l}}$	The resultant vertical force on the right and left foot, respectively	[N]
$h\left(\vec{x}(k), \vec{u}(k)\right)$	The discrete non-linear observation model	
$\boldsymbol{K}(k)$	The Kalman gain at time k	
n	The number of states in the state vector	$[\cdot]$
$\boldsymbol{P}_{\text{xx}}(k-1\|k-1)$	The estimate of the state covariance matrix at time $k-1$	
$\boldsymbol{P}_{\text{xx}}(k\|k-1)$	The predicted state covariance matrix at time k	
$\boldsymbol{P}_{\text{xz}}(k\|k-1)$	The predicted cross correlation matrix between the states and the observations at time k	
$\boldsymbol{P}_{\text{zz}}(k\|k-1)$	The predicted innovation covariance matrix at time k	
$\boldsymbol{P}_{\text{xx}}(k\|k)$	The estimate of the state covariance matrix at time k	
$\boldsymbol{Q}$	The process noise covariance matrix	
Q	The phase of the robot	$[\cdot]$
$\boldsymbol{R}$	The observation noise covariance matrix	

$\vec{u}(k)$	The input vector at time k	
v	The control signal	$\left[\mathrm{rad/s}^2\right]$
$\hat{\vec{x}}(k-1\|k-1)$	The estimate of the state vector at time $k-1$	
$\vec{\mathcal{X}}_i(k-1\|k-1)$	The ith sigma point at time $k-1$	
$\vec{\mathcal{X}}_i(k\|k-1)$	The ith sigma point propagated to time k	
$\hat{\vec{x}}(k\|k-1)$	The predicted mean of the state vector at time k	
$\hat{\vec{x}}(k\|k)$	The estimate of the state vector at time k	
$w_{\mathrm{c},i}$	The weight of the ith sigma point when calculating the covariance of the Gaussian	$[\cdot]$
$w_{\mathrm{m},i}$	The weight of the ith sigma point when the mean is calculated	$[\cdot]$
w	The weight factor between the models used in the control	$[\cdot]$
$\vec{\mathcal{Z}}_i(k\|k-1)$	The predicted ith observation vector at time k corresponding to the ith sigma point	
$\hat{\vec{z}}(k\|k-1)$	The predicted mean of the observation vector	
$\vec{z}(k)$	The observation vector at time k	
α and κ	Tuning parameters	$[\cdot]$
β	A tuning parameter which determines how much extra weight is put on the 0th sigma point	$[\cdot]$
λ	Parameter that determines how far the sigma points are spread from the mean	$[\cdot]$
$\vec{\tau}_{\mathrm{m,r}}$ and $\vec{\tau}_{\mathrm{m,l}}$	The motor torque calculated from the right and left foot model, respectively	[Nm]
θ_{e}	The error between the reference angle the actual angle	[rad]

I

Preliminaries

1 Introduction

Contents

The concept of having a biped robot walking like a human is interesting in many contexts. Optimally a biped robot equipped with limbs designed after human proportions should be capable of performing many of the same physical tasks as human do. This includes walking in rough terrain, up and down hills and stairs, bending down to pick up objects, and other similar tasks.

With regard to **AAU-BOT1** the overall purpose of the project is to end up with a human sized robot capable of walking dynamically, following different human gait patterns dependent on the set-up, e.g. simulating dysfunctional limbs. When the final goal is fulfilled the robot will be able to close the gab between the fields of health technology and robotics, hence, increase the collaboration between the Department of Health Science and Technology and the Section of Automation and Control at Aalborg University.

AAU-BOT1 is designed as a human size biped and is approximately 180 cm high and has a weight of approximately 70 kg. A picture of the assembled robot can be seen in Figure 1.1; additional pictures of **AAU-BOT1** can be seen on the attached DVD in [`Pictures/`].

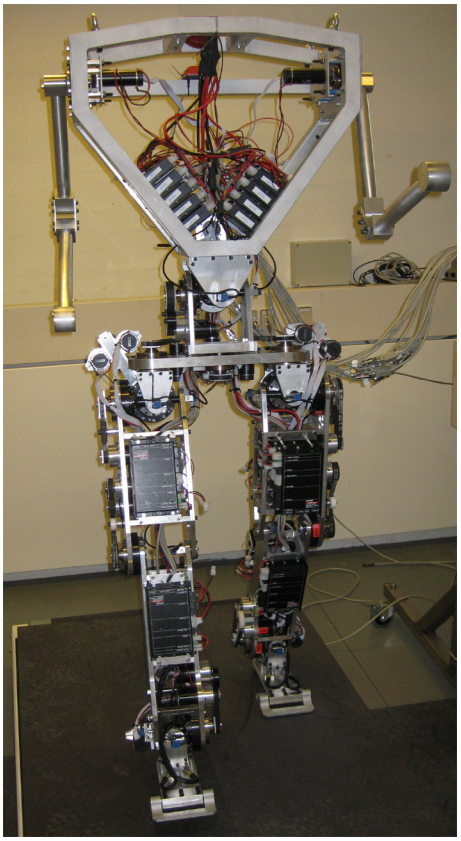
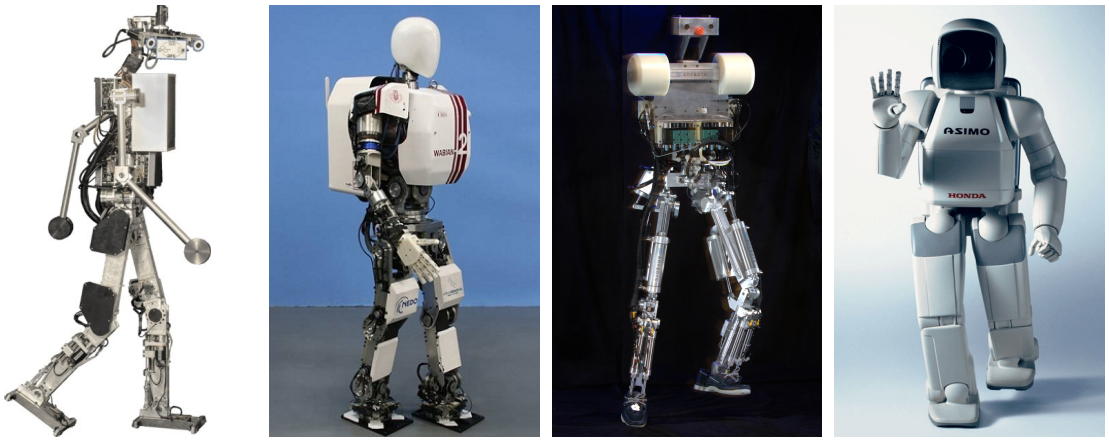


Figure 1.1: *A picture of* **AAU-BOT1** *fully assembled.*

1.1 Relation to Other Robot Projects

Both in the past and present other biped robots have been and are being developed throughout the world. However, in some ways **AAU-BOT1** stands out from the rest, first of all it is the first human size biped ever build in Denmark. **AAU-BOT1** also stands out from the bipeds in the rest of the world in the way that only very few of them have toe joints which should make it possible to achieve more human like dynamic walk.

To give an idea of other advanced biped robots, four of those currently being developed elsewhere in the world are shown in Figure 1.2. A short description of the four robots is given in this section, related to the **AAU-BOT1** project.



(a) Johnnie [Technische Universität München, 1998]

(b) WABIAN-2R [Takanishi Laboratory, Waseda University, 2006]

(c) Dexter [Anybots, 2007]

(d) ASIMO [HONDA, 2000]

Figure 1.2: *Four advanced biped robots with human proportions from different places in the world.*

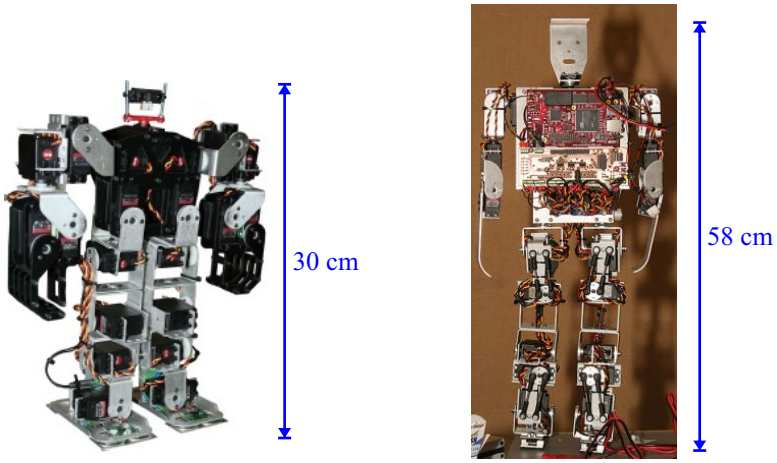
In the design phase of the **AAU-BOT1** project two of the other robots have been the main source of inspiration; these are Johnnie and WABIAN-2R, shown in Figure 1.2(a) and Figure 1.2(b), respectively. **AAU-BOT1** is in many ways similar to Johnnie which is developed at Technische Universität München where the project started in 1998. Johnnie has 17 degrees of freedom all located in the same fashion as the 17 actuated degrees of freedom which **AAU-BOT1** has. Another key feature that both Johnnie and WABIAN-2R have, and which has been adopted by **AAU-BOT1**, is the six axis force torque sensors located in both feet. These sensors are utilized to measure the forces and moments in the feet for use in the control of the robot.

The design of the toe joint in **AAU-BOT1** is inspired by the unactuated toe joints in WABIAN-2R. WABIAN-2R is developed at Waseda University and was first presented in 2006. The number of degrees of freedom in WABIAN-2R exceeds that of Johnnie as it has no less than 41 degrees of freedom. It should be noted though, that the high number also includes a much more sophisticated design of the arms and hands.

The robot ASIMO is a commercially build robot by HONDA, hence not much technical information is provided about the robot. However, the reason for including it here is that many people have either seen or heard of ASIMO. This means that even though ASIMO does not contribute much to the public knowledge about biped robots, it still has a great influence as it makes people around the world aware of the existence of biped robots.

Dexter is a robot completely different from the rest as it is driven by air cylinders instead of DC motors like the rest. The key difference in this is that the air cylinders are springy and flexible much like human muscles [Anybots, 2007], which also means that the robot has no stable posture that it can be put in without any active feedback. The interesting thing about Dexter is that the developers have actually succeeded in making it walk dynamically much like a human does it.

Also at Aalborg University other biped robot projects have been carried out prior to the actual **AAU-BOT1** project. Those projects were intended to clarify how to design a biped robot with human proportions. The first of these projects was the Sergeant which is a commercially build



(a) The Sergeant (b) Roberto

Figure 1.3: *The first two biped robots at Aalborg University.*

robot which was bought in 2005. A picture of the robot is shown in Figure 1.3(a). The Sergeant is a robot with a height of about 30 cm and with 20 degrees of freedom. The Sergeant was capable of walking static gait utilizing fuzzy logic, but it was discovered that the system was simply too slow to obtain human-like walk. The thesis written about the Sergeant is [Christensen et al., 2007b].

The second biped robot at Aalborg University was Roberto which was designed and constructed during a master's thesis from September 2006 to June 2007 [Christensen et al., 2007a]. The idea was to construct a biped robot with human proportions, just scaled down to a height of 58 cm. The robot is shown in Figure 1.3(b). Roberto is equipped with 21 actuated degrees of freedom. Due to limitations in the system, related to the interface to the servo-motors and a slow on-board computer, the project group was unfortunately unable to make the robot walk.

1.2 The History and Future of AAU-BOT1

The **AAU-BOT1** project at Aalborg University was initiated in September 2006 where a group at Department of Mechanical Engineering designed and built the mechanical platform of **AAU-BOT1** [Pedersen et al., 2007]. During the design some assistance was given from the Institute of Energy Technology in the choice of suitable motors and motor controllers for the robot.

In the spring of 2007 **AAU-BOT1** was handed over to Section of Automation and Control, under Department of Electronic Systems. In the period from September 2007 to June 2008 a group from Section of Automation and Control designed and implemented the sensor and actuator network for the robot. Additionally, some initial work was made in preparation for controlling **AAU-BOT1** [Jensen et al., 2008].

When the control of **AAU-BOT1** is finished and it is capable of walking, then the further development of the robot will be in collaboration between Section of Automation and Control and Department of Health Science and Technology. This collaboration will be necessary as one of the partners has the knowledge of how to make suitable controllers to perform certain tasks while the other has the knowledge of which walking patterns and situations that it will be beneficial to investigate. Optimally the robot will be equipped with an easily configurable controller by people at Section of Automation and Control so that it can be used easily by people at Department of Health Science and Technology. In this way projects concerning the robot can be conducted at both departments at the same time. The collaboration is a part of the overall purpose of **AAU-BOT1**, which is to simulate human like dysfunctional limbs and study the effect on the rest of the body. The intention is that these studies can be utilized to obtain new knowledge on how humans should be rehabilitated after injuries causing dysfunctional limbs.

1.3 Overview of AAU-BOT1

To give **AAU-BOT1** the capability to walk like a human, it has 19 movable joints where 17 of them are actuated; this is further described in Section 2.1 on page 10. The two toe joints are unactuated and are only affected by a passive spring that ensures that the heel and the toe become situated in the same plane when no external forces affect either of them. Since the overall purpose of **AAU-BOT1** is to simulate different gait patterns, most of the movable joints are located in the lower part of the body.

The following list provides a basic overview of the instrumentation of **AAU-BOT1**. Note that each of the items will be described in greater detail in Chapter 3 on page 21.

- **DC motors**
  The 17 actuated joints are actuated by DC motors, where 6 of the joints are actuated by two DC motors each due to heavy load. This results in a total of 26 DC motors on the robot.
- **Amplifiers**
  Each of the DC motors is powered by a separate digital amplifier located on **AAU-BOT1**.
- **Gears and belts**
  To transfer the power from the motors to the joints, they are connected via belts and gears.
- **Force Torque Sensor**
  On each foot a Force Torque Sensor (FTS), consisting of 6 strain gauge bridges, is mounted. The FTS is capable of measuring the forces and torques in three axes and thereby the Center of Pressure (CoP) can be determine.
- **Inertia Measurement Unit**
  The Inertia Measurement Unit (IMU) measures the orientation in all three axes of a standard right-hand Cartesian coordinate system. This unit provides useful measurement when having to make **AAU-BOT1** able to balance.
- **Network**
  The communication between the actuators, the sensors, and the on-board computer are done using several busses. The digital amplifiers for the motors are connected using CAN busses, the FTSs uses RS485 connections, and the IMU uses an RS232 connection.
- **On-board computer**
  The purpose of the on-board computer on **AAU-BOT1** is to control the movement based on the measurements. The mounting has not taken place yet, due to the fact that the mounting brackets have not been finished yet.

1.4 Project Objectives

The robot should, at the end, be able to imitate dysfunctional walking, e.g. limping. These imitations should be implemented through control of the robot, and not by any mechanical changes to the structure. However, before walking in a dysfunctional manner the robot must be able to walk as human like as possible. This walking pattern should be obtained by using the so called heel-strike and toe-off capabilities of the foot, which is further described in Section 2.3 on page 14.

At the beginning of the **AAU-BOT1** project some overall requirements were set up [Pedersen et al., 2007, p. 19], the ones that are most relevant to this part of the project are listed below:

- The robot must be able to perform dynamic walking. This includes starting, stopping, and standing still.
- The robot must be able to walk with a constant velocity of at least 1 m/s.

Due to the number of unfinished elements in the work done by previous project groups the goal for this part of the project is to make **AAU-BOT1** walk for the first time statically. This makes the thesis of this project:

Is it possible to make **AAU-BOT1** *perform static walk?*

To fulfill this goal the following must be satisfied:

- **A complete simulation model of the robot must be developed.** A simulation model capable of capturing all aspects important in relation to simulating walking correctly compared to the real world. This simulation model must include all gait phases of the robot.
- **All parameters needed to describe the physical robot must be determined.** The previous project groups have specified some of the physical parameters, but some are still missing and those specified earlier need to be checked against the physical robot.
- **The new computer must be set up to be able to perform the necessary tasks.** As a new computer has been bought for the robot, this must be set up in both hardware and software to be able to host the controller software.
- **An interface to the robot must be developed so that any controller can be implemented easily.** To make the controller development easier for both this and future groups, a suitable platform should be build without any bindings to a specific controller structure.
- **A trajectory suitable for static walk must be found.** A trajectory which it is possible to follow for the physical robot and makes it perform static walk must be determined. The chosen trajectory should have a high degree of stability to ensure that the robot does not tilt.
- **A controller capable of making the robot follow the determined trajectory satisfactory must be developed.** A suitable controller able to make the robot follow the trajectory in a sensible way must be designed. It should be investigated which controller structure that is necessary to fulfill the requirement of following the trajectory.

1.5 Thesis Outline

This thesis documents the work performed by group 09gr1031 in order to make **AAU-BOT1** walk statically. This work includes finalizing the hardware platform and developing a suitable software platform needed to implement controllers that can make the robot walk. Furthermore a complete model is set up to be able to simulate the robot and design controllers for it.

The thesis is divided into 5 parts with a number of chapters in each. A short description of all parts and chapters are given below:

- **Preliminaries**
  The purpose of this part is to introduce the reader to the **AAU-BOT1** project and the work described in this thesis. Furthermore, the fundamentals of bipedal walking are set up.
  - ***Introduction***
    This chapter provides some background knowledge about the **AAU-BOT1** project and the pre-studies for it. Additionally, the objectives of this project are presented.
  - ***Bipedal Walking***
    In order to provide the reader with a basis to understand the terminology used in the field of bipedal robots, this chapter defines the most commonly used expressions.
- **System Preparations and Modeling**
  This part describes the instrumentation of **AAU-BOT1** along with the developed software and the modeling of the robot.
  - ***Instrumentation***
    This chapter describes the instrumentation of the robot, both including the already implemented elements and those implemented during this project.
  - ***Software Structure***
    The software platform developed and set up to host the controllers for the robot is described in this chapter. The platform makes it possible to use a `MATLAB Simulink` implemented controller directly at the robot.
  - ***Modeling***
    The entire model of the robot is set up in this chapter. The models set up are a kinematic model, inverse kinematic model, dynamic models, and gear model. The dynamic models set up include both a complete simulation model utilized to simulate the robot and simplified models utilized in the controller design.
- **Control System Design**
  In this part the design of the actual control system for the robot is described. First of all a suitable trajectory for static gait is determined and afterwards a state estimator is designed, followed by the controller.
  - ***Trajectory Generation***
    To obtain proper gait a suitable trajectory is needed, hence the determination of such a trajectory is described in this chapter.
  - ***Estimator Design***
    As not all necessary system states are measured an estimator is designed to obtain proper estimates of the states. The design of the estimator is described in this chapter.
  - ***Controller Design***
    This chapter documents the development of the controllers which make the robot follow the trajectories, based on the estimated system states.
- **Control System Test and Conclusion**
  The purpose of this part is to finalize the thesis. This is obtained by a set of tests performed on the simulation model and finally a conclusion on the entire thesis.
  - ***System Test***
    The tests performed to validate the designed control system are described in this chapter.

  - ***Epilogue***
    In this chapter a summary and the conclusion on the entire thesis are given.

- **Appendices**
  This part contains all the appendices which further elaborate on some of the subject treated in the thesis.

  - ***Calibration of the Force Torque Sensors***
    To utilize the force torque sensors on the robot it is necessary to calibrate them properly. The calibration of the force torque sensors on the robot is described in this appendix.
  - ***Network Throughput Test***
    This appendix describes the tests performed in order to test the performance of the communication network of the robot. The tests are performed to ensure that the network can cope with the transmissions needed to control the robot.
  - ***Estimation of Friction Coefficients***
    As the friction in all joints of the robot is rather large it is important to obtain knowledge about it, both to include it in the simulation model and also to handle it in the control system. Therefore, the frictions are estimated and the procedure is described in this appendix.
  - ***Double Support Suggestions***
    During the project some different approaches to the implementation of the double support phase in the model have been tried out. Some of the proposals which did not end out in being utilized are described in this appendix.
  - ***Simple Control Design***
    This appendix describes a simple controller design which was tested and afterwards discarded as it could not cope with the task sufficiently well.

The project of **AAU-BOT1** has now been introduced, both concerning the overall purpose, which is to simulate dysfunctional limbs, and the goal in this master's thesis, which is to make **AAU-BOT1** perform static gait. In the following chapter some basic terms and definitions used in biped robotics are presented and described.

2 Bipedal Walking

In this chapter definitions concerning biped robots are introduced, along with some basic theory about biped locomotion containing anatomy, balance, and gait patterns.

Contents

2.1 Anatomy of the Body

In order to control **AAU-BOT1** it is necessary to know how it is pieced together; hence, this will be described in this section. Also, some basic definitions of terms used through the report will be set up.

In this report all coordinate systems used are right-hand Cartesian xyz-coordinate systems. The orientation of the global coordinate system in relation to the robot is shown in Figure 2.1. As it can be seen from the figure forward motion of the robot is defined as being in the x-direction, while the z-direction is in the upwards direction, and the y-axis is inserted in the left direction in order to obtain a right-hand xyz-coordinate system [Westervelt et al., 2007, p. 8].

For the description of rotation, the world of biped robots has adopted the terminology used to describe flight dynamics. This means that rotations are referred to as roll, pitch, and yaw, when it is around the x-, y-, and z-axis, respectively. All of these rotations are defined with the positive direction going counter-clockwise.

To ease the description of motion in a certain plane three names are set up to describe the xz-, yz-, and xy-planes, these names are sagittal plane, frontal plane, and transverse plane, respectively [Westervelt et al., 2007, p. 8].

AAU-BOT1 consists of several body parts connected in multiple joints. To reference these parts and joints the names shown in Figure 2.2 are used. It is also stated in the figure how many degrees of freedom (DoF) each of the joints have. All of the DoF are actuated except two, which are the toe joints in the feet. For both toe joints it applies that it is only influenced by a spring, keeping it leveled with the rest of the foot when it is not affected by external forces. In total the robot has 17 actuated DoF and 2 unactuated DoF.

To provide an understanding of each of the DoF, they are listed below together with a description of their use [Pedersen et al., 2007, p. 20].

- **Shoulder pitch axis**
  Improves the possibilities for balance maintenance and enables smooth walking.
- **Waist pitch axis**
  Improves the possibilities for balance maintenance.
- **Waist roll axis**
  Improves the possibilities for balance maintenance.

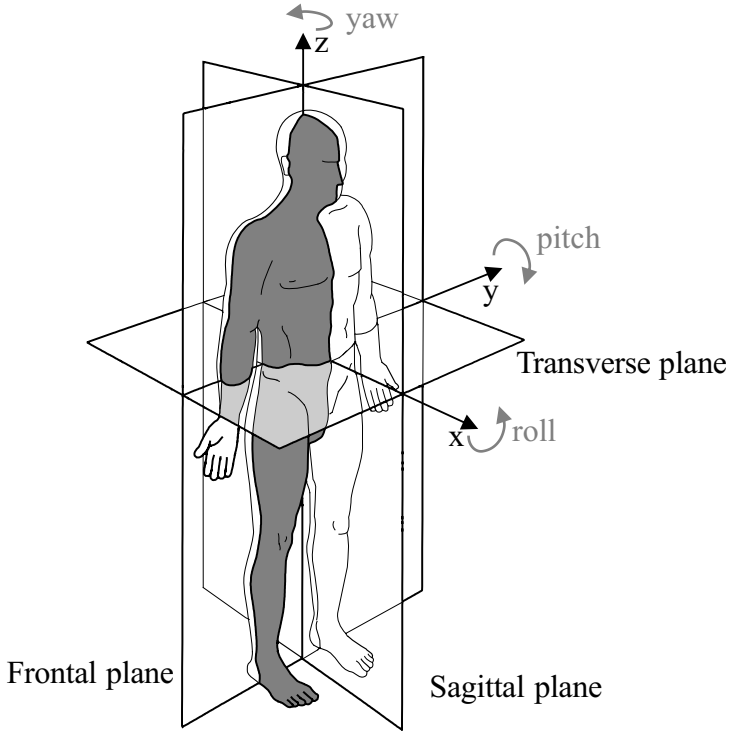


Figure 2.1: *Definition of the global axis, rotation names, and planes in relation to the robot. The figure is redrawn from [Westervelt et al., 2007, p. 8].*

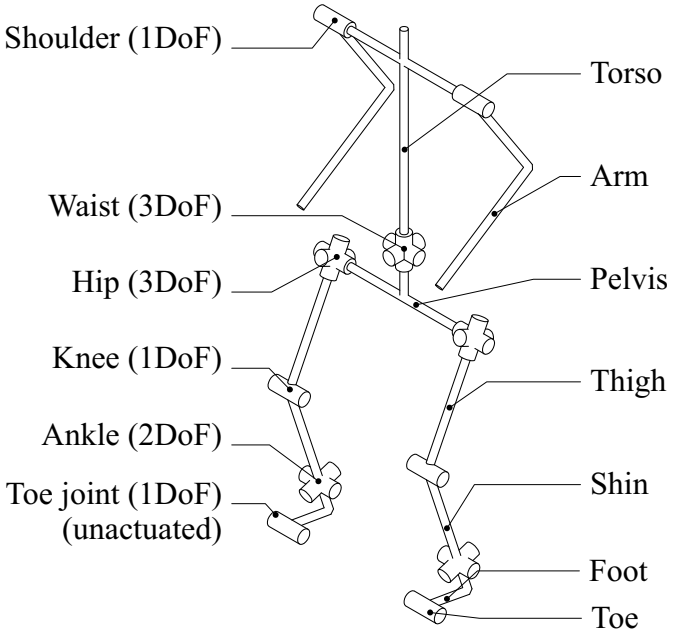


Figure 2.2: *Names of the different parts of the robot together with the joints between them. The figure is redrawn from [Pedersen et al., 2007, p. 20].*

- **Waist yaw axis**
  Used to keep the torso perpendicular to the walking direction.

- **Hip yaw axis**
  Enables pelvis rotation and enables the robot to make turns.

- **Hip pitch axis**
  Needed to swing the free leg forward.

- **Hip roll axis**
  Used together with ankle roll axis to shift the body weight over the standing leg, and used to enable pelvis list.

- **Knee pitch axis**
  Important to lift the leg during the swing phase and to absorb the shock on heel strike.

- **Ankle pitch axis**
  Necessary for performing toe-off during dynamic walking and for moving the body forward during the stance phase.

- **Ankle roll axis**
  Used together with hip roll axis to shift the body weight over the bearing leg, and used to enable pelvis list.

- **Toe pitch axis**
  Needed to perform toe-off during dynamic walking, which is described in Section 2.3.

2.2 Definitions

This section presents some important definitions that are generally used in the literature concerning bipedal robots.

2.2.1 Support Area (SA)

The support area is defined as being the convex hull of the ground support points [Wollherr, 2005, p. 8]. The support area of **AAU-BOT1** is shown in four different situations in Figure 2.3. In single support the support area only consists of the area spanned by the supporting foot, while in double support, the area between the feet is also included in the support area, as it can be seen from the figure.

2.2.2 Center of Mass (CoM)

A robot always experiences gravitational forces, which acts upon all parts of the robot. All of these forces can be summed up to a single virtual force acting upon the robots center of mass (CoM). The point of the center of mass is calculated as follows [Wollherr, 2005, p. 9]:

$$\vec{p}_{\text{CoM}} = \frac{\sum_i m_i \vec{p}_i}{\sum_i m_i} \tag{2.1}$$

where:

m_i is the mass of the ith part of the robot [kg]
$\vec{p}_{\text{CoM}}$ is the position of the CoM of the robot [m]
$\vec{p}_i$ is the position of the center of mass of part i of the robot [m]

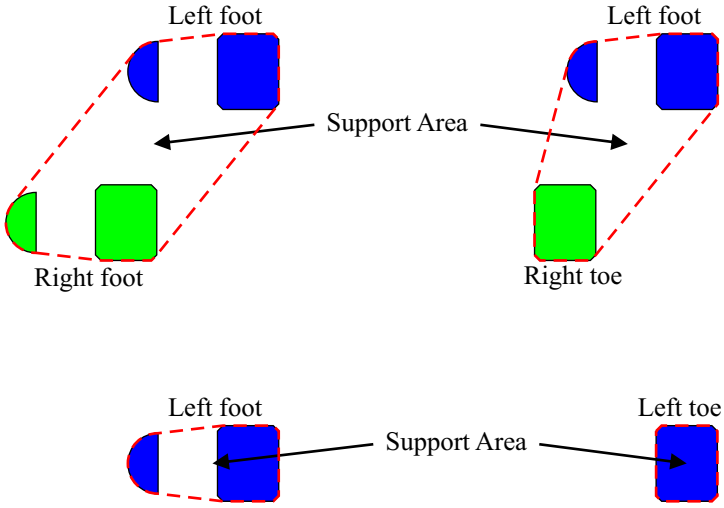


Figure 2.3: *The support area shown in four different situations as examples. The two top ones are in situations with double support and the two bottom ones are in single support.*

2.2.3 Ground Center of Mass (GCoM)

The ground center of mass (GCoM) is the projection of the center of mass point onto the ground plane [Wollherr, 2005, p. 9]. The point can mathematically be expressed as the point fulfilling the following equation.

$$\sum_i ((\vec{p}_{\text{GCoM}} - \vec{p}_i) \times m_i \vec{g}) = 0 \tag{2.2}$$

where:

$\vec{g}$ is the gravitational acceleration vector: $[0\ 0\ -9.82]^T$ [m/s^2]
$\vec{p}_{\text{GCoM}}$ is the position of the ground projection of the CoM [m]

2.2.4 Center of Pressure (CoP)

The center of pressure (CoP) is defined as the point on the ground plane where the resultant of the ground reaction forces acts [Goswami, 1999, p. 527]. It should be noted that there are two different interactions between the foot and the ground, namely the normal forces $\vec{F}_{\text{N},i}$ and the frictional tangential forces $\vec{F}_{\text{T},i}$. The CoP is the point given by [Wollherr, 2005, p. 11]:

$$\vec{p}_{\text{CoP}} = \frac{\sum_i \vec{p}_i |\vec{F}_{\text{N},i}|}{\sum_i |\vec{F}_{\text{N},i}|} \tag{2.3}$$

where:

$\vec{F}_{\text{N},i}$ is the normal force acting on the ith link [N]
$\vec{p}_{\text{CoP}}$ is the position of the CoP [m]
$\vec{p}_i$ is the position of the point where the ith force acts [m]

At the CoP point the resultant force $\vec{F}_{\text{N}} = \sum_i \vec{F}_{\text{N},i}$ acts. As a result of the definition the CoP always lies within the SA. In a situation were the robot is out of balance and starts falling down, the CoP lies on the edge of the SA.

During balanced walking the ZMP and the CoP are identical [Goswami, 1999]. This implies that the definitions for both points can be used together. This is a usable feature as the ZMP is easy to compute, hence well suited for trajectory generation, while the CoP is easy to measure, using force-torque sensors in the feet, hence more suited for control purposes.

2.2.5 Zero Moment Point (ZMP)

For the zero moment point (ZMP) there exist several definitions, which are all valid. Four of these are presented in [Goswami, 1999, p. 527] and one of them say that, the ZMP is the point on the ground where the total moment generated due to gravity and inertia equals zero as shown below.

$$\sum M_{\mathrm{x}} = \sum M_{\mathrm{y}} = 0 \tag{2.4}$$

where:

M_{x} and M_{y} are the torques around the x-axis and y-axis respectively [Nm]

From the definitions the ZMP must satisfy Eq. (2.5) and Eq. (2.6) [Bachar, 2004, p. 34].

$$\sum_i \left\{ m_i \left(\vec{p}_i - \vec{p}_{\mathrm{ZMP}} \right) \times \left(\ddot{\vec{p}}_i + \vec{\mathrm{g}} \right) + \boldsymbol{I}_i \dot{\vec{\omega}}_i + \vec{\omega}_i \times \boldsymbol{I}_i \vec{\omega}_i \right\} = \vec{M} \tag{2.5}$$

$$\vec{M} \times \vec{\mathrm{g}} = 0 \tag{2.6}$$

where:

$\boldsymbol{I}_i$	is the inertia tensor of link i around its CoM [kgm^2]
$\vec{M}$	is the sum of all moments affecting the robot [Nm]
$\vec{p}_{\mathrm{ZMP}}$	is the ZMP [m]
$\vec{\omega}_i$	is the angular velocity vector of link i [rad]

Eq. (2.5) and Eq. (2.6) can be combined into Eq. (2.7), where * denotes that the element is insignificant for the calculation, since it is only the x- and y-coordinate of the ZMP that is important.

$$\sum_i \left\{ \left(\vec{p}_i - \vec{p}_{\mathrm{ZMP}} \right) \times m_i \ddot{\vec{p}}_i + \boldsymbol{I}_i \dot{\vec{\omega}}_i + \vec{\omega}_i \times \boldsymbol{I}_i \vec{\omega}_i + \left(\vec{p}_i - \vec{p}_{\mathrm{ZMP}} \right) \times m_i \vec{\mathrm{g}} \right\} = \left[0 \; 0 \; * \right]^{\mathrm{T}} \tag{2.7}$$

By assuming that the ground is flat with no slope, the x- and y-coordinates of the ZMP can be solved from Eq. (2.7) and calculated as shown in Eq. (2.8) and Eq. (2.9) [Bachar, 2004, p. 35].

$$x_{\mathrm{ZMP}} = \frac{\sum_i \left\{ m_i \left(x_i \left(\ddot{z}_i + \mathrm{g} \right) - \ddot{x}_i z_i \right) - I_{i,\mathrm{y}} \omega_{i,\mathrm{y}} \right\}}{\sum_i \left\{ m_i \left(\ddot{z}_i + \mathrm{g} \right) \right\}} \tag{2.8}$$

$$y_{\mathrm{ZMP}} = \frac{\sum_i \left\{ m_i \left(y_i \left(\ddot{z}_i + \mathrm{g} \right) - \ddot{y}_i z_i \right) - I_{i,\mathrm{x}} \omega_{i,\mathrm{x}} \right\}}{\sum_i m_i \left(\ddot{z}_{\mathrm{i}} + \mathrm{g} \right)} \tag{2.9}$$

where:

x_{ZMP} and y_{ZMP} are the x- and y-coordinates of the ZMP respectively [m]

2.3 Gait Patterns

In this section the locomotion of the human body during walk will be described. For controlling a biped robot to walk like a human, the robot should have the same characteristics in the walking pattern as a human. Therefore the human walking pattern will be described concerning notions, gait cycle, and the body movement during walk.

In the following list some basic definitions, which are used in biped locomotion, are set up. The definitions concerns walk, gait, and step, and are based on [Vukobratović et al., 2007, p. 2].

- **Walk** is defined as the movement of the body by putting each foot in front in turns, while both feet never leave the ground at the same time.
- **Gait** is known as the manner of walking or running. Since the walk is specific for each individual, and is different in different situations, each of these walking patterns is said to represent a particular gait.

- **Step** is the basic cycle of gait that is repeated. Even though each walking pattern is specific there exists some general repeatability. This repeatability consists of the locomotion of the legs, i.e. the rear leg leaves the ground to become the front leg, while the front leg becomes the rear leg, and so forth.

To prevent the robot from falling down it has to achieve what is called balanced gait. Basically there are two types of balanced gait, which are static gait and dynamic gait [Vukobratović et al., 2007, p. 97].

- **Static balanced gait** occurs when the ground center of mass (GCoM) and the zero moment point (ZMP) are coincident and are located inside the support area (SA). The static balanced gait is characterized by slow movements to be able to neglect the effect of the dynamics of the robot on the movements. This also means that during static balanced gait, a robot can be interrupted at any time and stay in its current posture without falling down.
- **Dynamic balanced gait** occurs when the ZMP and GCoM are not coincident at all times. The ZMP must be located inside the SA and the GCoM is allowed to be both inside and outside the SA. In principle the body is falling forward during dynamic gait when the GCoM is located in front of the support area; this enables a faster gait with higher accelerations of the legs.

2.3.1 Gait Cycle

The human gait consists of a repeatable cycle which is described by two phases and eight events [Vaughan et al., 1992, p. 9]. The cycle is obtained by looking at one foot starting as the front foot by touching the ground with the heel. During the cycle the foot becomes the rearmost foot, leaves the ground and becomes the front foot again. While the foot is in contact with the ground the cycle is in the stance phase, while during the rest of the cycle the foot is not in contact with the ground, which is called the swing phase. The eight events in the cycle are shown in Figure 2.4 and described in the following for the right foot.

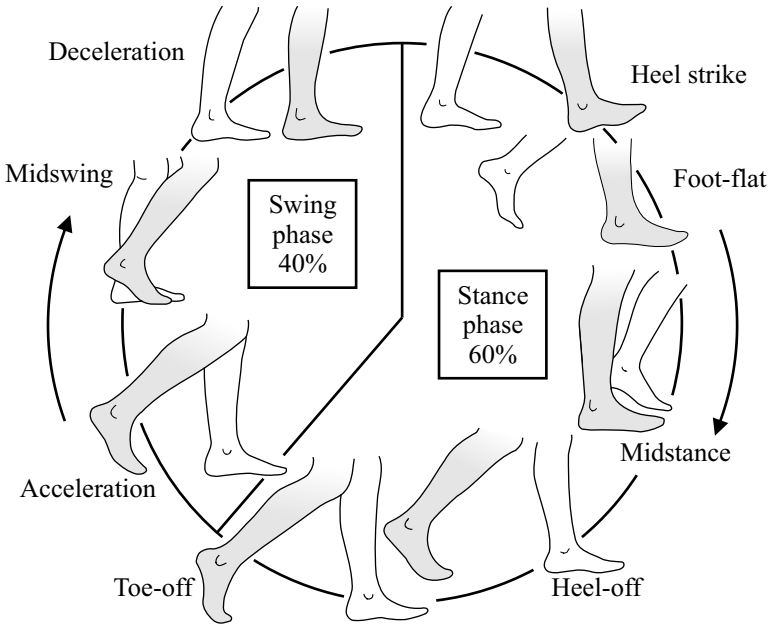


Figure 2.4: *The human gait cycle divided into a stance phase with five events and a swing phase with three events [Vaughan et al., 1992, p. 11].*

1. **Heel strike** initiates the gait cycle by the heel touching the ground. During this event the center of mass of the body has its lowest position.
2. **Foot-flat** is the situation where the plantar surface of the foot touches the ground.

3. **Midstance** is when the swinging foot passes the standing foot. The center of mass of the body is at this time at its highest position.
4. **Heel-off** is the situation where the back heel loses contact with the ground and a push is initiated via the triceps surae muscles, which plantar flex the ankle.
5. **Toe-off** is the last event in the stance phase as the foot leaves the ground.
6. **Acceleration** is the first event in the swing phase and occurs as soon as the foot leaves the ground, and the hip flexor muscles are activated to accelerate the leg forward.
7. **Midswing** occurs when the right foot passes directly beneath the body, corresponding to the midstance of the opposite foot.
8. **Deceleration** is when the muscles slow the leg down and stabilize the foot in preparation for the next heel strike to initiate the next step.

The support area during the gait cycle is illustrated in Figure 2.5. From the figure it is clear that the events heel strike and toe-off are double support areas and the events flat-foot, midstance, heel-off, acceleration, midswing, and deceleration are single support situations.

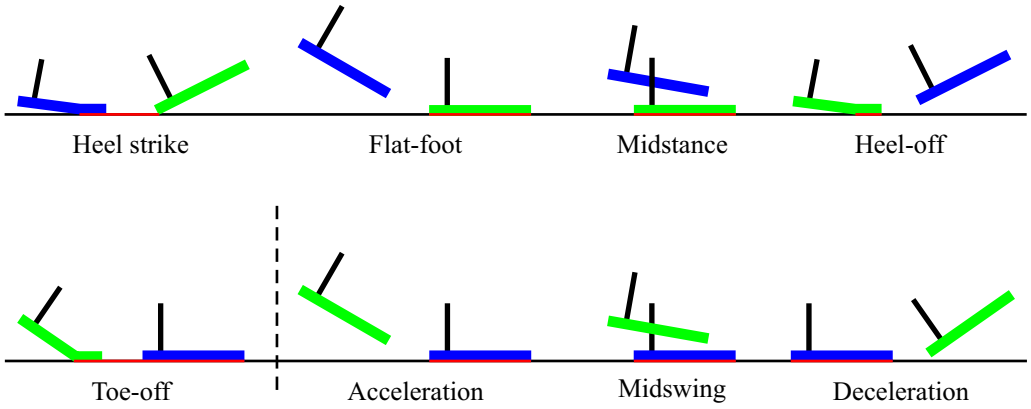


Figure 2.5: *The support area illustrated by a red line for each gait phase. The vertical dotted line illustrates the transition from stance phase to swing phase. The right foot is marked with green.*

2.3.2 Movements of the Body

Each body part moves in a very certain way during the gait cycle to achieve the most optimal and efficient gait. In the following the movements are described based on [Inman et al., 1981, p. 2-21].

- **Pelvis rotation**
  During gait the pelvis will rotate around the yaw axis for extending the step length. The pelvis rotates forward during the swing phase and backward during the stance phase.
- **Pelvic tilt**
  In the single support phases the pelvis will tilt downward in the non-weight bearing side, resulting in a rotation in the frontal plane around the roll axis. The rotation occurs in the hip joints and results in the non-supporting leg having to flex the knee in order to stay clear of the ground during the swing phase.
- **Knee flexion in stance phase**
  At heel strike the knee is almost straight, but starts to flex until the foot is flat on the ground. The flexion absorbs some of the impact of the heel-strike and prevents an excessive rise of the CoM.
- **Sideway displacement of the body**
  During the stance phase the entire body shifts to the side of the weight bearing leg, by rotating around the roll axis.

- **Body rotation**
  Besides rotation of the pelvis, the body also rotates around the yaw axis in the transverse plane in the following movements.
  - **Torso rotation**
    During walk the torso rotates in opposite direction of the pelvis, causing the arms to swing. When a leg swings forward the opposite arm simultaneously swings forward.
  - **Thigh and shin rotation**
    The lower part of the leg rotates in phase with the pelvis, and the rotation increases from the pelvis to the shin. From the midswing phase the leg starts to rotate inwards, until midstance where the outward rotation begins.
  - **Ankle and foot rotation**
    Mechanisms in the ankle joint and the foot absorb rotations from the leg while the foot is in contact with the ground in the stance phase.

Definitions and terms used in biped robotics have been introduced and described, concerning the anatomy, balance, and gait patterns. In the following part some necessary preparations for **AAU-BOT1** are presented, starting with the instrumentation of **AAU-BOT1**.

II

System Preparations and Modeling

3 Instrumentation

In this chapter the instrumentation of **AAU-BOT1** will be described. First, all the sensors and actuators available on **AAU-BOT1** will be described in detail. Finally, the network structure between the sensors, actuators, and the on-board computer is discussed, together with the set-up of the on-board computer.

Contents

3.1 Actuators

The actuators available on **AAU-BOT1** are 23 DC motors distributed on the 17 actuated joints, as 6 of the joints are actuated by two DC motors each due to heavy load. Each motor is controlled by an amplifier that is connected to the on-board computer, as described in Section 3.3 on page 26.

3.1.1 DC Motors

The DC motors on the robot are from the Maxon RE series which were found suitable for **AAU-BOT1** by a previous group [Pedersen et al., 2007, p. 51]. The 23 motors have different power ratings, depending on the load in the joints; the used models are 60 W, 90 W, and 150 W. In Table 3.1 it is shown which type of DC motor that is mounted in each joint. As 6 of the joints are actuated by two motors each, 12 of the motors are controlled synchronously in pairs of 2. The implementation of the dual actuated joints is further discussed in Section 4.2 on page 29.

Each motor is mounted with an encoder that is used to determine the relative position of the motor shaft, which is further described in Subsection 3.2.1. The motors are controlled through the amplifiers. The total network set-up is described in Section 3.3 on page 26.

3.1.2 Amplifiers for DC Motors

To power and control the DC motors each motor is connected to an amplifier. The amplifiers chosen by the first group, [Pedersen et al., 2007], were found unsuitable based on the network analysis made by [Jensen et al., 2008]. Therefore, the amplifiers used on **AAU-BOT1** were changed to the ones currently in use, namely the Maxon EPOS 70/10 amplifiers, which have a built in PID-controller, A/D converter, digital I/O, and CAN-bus. The build in PID-controller enables speed and position control, by use of the encoders mounted on the motors. Additionally, the EPOS amplifier also enables current control, i.e. torque control of the motor. The interface to the EPOS amplifier is the CANopen protocol which ensures that the measurements can be fetched and the control of the motors can be achieved digitally via the CAN bus, i.e. the entire motor set-up is very noise immune.

Joint	Motor	Gear	Belt Gear Ratio	Total Gear Ratio
Shoulder pitch	60 W	CPU-S 14-100	20/16	125
Waist roll	90 W	CPU-S 14-100	57/19	300
Waist pitch	150 W	CPU-S 17-120	57/19	360
Waist yaw	60 W	CPU-S 17-100	46/16	288
Hip roll	2x150 W	CPU-S 20-120	69/30	276
Hip pitch	150 W	CPU-S 20-120	57/28	244
Hip yaw	60 W	CPU-S 17-100	40/19	211
Knee pitch	2x150 W	CPU-S 17-100	52/39	133
Ankle roll	90 W	CPU-S 14-100	44/22	200
Ankle pitch	2x150 W	CPU-S 20-160	35/28	311

Table 3.1: *The location and size of motors and gears on* **AAU-BOT1**. *Note that for the gears the first number refers to the physical gear size and the last number refers to the gear ratio. The belt gear ratio is the gear ratio of the sprockets and belts. The total gear ratio is the gear ratio from the joint to the motor composed by the gear and the different sprockets determined by [Pedersen et al., 2007, p. 101]. Please read the note in* Subsection 3.1.3 *concerning the gear ratios.*

3.1.3 Gears and Belts

The motors are connected to the joints via belts and gears. The gears are a combination of sprockets, belts, and Harmonic Drive gears and as for the motors different sizes of gears are used depended on the load in the joint. The Harmonic Drive gear is characterized by having high gear ratio and insignificant backlash. The different gears are listed in Table 3.1.

Note that the gear ratios listed in the Table 3.1 are different from the ones stated in both of the previous project reports about **AAU-BOT1**. This is deliberate as it was discovered that the numbers in none of the reports corresponds to the physical robot. The gear ratios of the Harmonic Drive gears were correct in [Pedersen et al., 2007] but not in [Jensen et al., 2008], and some of the belt gear ratios were wrong in both reports. The belt gear ratios stated in the table have been measured using a vernier caliper.

3.2 Sensors

To provide feedback for the controller a number of sensors are available on **AAU-BOT1**. The available sensors provide measurements of angles, forces and torques from each foot, and orientation.

3.2.1 Joint Angle Measurements

In order to determine the posture of **AAU-BOT1** it is necessary to measure the angle of each joint. For this purpose a number of sensors are mounted. To give an absolute measurement of the angle a potentiometer is mounted on the axle of each joint. As the measurement from a potentiometer is often rather noisy it does not provide a very accurate measurement. As a consequence the potentiometer is only used initially to get an absolute angle measurement, as an encoder is also available on each motor and provides a more accurate reading.

The potentiometer is connected to a 10 bit A/D converter in the EPOS, described in Subsection 3.1.2. According to [Pedersen et al., 2007, p. 49] the maximum required angle span is 94°, but all of the used potentiometers have a measurement range of 270°, which results in a resolution as shown in Eq. (3.1).

$$\theta_{\text{res,min}} = \frac{\theta_{\text{span}}}{n_{\text{A/D}}} = \frac{270^\circ}{2^{10}\text{ bit}} = 0.264^\circ/\text{bit} \tag{3.1}$$

where:

θ_{span} is the angle span of the potentiometers [°]
$\theta_{\text{res,min}}$ is the resolution of the angle measurement from the potentiometers [°/bit]
$n_{\text{A/D}}$ is the resolution of the A/D converters [bit]

During the project two problems with the potentiometers was discovered. The first problem is that the potentiometers are not precise, when the robot is put in the same position the measurement can easily vary several degrees. This makes it very difficult to utilize the measurement to position the robot in its global zero position. The second problem discovered is that the potentiometers cannot stand the movements of the robot. This means that the potentiometers get points where the voltage is measured as zero even though it should have been something else. This problem is very unfortunate as there is a safety circuit connected to the potentiometers to prevent damaging the robot by moving outside the possible angle range. This safety circuit stops the corresponding EPOS immediately if the potentiometer suddenly outputs zero voltage. As a consequence several of the potentiometers have been changed during the project, and some safety circuits have been disabled as the potentiometers continued to break.

For precise measurement of the angle during operation an encoder is mounted on each motor. These are of the type Maxon motor Encoder MR, Type L, 512CPT, 3 Channels, with Line Driver [maxon motor, 2008a], and provides a relative measurement of the angle with 512 pulses each turn. In relation to the data provided by Maxon an encoder with 512 pulses per turn results in a resolution of 2048 steps per turn. According to Table 3.1 the minimum gear ratio is 111 and the maximum gear ratio is 320. This results in a resolution, for all joints, in the interval given by the following two equations:

$$\theta_{\text{res,enc,min}} = \frac{360^\circ}{n_{\text{enc}} \cdot G_{\text{min}}} = \frac{360^\circ}{2048 \cdot 125} = 0.0014^\circ/\text{pulse} \tag{3.2}$$

$$\theta_{\text{res,enc,max}} = \frac{360^\circ}{n_{\text{enc}} \cdot G_{\text{max}}} = \frac{360^\circ}{2048 \cdot 360} = 0.0004^\circ/\text{pulse} \tag{3.3}$$

where:

$\theta_{\text{res,enc,max}}$ is the maximum resolution of the angle measurement from the encoders [°/bit]
$\theta_{\text{res,enc,min}}$ is the minimum resolution of the angle measurement from the encoders [°/bit]
G_{max} is the maximum gear ratio in any joint [°]
G_{min} is the minimum gear ratio in any joint [°]
n_{enc} is the number of pulses from the encoders for each turn [·]

3.2.2 Force Torque Sensors

In order to allow for calculation of the CoP of the robot, it is equipped with Force Torque Sensors (FTS) which measures the forces and torques acting upon them. A six axis FTS is mounted below each of the ankles of **AAU-BOT1**. These FTSs were developed at Aalborg University by an earlier project group; the design is described in [Pedersen et al., 2007, p. 155]. The usage of the FTSs is further described in [Jensen et al., 2008, p. 38]. From the measured forces and torques the x- and y-coordinates of the CoP can be calculated relative to the center of the FTS, as shown in Eq. (3.4) and Eq. (3.5) [Choi et al., 2004].

$$x_{\text{CoP}} = \frac{-M_{\text{y}}}{F_{\text{z}}} \tag{3.4}$$

$$y_{\text{CoP}} = \frac{M_{\text{x}}}{F_{\text{z}}} \tag{3.5}$$

where:

F_{z} is the force in the z-axis of the FTS [N]
M_{x} is the torque acting around the x-axis of the FTS [Nm]
M_{y} is the torque acting around the y-axis of the FTS [Nm]
x_{CoP} is the x-coordinate of the CoP [m]
y_{CoP} is the y-coordinate of the CoP [m]

During the testing and control of **AAU-BOT1** it is important not to overload either of the FTSs as that might damage them. To avoid overload it must be ensured throughout all tests that the maximum limits given in Table 3.2 are not violated.

Force/Torque	**Max value**
F_x	$\pm$1000 N
F_y	$\pm$1000 N
F_z	$\pm$2000 N
M_x (roll)	$\pm$200 Nm
M_y (pitch)	$\pm$230 Nm
M_z (yaw)	$\pm$30 Nm

Table 3.2: *Force/torque maximum loads [Pedersen et al., 2007, p. 155].*

One of the FTSs is illustrated in Figure 3.1. The left part shows the FTS mounted to the ankle and the foot, while the right part shows only the core of the FTS, which is the active part as the cylindrical part around it is merely supporting the core.

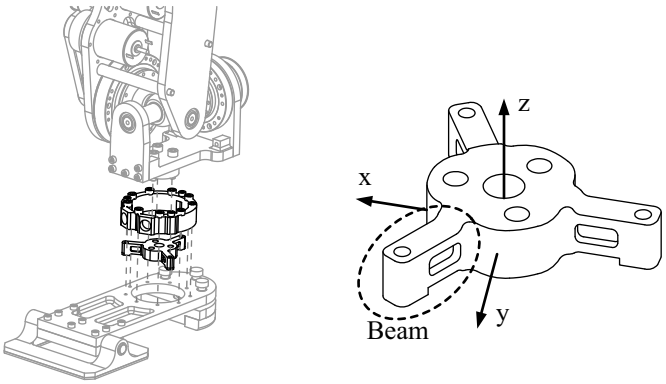


Figure 3.1: *Left: An exploded view of an FTS mounted in between the ankle and the foot of* **AAU-BOT1**. *Right: The core of the FTS with one of the beams marked. The figure is partly inspired by [Jensen et al., 2008, p. 39].*

Each of the three beams of the FTS has 8 strain gauges mounted on them, mounted in two full bridges. One of these bridges enables measurement of the shear forces and the other bridge allows for measurement of the bending moments. With two full strain gauge bridges per beam this gives a total of 6 sensor measurements per FTS. These 6 sensors are each connected to amplifiers from Lorenz Messtechnik GmbH of the type SI-RS485 [Lorenz Messtechnik GmbH, 2008]. The amplifiers each have 2 channels, capable of measuring with a 16 bit resolution and an update rate of 2,500 Hz. The output from the amplifiers are serial RS485 connections.

The displacement of the FTS core, which is measured as shear and bending, can be translated into forces and moments, in and around the three major axes of the coordinate system. The measurement of the bending related strain is mainly used to determine the in-plane traction forces F_x and F_y and the moment around the vertical axis M_z. These three displacements are illustrated in the top row of Figure 3.2. The measurement of the shear related strain is primarily used to calculate the out of plane reactions, namely the vertical force F_z and the two horizontal moments M_x and M_y, illustrated in the bottom row of Figure 3.2.

In order to use the FTSs they need to be calibrated. The calibration procedure for the sensors is described in Appendix A on page 117, and the results are presented in Eq. (A.5) on page 119 for the right foot and in Eq. (A.6) on page 120 for the left foot.

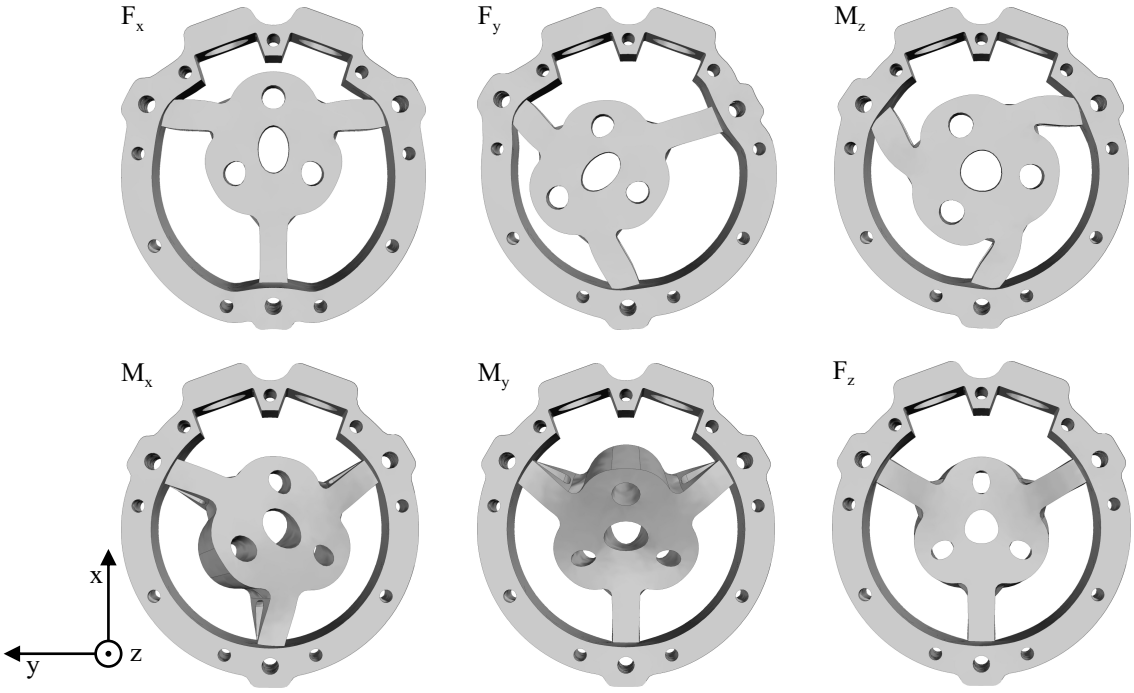


Figure 3.2: *One FTS affected by a single force or moment. The top row shows in-plane displacement, while the bottom row is out of plane displacement [Pedersen et al., 2007, p. 159]. Note that the forces and moments are highly exaggerated in order to make the effect clear in the illustration.*

3.2.3 Inertia Measurement Unit

When humans are walking they use their inner ear to sense motion and orientation, and when the orientation is felt it is possible to balance. Hence, for **AAU-BOT1** to be able to walk like a human it is desirable that it is capable of sensing the 3D orientation. For this reason **AAU-BOT1** is equipped with an Inertia Measurement Unit (IMU) which can measure the 3D orientation and accelerations. The IMU is an xsens MTi - miniature gyro-enhanced Attitude and Heading Reference Sensor.

3.3 Network

This section presents the connections between the on-board computer (OBC), the sensors, and the actuators. In between the motor controllers and the OBC, CAN-buses are utilized. The FTSs uses RS485 interfaces which are connected directly to the OBC. The IMU has an RS232 interface and is also connected directly to the OBC. All of the network set-up is shown in Figure 3.3.

Some parts of the network were ready to use at the start of this project, but the RS485 and RS232 connections needed to be made along with the interface to the CAN buses.

The drivers to handle the network buses are all described in Chapter 4.

3.4 On-board Computer

The purpose of this section is to explain the structure of the on-board computer (OBC) of **AAU-BOT1**. The computer hardware is a standard PC motherboard equipped with an Intel® Core™2 Duo 3.0 GHz CPU, 2 Gb memory, and a 32 Gb solid-state flash hard drive. Note that due to a mismatch between the CPU and the motherboard only 2.4 GHz of the CPU can be utilized.

To be able to interface the CAN buses and the serial connections needed in **AAU-BOT1**, the computer is extended with both a TPMC901 PCI extension card to provide 6 CAN interfaces and a TPMC465 PCI extension card to provide 8 serial interfaces. The serial interfaces can be individually set in software to be RS232, RS422, or RS485.

At project start the on-board computer had just been assembled, no operating system or other software was installed. Furthermore, the interfaces between the PCI extension cards and the buses on **AAU-BOT1** had to be made.

As the instrumentation of **AAU-BOT1** has now been described, the next chapter will elaborate on the software on the on-board computer.

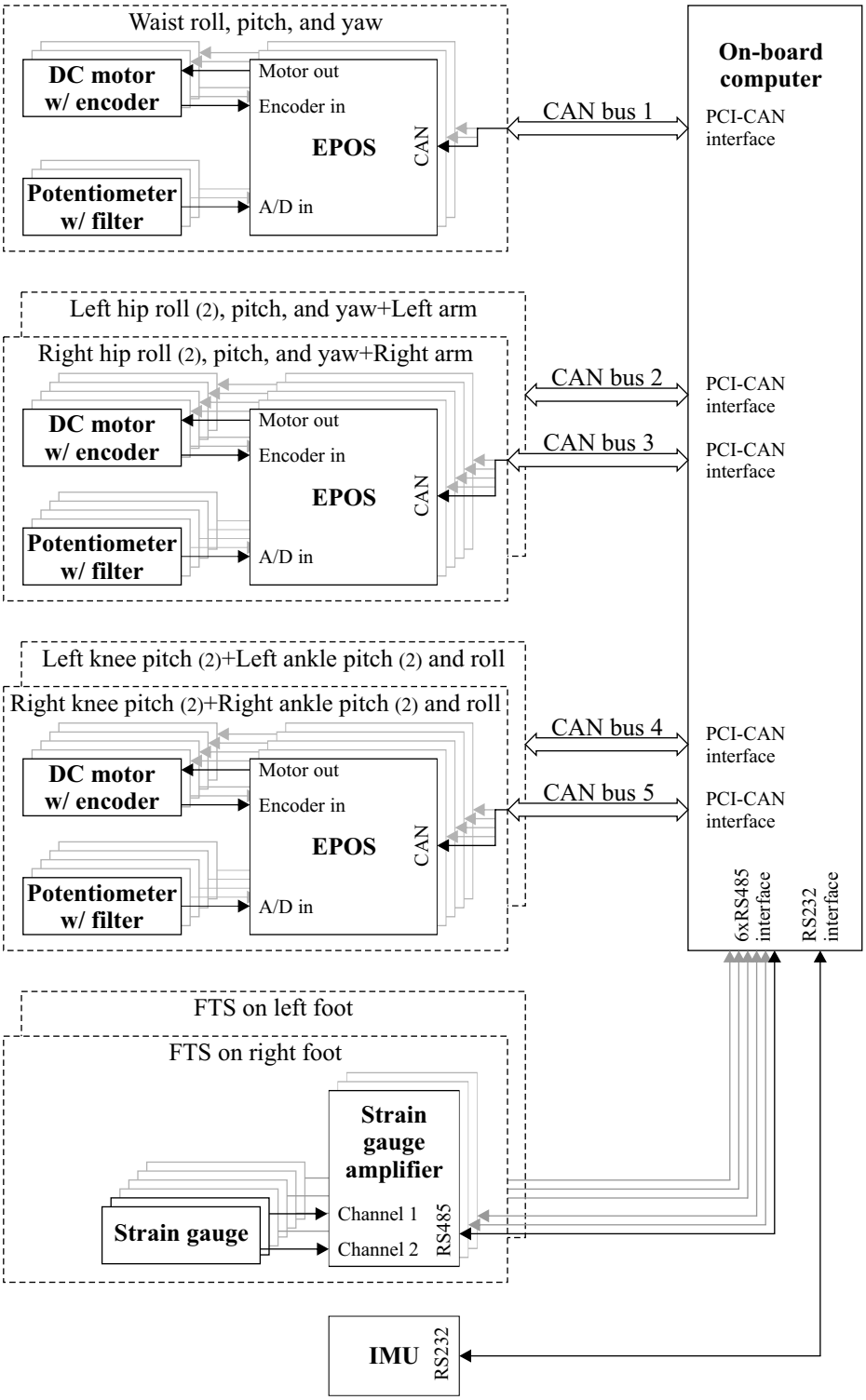


Figure 3.3: *The network in between the onboard computer, the actuators and the sensors. Note that the overlapping boxes illustrate several identical items. The joints marked with (2) are those with two sets of motors and amplifiers.*

4 Software Structure

This chapter will describe the software structure which forms the basis for the control system of the robot. The following section will describe the overall set-up of the designed software and the following sections will explain each separate part in greater detail. The developed source code for the **AAU-BOT1** programs is located on the DVD in [`Source_Code/`].

Contents

4.1 General Description

The on-board computer has been installed with Linux as operating system and modified by an RTAI installation, which provides hard real time capabilities, as it is desired for the control of **AAU-BOT1**.

The reason for choosing RTAI as the real time operating system is that the project group had some previous knowledge about it from an earlier project [Esbensen et al., 2007, pp. 25-26]. Additionally, RTAI is a natural choice as it is an open source project which is being further developed all the time by people around the world, hence it is a system that can be changed as needed and it will not be antiquated in the future when other project groups start working with **AAU-BOT1**. Another important aspect with RTAI is that it has an interface to the `MATLAB Simulink Real Time Workshop` which makes it possible to develop own blocks within a pre-defined framework.

Figure 4.1 provides an overview of the developed software on the OBC. As it can be seen from the figure the first layers above the PCI extension cards are the respective hardware drivers, separated into two parts for the TPMC465 extension card.

To be able to use the TPMC465 PCI-Serial extension cards it has been necessary to develop real time drivers for it, as only regular Linux drivers were available. To ease the development of the device driver the existing non-real time driver was used as a basis. The original driver is separated into two parts, a hardware application layer (HAL) which handles the low level communication to the hardware and an higher level driver which acts as the actual hardware driver.

Both hardware device drivers utilize functions in the RTAI module rtai_hal to communicate with the hardware and to handle interrupt controlling; hence, the RTAI module can be seen as an interface between the hardware and the device drivers.

To have a unified interface to all CAN and serial buses an I/O module is built on top of both device drivers. The I/O module is also build as a kernel module and handles the continuous reception of measurements and at all times provides the most recent measurement from all sensors to the layers above. Also, when control commands has to be sent to either the sensors or the actuators this is also handled through the I/O module.

RTAI provides the possibility to run programs in hard real time, even in user space. As it is more practical to run the controller in user space and the same real time possibilities are available there, it is decided to do so. To assist in the development of modules across the user/kernel space boundary, RTAI provides the module rtai_lxrt which provides an easy interface between kernel and user space.

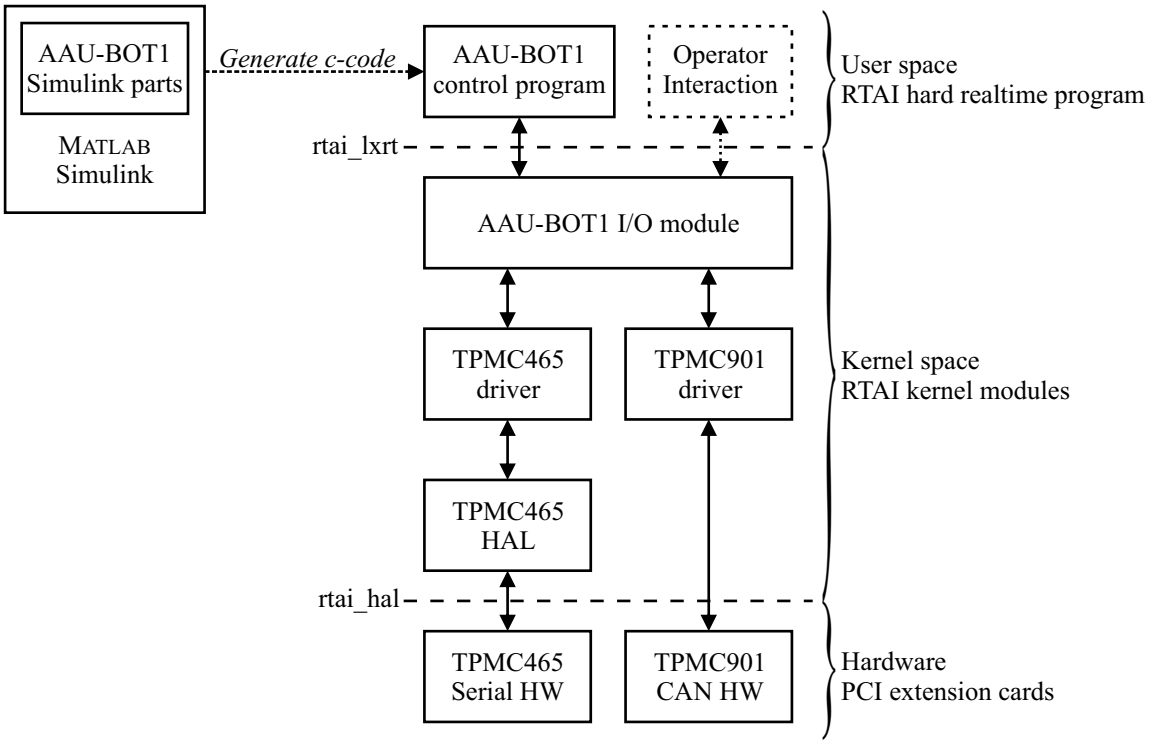


Figure 4.1: *The overall software structure on the OBC.*

When the controller is developed for user space the Real Time Workshop in `MATLAB Simulink` can be used to generate the C-code for the program. This necessitates the installation of an RTAI real time target in `MATLAB Simulink`, and the development of `MATLAB Simulink` blocks that are capable of interfacing the developed AAU-BOT1 I/O module. Furthermore, an operator interaction program is developed, which can interface the AAU-BOT1 I/O module directly, in order to have the control process under supervision at all times. Also, the operator interaction program can gather debug data in the development process.

The advantages of this set-up is that changes can easily be made to the controller in `MATLAB Simulink` and afterwards be compiled into a C-program which can be tested immediately on **AAU-BOT1**, without having to change anything in C-code or other code parts.

4.2 Simulink Interface

The `MATLAB Simulink` interface blocks are developed quite similar to those designed by [Jensen et al., 2008, p. 50]. The purpose of the blocks is to receive measurements from the sensors and to transmit commands to the actuators on **AAU-BOT1** by interfacing the AAU-BOT1 I/O module.

As the reference inputs to **AAU-BOT1** can be of three different types, namely current, velocity, or position, a block is developed, which has three different input ports and it can be selected in the Simulink GUI which of them is the active one. A general description of how a `MATLAB Simulink` S-function block is structured can be found in [Jensen et al., 2008, p.51].

In this project the double actuated joints have been implemented, which was not achieved by the previous project groups. When the joints are controlled in current mode, the current reference is divided by two as the same reference is sent to both motors. This ensures that the EPOSs makes the motors deliver half the torque each. Also, when the motors are controlled in either position or velocity mode only one of the motors are utilized while the other one is automatically put into current mode with a zero reference to prevent it from getting damaged.

The sensor input block is divided into three different parts as this is found to be more practical than the single block used by [Jensen et al., 2008]. The three sensor blocks are: one for the IMU

part, one for the FTSs, and finally one for the EPOSs.

To fetch measurement data from the sensors a function call to the AAU-BOT1 I/O module is utilized as the module always has the most resent measurement stored. When transmitting data to either of the buses this is also done using a function call to the AAU-BOT1 I/O module. This function call puts the message into a mailbox corresponding to the bus and then it is afterwards handled by the I/O module as described in the next section.

4.3 AAU-BOT1 I/O module

To serve all the CAN buses and the serial buses in the easiest way, this module starts two individual RTAI threads for each of the buses. By doing so, it is possible to utilize blocking calls for everything and not waste processing power to loop around to check several buses in the same thread. The reason for having two threads for each bus is to have one thread handling the receptions from the bus and another thread to handle the transmissions to the bus.

The threads handling the transmissions are almost similar. The flow in one of the tx threads is illustrated in Figure 4.2. First, the bus connection is initialized and secondly the rx thread

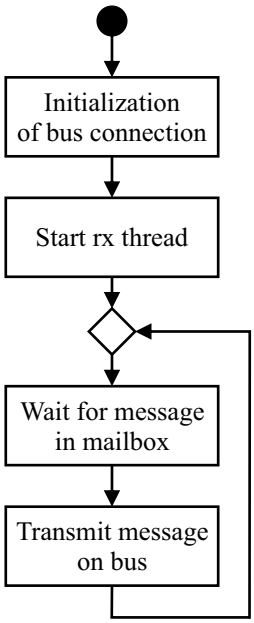


Figure 4.2: *The software flow of one of the transmit (tx) threads.*

corresponding to the tx thread is started. The reason for this is that it is ensured that the connection has been initialized before the rx thread is started. After the initialization procedure the tx thread enters a loop, in this loop it waits for messages in a mailbox, and on reception of something, it transmits it on the bus. All transmissions have a timeout value set and if for some reason the transmission of a message exceeds this timeout the thread prints an error message using rt_printk and continues operation. The mailbox receive call is a blocking call, meaning that until something is in the mailbox the thread is put to sleep by the real time scheduler. Furthermore, the bus transmission call can also be blocking, depending on the time it takes to transmit the message onto the bus.

The flow of one of the rx threads is shown in Figure 4.3. The thread consists only of a loop. In the loop it uses a blocking call to receive data from the bus, and on reception of something, it calls the protocol handler function, which is different for the EPOSs, the FTSs, and the IMU. On reception of measurement data these are stored in variables in the memory area related to the I/O module. From there the measurements can be fetched by the controller program using a function call.

The protocol handling of each of the EPOSs, the FTSs, and the IMU is handled by separate drivers for each. These drivers are made separately and then incorporated in the AAU-BOT1 I/O

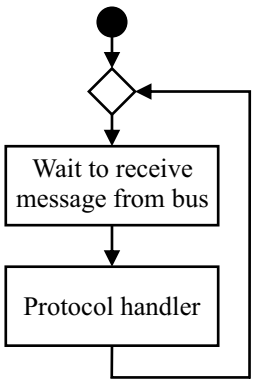


Figure 4.3: *The software flow of one of the receive (rx) threads.*

module. A description of each driver is given in the following three subsections.

4.3.1 FTS Driver

To utilize the FTSs the communication protocol for the strain gauge amplifiers must be obeyed. As written in [Jensen et al., 2008, p. 60] the amplifiers have 3 different protocol modes, where two of them are utilized in this project. The normal Lorenz mode is used during the setup of the amplifiers. This protocol mode uses a 0x02 to indicate the beginning of a packet, and therefore this byte is doubled whenever it appears elsewhere in the packet. Figure 4.4 illustrates how the protocol is structured, and each of the fields is explained in Table 4.1.

Figure 4.4: *Overview of the FTS protocol structure.*

Field	Field width	Description
PREAMBLE	1 byte	Indicator of start of packet (0x02)
CMD	1 byte	Command specification byte
RX ADR	1 byte	Address of the receiver of this packet
TX ADR	1 byte	Address of the transmitter of this packet
LEN	1 byte	Value equals number of bytes in DATA field
DATA	0-255 bytes	Data bytes
CHECKSUM	1 byte	Checksum of message
WCHECKSUM	1 byte	Weighted checksum of message

Table 4.1: *Explanation of each of the fields in the Lorenz protocol.*

The second protocol mode that is used is the Speed Optimized Streaming Mode (SOSM). In this mode each packet consists of 5 bytes each. The first byte is a synchronization byte, which is incremented by one for each packet, and when it reaches 255 it is reset to 0. The following two bytes are the measurement for channel A of the amplifier and the remaining two are the measurement for channel B, both sent as big-endian. In the streaming mode the start byte is also doubled, which is not necessary, but as it cannot be disabled it must be handled in the driver.

As determined by [Jensen et al., 2008, p. 62] it is not possible to use hardware triggering of the measurements, so jitter will always occur. To reduce the jitter, the sampling rate is set at 1,250 Hz which is the maximum possible sampling frequency when using two channels at the serial baud rate of 115,200. By doing so the maximum jitter is reduced to $1/1,250 = 800$ µs.

When the FTSs are running at full sampling rate there is a problem when they should be stopped again. The problem is related to the FTS protocol as it is not very robust. There is no easy way to identify measurement streaming messages against standard protocol messages, and the result is

that if something fails in the set up procedure, then the FTS and the computer do not agree upon the protocol mode and the messages are not received correctly. Furthermore, the high load on the serial bus when an FTS is running at full sampling rate makes it almost impossible to interrupt the message stream and bring the FTS out of streaming mode and back to the standard protocol mode.

4.3.2 IMU Driver

The communication protocol used by the IMU is rather simple; it utilizes a fixed message format without variations, contrary to the doubling of bytes in the FTS protocol. The message format is shown in Figure 4.5. To explain each of the fields they are listed in Table 4.2.

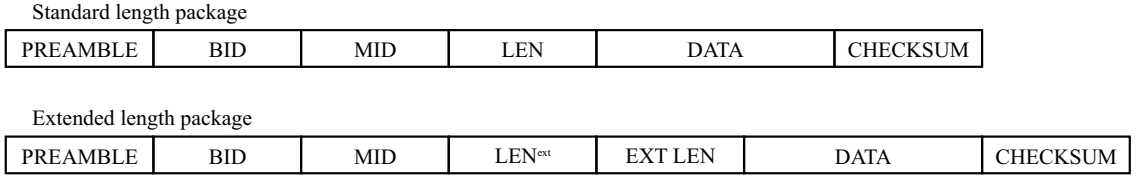


Figure 4.5: *Overview of the IMU protocol structure.*

Field	**Field width**	**Description**
PREAMBLE	1 byte	Indicator of start of packet (0xFA)
BID	1 byte	Bus identifier or Address (0xFF/0x01)
MID	1 byte	Message identifier
LEN	1 byte	*For standard length message:* Value equals number of bytes in DATA field. Maximum value is 254 (0xFE) *For extended length message:* Field value is always 255 (0xFF)
EXT LEN	2 bytes	The number of data bytes for extended length messages. Maximum value is 2048 (0x0800)
DATA (standard length)	0-254 bytes	Data bytes (optional)
DATA (extended length)	255-2048 bytes	Data bytes
CHECKSUM	1 byte	Checksum of message

Table 4.2: *Explanation of each of the fields in the IMU protocol [xsens, 2006, p. 4].*

The IMU can be put into different modes, but for the purpose of this project it is decided to use a streaming mode just as with the FTSs. The streaming frequency can be set to values ranging from 100 Hz to 512 Hz. Using the streaming mode it is possible to determine what should be sent each time. For now the IMU is set to transmit only the angles, but later it might be beneficial to also get accelerations.

4.3.3 EPOS Driver

To communicate with the EPOSs over the CAN buses a driver is used to handle the communication protocol. The protocol used by the EPOSs is the CANopen protocol [maxon motor, 2006, p. 19]. In this project it is not considered which of the communication procedures is the best, as this has already been analyzed by [Jensen et al., 2008, p. 57]. Their driver library is utilized, and improved to work with the developed AAU-BOT1 I/O module. Furthermore, extra commands have been added to the library as these are useful in the startup phase and for debugging purposes.

For the communication during control of the robot it was decided by [Jensen et al., 2008] to use PDO packages. To provide an overview of the used rxPDOs, they are all listed in Table 4.3, Table 4.4, Table 4.5, and Table 4.6. Note that the CAN ids have been changed compared to those given by [Jensen et al., 2008, p. 223], as it is decided by the project group to use a more logic numbering scheme, where the id relates to the first command element given in the package.

CAN bus	CAN id	Data length	Data	
5	0x202	8	$\dot{\theta}_2$	$\dot{\theta}_3$
	0x204	4	$\dot{\theta}_4$	
3	0x205	8	$\dot{\theta}_5$	$\dot{\theta}_6$
	0x207	8	$\dot{\theta}_7$	$\dot{\theta}_{18}$
2	0x208	8	$\dot{\theta}_8$	$\dot{\theta}_9$
	0x20A	8	$\dot{\theta}_{10}$	$\dot{\theta}_{19}$
4	0x20B	8	$\dot{\theta}_{11}$	$\dot{\theta}_{12}$
	0x20D	4	$\dot{\theta}_{13}$	
1	0x20F	8	$\dot{\theta}_{15}$	$\dot{\theta}_{16}$
	0x211	4	$\dot{\theta}_{17}$	

Table 4.3: *PDO frames sent from the OBC to the EPOS amplifiers to control the velocities (rxPDOs).*

CAN bus	CAN id	Data length	Data	
5	0x302	8	θ_2	θ_3
	0x304	4	θ_4	
3	0x305	8	θ_5	θ_6
	0x307	8	θ_7	θ_{18}
2	0x308	8	θ_8	θ_9
	0x30A	8	θ_{10}	θ_{19}
4	0x30B	8	θ_{11}	θ_{12}
	0x30D	4	θ_{13}	
1	0x30F	8	θ_{15}	θ_{16}
	0x311	4	θ_{17}	

Table 4.4: *PDO frames sent from the OBC to the EPOS amplifiers to control the angles (rxPDOs).*

CAN bus	CAN id	Data length	Data			
5	0x402	6	i_2	i_3	i_4	
3	0x405	8	i_5	i_6	i_7	i_{18}
2	0x408	8	i_8	i_9	i_{10}	i_{19}
4	0x40B	6	i_{11}	i_{12}	i_{13}	i_{14}
1	0x40F	6	i_{15}	i_{16}	i_{17}	

Table 4.5: *PDO frames sent from the OBC to the EPOS amplifiers to control the current (rxPDOs).*

The txPDOs utilized are not changed compared to those given by [Jensen et al., 2008], so they are only listed here, in Table 4.7, to complete the description.

CAN bus	CAN id	Data length	Data
1-5	0x501	2	Control word
1-5	0x000	2	Operational mode
1-5	0x080	0	

Table 4.6: *PDO frames sent from the OBC to all the EPOS amplifiers at once (rxPDOs). When receiving one of these, all of the EPOS amplifiers react upon it.*

CAN id	Data length	Data		
0x180+i	8	$\dot{\theta}_i$	$\theta_{\text{rel},i}$	
0x280+i	8	$\dot{\theta}_i$	i_i	$\theta_{\text{abs},i}$
0x380+i	8	$\theta_{\text{rel},i}$	i_i	$\theta_{\text{abs},i}$

Table 4.7: *PDO frames sent from the EPOS amplifiers to the OBC (txPDOs). The indices, i, indicates the EPOS id. One or more of these are sent by the EPOSs every time they receive a sync command, dependent on which of the PDOs that has been enabled.*

4.4 Operator Interaction

One of the essential features made to assist in the development and debugging of programs for **AAU-BOT1** is a comprehensive operator interaction program. The program is interfaced using a simple console menu and has all the functionalities needed to perform numerous tasks on the robot. A screenshot of the main menu of the interaction program is shown in Figure 4.6.

The interaction program is used after a power on to set up the FTSs and the IMU initially and start the measurement receiving from all of them. For the FTSs it is also possible to get a status of the streaming process, which has been a valuable feature during tests as the protocol used by the FTSs is not that robust, as elaborated on in Subsection 4.3.1 on page 31.

For debugging purposes it is possible to completely disable all communication on the CAN buses, i.e. all messages that should have been sent to the EPOSs are trashed without any other action. At very high loads the CAN buses can fail, even though they are rather robust. A failure on the CAN bus is registered and at any time a log can be fetched in the interaction program showing the count of the different types failures on the CAN bus. Note that at the sampling frequency of 250 Hz as used in this project the CAN buses practically never fails, as indicated by the network throughput test described in Appendix B on page 122.

The EPOSs/motors can also be controlled directly from the interaction program. The most important function is the quick stop command which, when executed, halts all motors immediately, which is useful if a dangerous situation should occur. Other implemented features for the EPOSs include resetting to normal state after an error situation, set up of EPOSs for any of its running modes, getting the current error log from the EPOSs, and moving all joints to their zero position. Also, to ease the determination of the correct zero position of the joints, they can all be locked and afterwards moved small bits at a time until the desired position is reached in all joints. When the robot is situated in the desired zero position a measurement of the absolute angles can be performed, describing the zero position.

Features that are directly tied to the programs generated from `MATLAB Simulink` are, enable/disable initial zero positioning, which controls if the robot should be moved to its absolute zero position when the control program is started, and "give order to go", which is needed when the control program is started in wait mode. This mode has been implemented in order to make it possible to ease the initialization of the control program. Using this feature the control program can be started and it initializes EPOSs and sensors, afterwards puts all joints in the desired initial position and holds it there using the EPOS position controllers, and then the control program waits to get the signal to proceed from the interaction program. This is very useful as the robot can be put down on the ground with the desired posture and the controllers started afterwards and avoiding any controller initialization problems.

The developed software structure has been tested in a coherent test of the network throughput. This

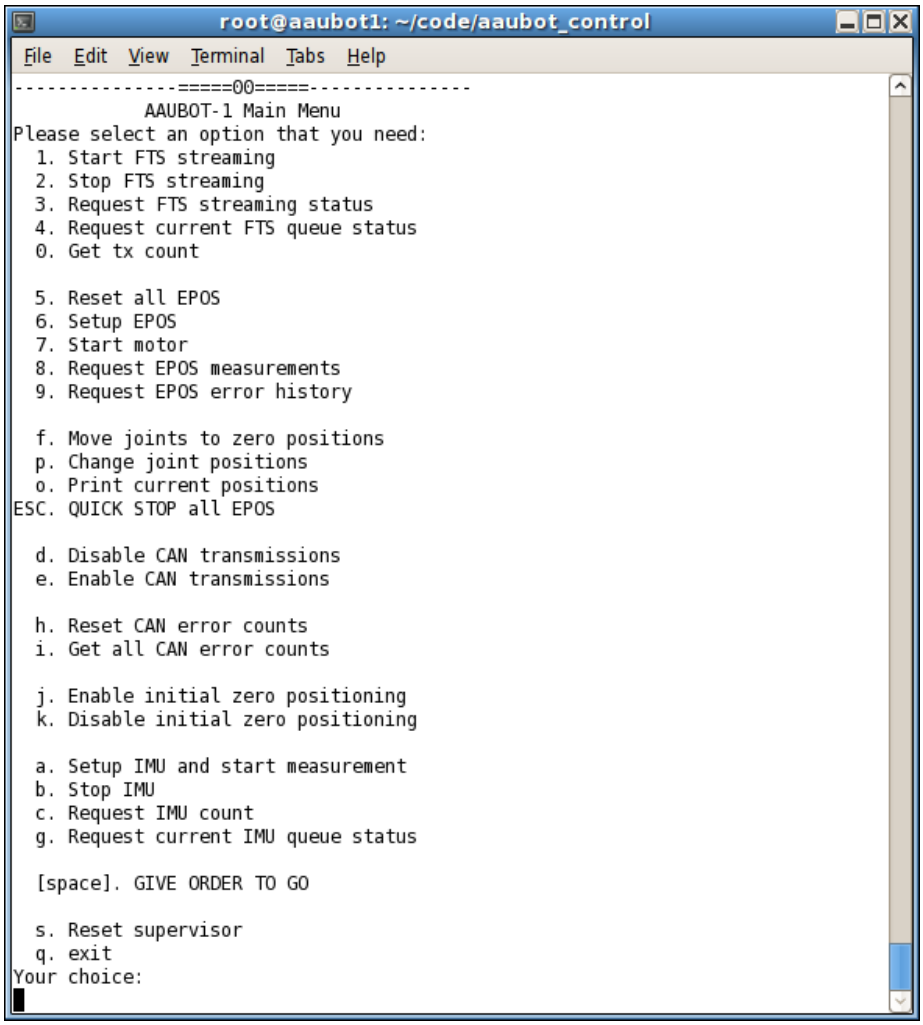


Figure 4.6: *Screenshot of the operator interaction program main menu.*

test includes the interface to Simulink, the kernel module, the developed PCI extension card driver, and the physical buses. The test and the result of it are described in Appendix B on page 122, and at a system frequency of 250 Hz the error rate is only 0.0017 % which is a significant improvement compared to the 13.6 % obtained in [Jensen et al., 2008, p. 221]. The developed `MATLAB Simulink` interface makes it easy to implement new controllers on **AAU-BOT1** with direct interface to the communication buses via the developed software platform.

5 Modeling

In this chapter a model of **AAU-BOT1** is set up, containing a kinematic and a dynamic model, together with an inverse kinematic model. The model is set up with background in the previous work done on **AAU-BOT1** and extended with the parts that are needed to make **AAU-BOT1** able to walk. The model is used to simulate the behavior of the robot, and planning the motion to achieve the desired gait pattern, as described in Chapter 7 on page 76. Finally, the model is used to design a suitable estimator and controller for the robot, described in Chapter 8 on page 87 and Chapter 6 on page 61, respectively. The implementation of the model together with the control system can be seen in the `Matlab Simulink` model located on the DVD in [`Model_with_Control_System/`].

Contents

5.1 Introduction to Modeling

To model a biped robot many different things must be taken into consideration to achieve a sufficiently accurate model. The necessary accuracy of the model depends on the purpose of the model, e.g. if the model is used for simulation purpose or control purpose. The two main parts of the model is the dynamic part and the kinematic part. The dynamic part models the dynamic behavior of the robot by taking an input torque for each joint and then determining the accelerations, velocities, and positions of the joints. The kinematic part maps the acceleration, velocity, and position of the joint angles into Cartesian coordinates for each CoM of the links.

Since **AAU-BOT1** is a long-term project there has been made some work on **AAU-BOT1** by the previous project groups. The robot is built at the department of mechanics by [Pedersen et al., 2007], who also made the first modeling of the robot. Their model is a single support model with the purpose of simulating the behavior of the robot. The dynamic part is modeled using Newton-Euler equations which is a strait forward method, that unfortunately becomes very complex when applied to a biped robot, and furthermore becomes very heavy computational wise.

The last group working on **AAU-BOT1** was [Jensen et al., 2008], who has developed a model which should be suitable to simulate a static walk of the robot. The model is however not complete to simulate an entire static gait cycle since a ground contact model is not included. The model had not been verified since it was not possible to get all necessary parameters. The dynamic part is based on Lagrange-d'Alembert equations which is more suitable for biped robots than the Newton-Euler approach.

The model derived in this project will be based on the work made by the two previous groups, that have worked on **AAU-BOT1**, and extended and modified to a degree necessary to make the robot walk both statically and dynamically.

The complete model is illustrated in Figure 5.1 and the different sub-models are listed and described in the following.

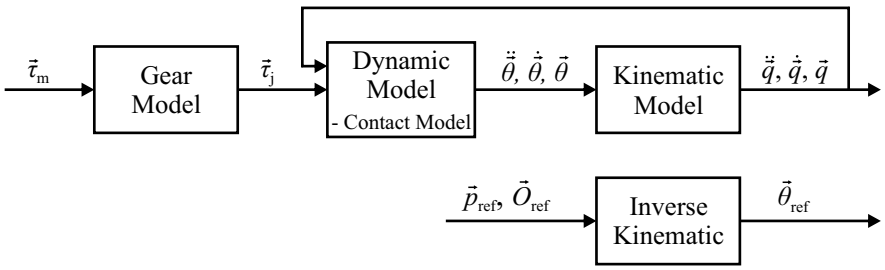


Figure 5.1: *Overview of the complete model illustrated by a block diagram with inputs and outputs to each sub model.*

- **Dynamic Model**
  The purpose of the dynamic model is to model the angular acceleration of each joint based on an applied torque to the joints. The dynamic model thus includes the inertia, friction, and mass of each link, along with the gravity force.
- **Contact Model**
  The contact model is a part of the dynamic model and is used to model the forces and torques that the ground exerts on the foot. The contact model also introduces a damping effect when the foot makes contact with the ground.
- **Kinematic Model**
  The kinematic model is used to calculate the global position of each joint and the CoM for each link in Cartesian coordinates. The global positions are calculated from the joint angles determined by the dynamic model. Besides the positions, the accelerations and velocities for each CoM are also calculated, based on the angular accelerations and velocities of the joints.
- **Inverse Kinematic**
  The inverse kinematic model is used to calculate the joint angles from requirements given to the position of the joints. The inverse kinematics is therefore not actually a part of the model of the robot, but is used in the generation of trajectories for the joints.

Having provided an overview of the model, the model will be derived in the following sections.

5.2 Kinematic Model

In this section the set up of the kinematic model will be described. Since a kinematic model for **AAU-BOT1** has already been devised by [Jensen et al., 2008, p. 77], the kinematic model will only be described shortly in this section with the main purpose of defining the representation of the robot. The purpose of the kinematic model is to be able to represent all the links of **AAU-BOT1** in global Cartesian coordinates. This is done by a transformation from each joint frame to the global frame using the joint angles. Each link is represented by its CoM point and thereby the entire robot is represented by a combination of the CoM of each link. The output from the kinematic model is the generalized coordinate vector, as shown in Figure 5.2. The content of the generalized coordinate vector is shown in Eq. (5.1).

$$\vec{q} = \begin{bmatrix} \vec{x} & \vec{y} & \vec{z} & \vec{\theta} \end{bmatrix}^T = \begin{bmatrix} x_1 \cdots x_{20} & y_1 \cdots y_{20} & z_1 \cdots z_{20} & \theta_1 \cdots \theta_{19} \end{bmatrix}^T \tag{5.1}$$

where:

$\vec{q}$ is the generalized coordinate vector
$\vec{x}$, $\vec{y}$, and $\vec{z}$ are the global coordinates for each link CoM [m]
$\vec{\theta}$ is the angle vector containing the joint angles [rad]

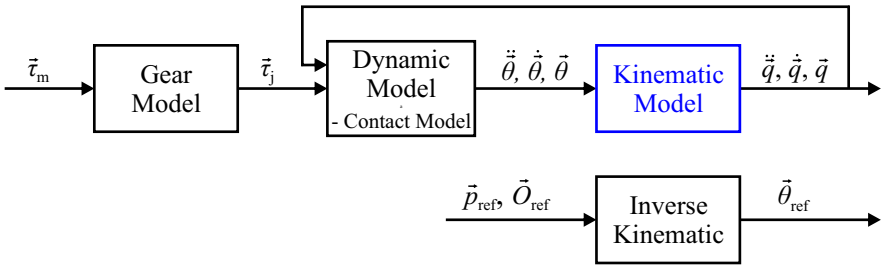


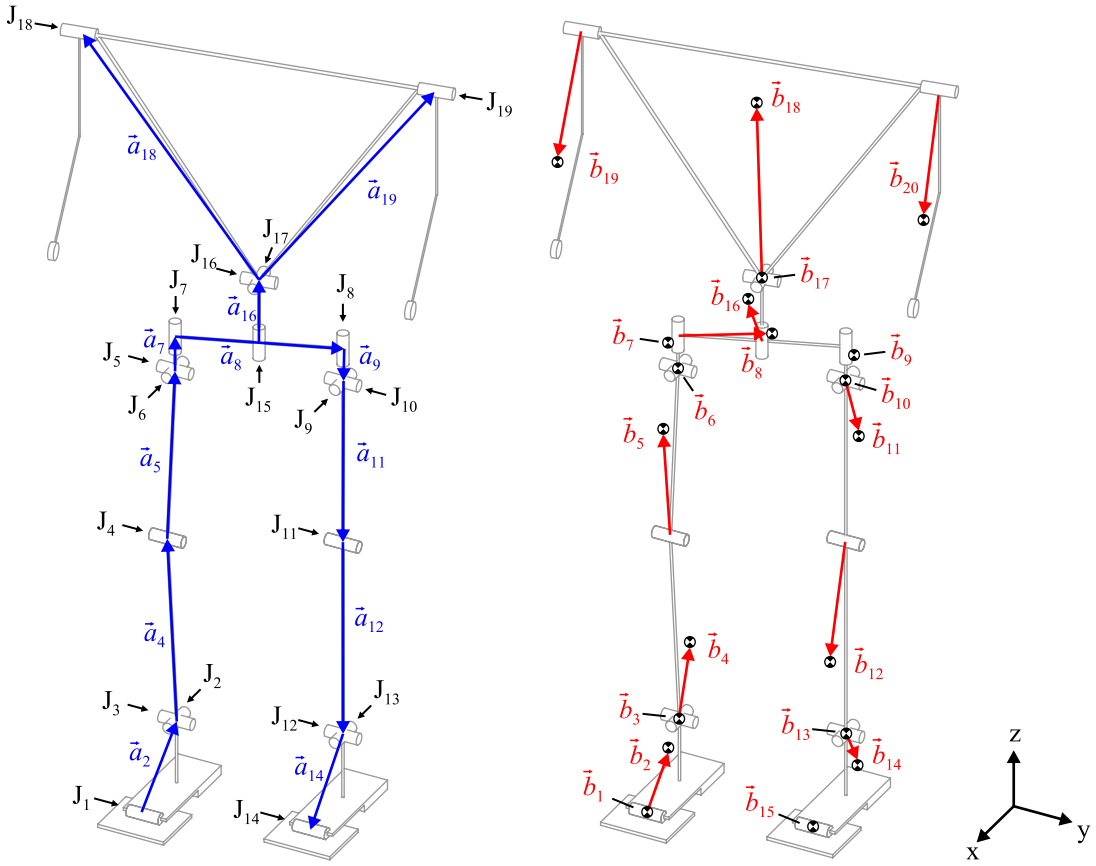
Figure 5.2: *The kinematic model takes the joint angles as input, and returns the generalized coordinate vector containing the global position of each CoM and the joint angles.*

The kinematic model must be able to describe the robot in all the different phases described in Section 2.3 on page 14.

In the following the representation of the mechanical structure is defined.

5.2.1 Representation of the Mechanical Structure

The mechanical system of **AAU-BOT1** consists of joints and links connected in a unique structure to represent a humanoid robot. The representation of **AAU-BOT1** is based on the situation where the robot is in SSP-R, thus the mechanical structure can be seen as a number of kinematic chains starting at the toe of the supporting foot. The position of the links, link vectors, and CoM vectors are illustrated in Figure 5.3. The link and CoM vectors are extracted from the `Solidworks` drawing of **AAU-BOT1**, which was originally made by [Pedersen et al., 2007] and has been modified to correspond to the physical robot by this project group. The vectors are presented in Table 5.1.



(a) Defining the joints J and link vectors a.

(b) Illustration of the CoM vectors for each link.

Figure 5.3: *Illustration of the position of the links, link vectors, and CoM vectors.*

Link vectors [m]				CoM vectors [m]			
$\vec{a}_1$ =[	0.000	0.000	$0.000]^T$	$\vec{b}_1$ =[	0.000	0.000	$-0.014]^T$
$\vec{a}_2$ =[	-0.128	0.000	$0.093]^T$	$\vec{b}_2$ =[	-0.151	-0.011	$0.053]^T$
$\vec{a}_3$ =[	0.000	0.000	$0.000]^T$	$\vec{b}_3$ =[	-0.006	0.006	$0.000]^T$
$\vec{a}_4$ =[	0.000	0.000	$0.370]^T$	$\vec{b}_4$ =[	-0.002	0.032	$0.124]^T$
$\vec{a}_5$ =[	0.000	0.012	$0.310]^T$	$\vec{b}_5$ =[	-0.002	0.034	$0.166]^T$
$\vec{a}_6$ =[	0.000	0.000	$0.000]^T$	$\vec{b}_6$ =[	-0.002	-0.007	$0.000]^T$
$\vec{a}_7$ =[	-0.003	0.000	$0.064]^T$	$\vec{b}_7$ =[	-0.050	-0.028	$0.044]^T$
$\vec{a}_8$ =[	0.000	0.280	$0.000]^T$	$\vec{b}_8$ =[	-0.022	0.140	$0.016]^T$
$\vec{a}_9$ =[	0.003	0.000	$-0.064]^T$	$\vec{b}_9$ =[	-0.046	0.028	$-0.021]^T$
$\vec{a}_{10}$=[	0.000	0.000	$0.000]^T$	$\vec{b}_{10}$=[	-0.002	0.007	$0.000]^T$
$\vec{a}_{11}$=[	0.000	0.012	$-0.310]^T$	$\vec{b}_{11}$=[	-0.002	0.046	$-0.144]^T$
$\vec{a}_{12}$=[	0.000	0.000	$-0.370]^T$	$\vec{b}_{12}$=[	-0.002	-0.031	$-0.245]^T$
$\vec{a}_{13}$=[	0.000	0.000	$0.000]^T$	$\vec{b}_{13}$=[	0.006	-0.006	$0.000]^T$
$\vec{a}_{14}$=[	0.128	0.000	$-0.093]^T$	$\vec{b}_{14}$=[	-0.024	0.011	$-0.040]^T$
$\vec{a}_{15}$=[	0.000	0.000	$0.000]^T$	$\vec{b}_{15}$=[	0.000	0.000	$-0.015]^T$
$\vec{a}_{16}$=[	0.000	0.000	$0.150]^T$	$\vec{b}_{16}$=[	-0.004	-0.038	$0.109]^T$
$\vec{a}_{17}$=[	0.000	0.000	$0.000]^T$	$\vec{b}_{17}$=[	-0.005	-0.003	$0.000]^T$
$\vec{a}_{18}$=[	0.005	-0.305	$0.419]^T$	$\vec{b}_{18}$=[	-0.007	-0.005	$0.281]^T$
$\vec{a}_{19}$=[	0.005	0.305	$0.419]^T$	$\vec{b}_{19}$=[	0.096	0.000	$-0.238]^T$
				$\vec{b}_{20}$=[	0.096	0.000	$-0.238]^T$

Table 5.1: *The link and CoM vectors extracted from the drawing of* **AAU-BOT1** *made in* `Solidworks`*. Note that the vectors are different from those provided by the previous project groups. The link vectors are different as the previous groups apparently had extracted the wrong parameters from* `Solidworks`*, while the CoM vectors are different simply because the drawings have been updated.*

The body frame in which the entire robot is represented is fixed at the toe joint at the supporting foot, meaning that the body frame is alternating between the right foot and the left foot. The global frame is fixed to the ground and is used to keep track of the position of the body frame; the global frame is fixed at the initial location of the body frame.

The kinematics of **AAU-BOT1** is modeled as three open loop kinematic chains, where each chain starts at the supporting foot and ends at the last joint in the chain. The kinematic chains for SSP-R and SSP-L are shown in Table 5.2 and Table 5.3, respectively.

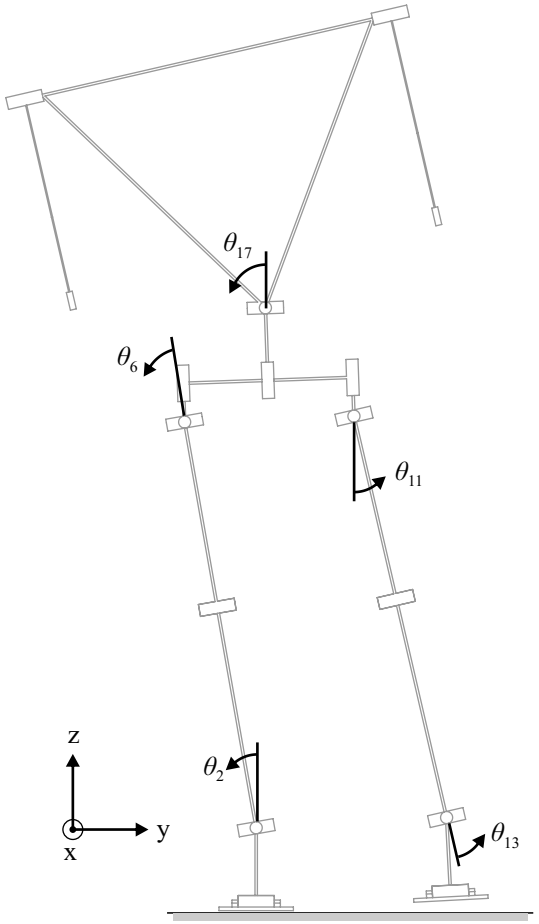
Kinematic chain	Link no.
1,2,3,4,5,6,7,8,9,10,11,12,13,14	$i \leq 14$
1,2,3,4,5,6,7,15,16,17,18	$15 \leq i \leq 18$
1,2,3,4,5,6,7,15,16,17,19	$i = 19$

Table 5.2: *The kinematic chains for SSP-R.*

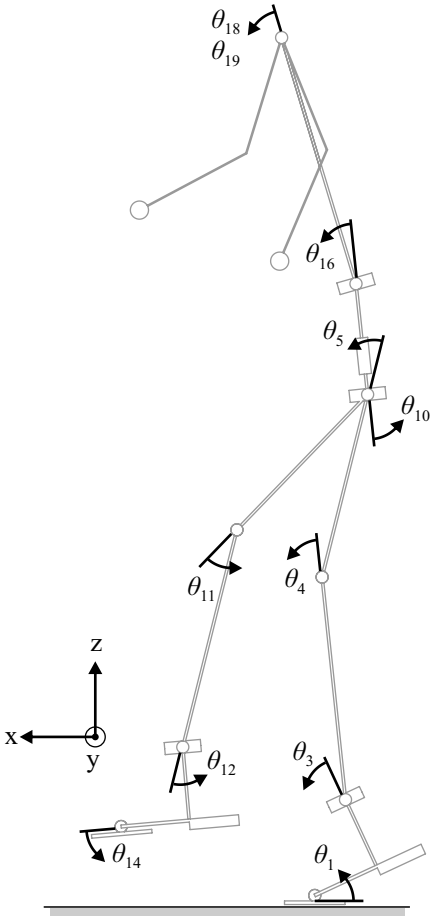
Kinematic chain	Link no.
14,13,12,11,10,9,8,7,6,5,4,3,2,1	$i \leq 14$
14,13,12,11,10,9,8,15,16,17,18	$15 \leq i \leq 18$
14,13,12,11,10,9,8,15,16,17,19	$i = 19$

Table 5.3: *The kinematic chains for SSP-L.*

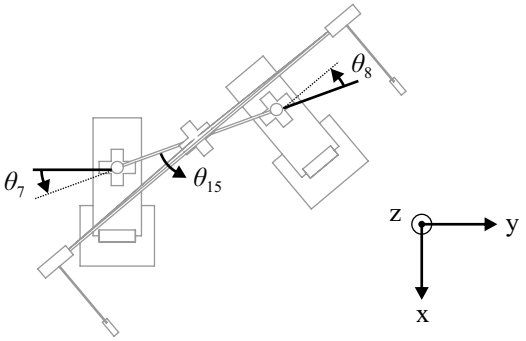
The joint angles are defined to have counter clockwise positive direction as illustrated in Figure 5.4.



(a) Angles in frontal plane, i.e. roll angles.



(b) Angles in sagittal plane, i.e. pitch angles.



(c) Angles in transverse plane, i.e. yaw angles.

Figure 5.4: *Illustration of the locations of the joint angles and their positive directions in the three planes. Note that the position of the robot is not one it will achieve during walk, but is only used for illustrative purposes.*

Having defined the representation of **AAU-BOT1** the transformation from each joint frame to the body frame is made in the following.

5.2.2 Joint Frame Transformation

The joint frames are defined in such a way that when all the joint angles are zero the joint frames have the same orientation as the body frame. This has the advantage that the orientation between two consecutive joint frames is described by the actual joint angle only. Since all the joints only rotate around one of the three Cartesian axes, the transformation from a joint frame to the body frame can be made by use of rotation matrices only.

The rotation matrices for rotation around the x-, y-, and z-axis are given as shown in Eq. (5.2), Eq. (5.3), and Eq. (5.4), respectively [Craig, 2005, p. 46].

$$ {}^{i-1}_{i}\boldsymbol{R}_{\mathrm{x}}(\theta_i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_i & -\sin\theta_i \\ 0 & \sin\theta_i & \cos\theta_i \end{bmatrix} \tag{5.2} $$

$$ {}^{i-1}_{i}\boldsymbol{R}_{\mathrm{y}}(\theta_i) = \begin{bmatrix} \cos\theta_i & 0 & \sin\theta_i \\ 0 & 1 & 0 \\ -\sin\theta_i & 0 & \cos\theta_i \end{bmatrix} \tag{5.3} $$

$$ {}^{i-1}_{i}\boldsymbol{R}_{\mathrm{z}}(\theta_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.4} $$

When rotating around several joints the rotation matrices for each joint can be multiplied into one rotation matrix as shown in Eq. (5.5).

$$ {}^{0}_{n}\boldsymbol{R}(\theta_1, \ldots, \theta_n) = {}^{0}_{1}\boldsymbol{R}(\theta_1)\ {}^{1}_{2}\boldsymbol{R}(\theta_2)\ \ldots\ {}^{n-1}_{n}\boldsymbol{R}(\theta_n) \tag{5.5} $$

By use of a rotation matrix, the CoM vector, and the joint frame origin, the CoM point for each link can be transformed into body frame coordinates, as shown in Eq. (5.6).

$$ {}^{0}\vec{p}_i = {}_{i-1}{}^{0}\boldsymbol{R}\vec{b}_i + {}^{0}\vec{p}_{\mathrm{j},i\text{-}1} \tag{5.6} $$

where:

$\vec{b}_i$ is the CoM vector for link i seen in frame $i-1$ [m]
${}^{0}\vec{p}_i$ is the CoM point for link i in the body frame [m]
${}^{0}\vec{p}_{\mathrm{j},i\text{-}1}$ is the origin of the frame in joint $i-1$ seen in the body frame [m]
${}^{0}_{i-1}\boldsymbol{R}$ is the rotation matrix for joint $i-1$ into the body frame

The coordinates for the origin of each joint frame seen in the body frame is determined using the link vectors as shown in Eq. (5.7).

$$ {}^{0}\vec{p}_{\mathrm{j},i} = {}_{i-1}{}^{0}\boldsymbol{R}\vec{a}_i + {}^{0}\vec{p}_{\mathrm{j},i\text{-}1} \tag{5.7} $$

where:

$\vec{a}_i$ is the link vector from joint $i-1$ to i [m]

The velocity and acceleration vectors are given as shown in Eq. (5.8) and Eq. (5.9).

$$ \dot{\vec{q}} = \frac{\partial \vec{q}}{\partial t} \tag{5.8} $$

$$ \ddot{\vec{q}} = \frac{\partial^2 \vec{q}}{\partial^2 t} \tag{5.9} $$

Having set up the kinematic model the global position, velocity, and acceleration can be determined.

5.3 Dynamic Models

In this section the set up of the dynamic model will be described. The purpose of the dynamic model is to calculate, for all joints, the angle and the first and second derivatives of it. The calculations are based on the link positions and the first and second derivative of it, along with the torque applied to the joints by the DC motors. The inputs and outputs from the dynamic model are illustrated in Figure 5.5.

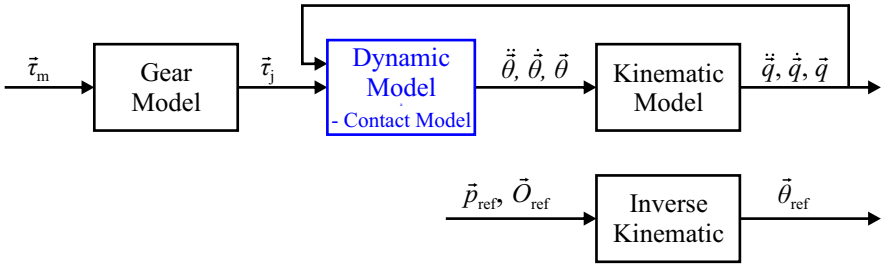


Figure 5.5: *The dynamic model takes as input: the position of the CoMs in global coordinates and the torque in each joint. The output is: the angle, angular velocity, and angular acceleration of each joint.*

In the following subsection the set up of the dynamic model is described. In total two different dynamic models are set up, a complete simulation model and a simplified version used in the controller. The procedure described here is utilized for each model to obtain both dynamic models with only a few minor changes which are described in the summary of the dynamic model presented in Subsection 5.3.4 on page 49. The difference between the models is illustrated in Figure 5.6, and is that the simulation model has its origin in the center of the pelvis, while the control system model has its origin(s) in the feet.

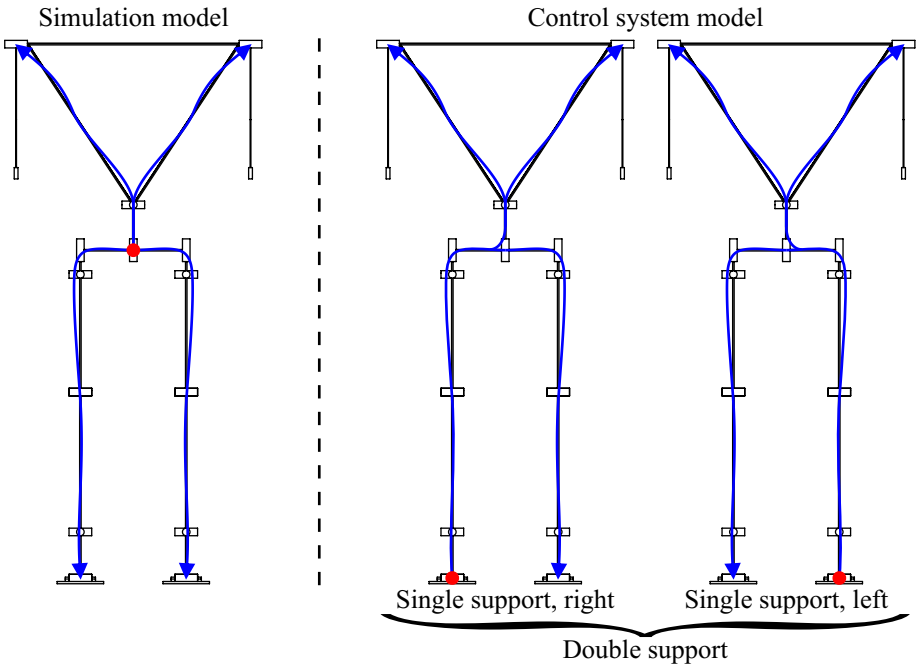


Figure 5.6: *Overview of the difference between the models utilized in the project. The red dot illustrates the origin of the model, and the blue arrows are the kinematics chains, all starting at the origin.*

5.3.1 Set Up Procedure for the Dynamic Models

The modeling is accomplished using Lagrange modeling, and has its starting point in the Lagrange-d'Alembert equation [Craig, 2005, p. 183]:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\vec{q}}}\right) - \frac{\partial \mathcal{L}}{\partial \vec{q}} = \mathcal{F} \tag{5.10}$$

where:

$\mathcal{F}$ is the external forces for **AAU-BOT1** in SSP
$\mathcal{L}$ is the Lagrangian of **AAU-BOT1** in SSP

The reason for choosing the Lagrange modeling approach contrary to a Newton-Euler approach is that the Lagrange model is simpler to set up and maintain for larger systems as **AAU-BOT1**. Furthermore, the Lagrange approach is the method most often used in biped modeling. The basic principle of the Lagrange modeling is an energy-based approach to the robot dynamics. The first to calculate is the Lagrangian defined as the difference between the kinetic and the potential energy of the system.

The Lagrangian is determined using the following relation:

$$\mathcal{L} = E_{\text{kin}} - E_{\text{pot}} \tag{5.11}$$

where:

E_{kin} is the kinetic energy of the system [J]
E_{pot} is the potential energy of the system [J]

As stated in [Jensen et al., 2008, p. 86] the toe springs are modeled as potential energy approximated by a constant times the toe angle, which results in the following equation for calculating the full potential energy [Craig, 2005, p. 183]:

$$E_{\text{pot}} = \sum_{i=1}^{n} (m_i g z_i) + k_{\text{t}} (\theta_1 + \theta_{14}) \tag{5.12}$$

where:

g is the gravitational acceleration constant [m/s^2]
$m_i$is the mass of link i [kg]
n is the number of links, i.e. 20 for **AAU-BOT1** [·]
z_i is the z-coordinate of the CoM of link i [m]
k_{t} is a constant [J/rad]

The kinetic energy is calculated using the following equation [Craig, 2005, p. 182]:

$$E_{\text{kin}} = \frac{1}{2} \sum_{i=1}^{n} \left(m_i \dot{p}_i^T \dot{p}_i + \vec{\omega}_i^T \boldsymbol{I}_i \vec{\omega}_i \right) \tag{5.13}$$

where:

$\boldsymbol{I}_i$ is the inertia tensor of link i around its CoM [kgm^2]
$\vec{\omega}_i$ is the angular velocity vector of link i [rad]

The angular velocities can be determined using [Craig, 2005, p. 146]:

$$\vec{\omega}_i = \vec{\omega}_{i-1} + {}^0_i\boldsymbol{R}\dot{\theta}_i\vec{\zeta}_i \tag{5.14}$$

where:

$\vec{\zeta}_i$ is a vector that denotes the rotational axis of $\dot{\theta}_i$ [·]

The diagonal elements of the inertia tensor of each of the links can be found in Table 5.4. Only the diagonal elements of the inertia tensor are provided as the off-diagonal elements are assumed to be 0 in order to reduce the number of calculations needed. Making this assumption means that

the principal axes of each of the links are assumed to follow the used coordinate systems. By looking at the full inertia tensors determined using `Solidworks`, it is seen that for most of the links the off-diagonal elements are insignificantly small compared to the diagonal elements. For the remaining links later experiments may show if it is beneficial to include the full inertia tensor in the calculations for these links.

Note that before extracting the mass and inertia parameters the `Solidworks` model has been updated to correspond to the current set-up of **AAU-BOT1** with the additions made since the drawings were last edited. The updates include changing the EPOSs to the correct model, along with the positioning of them. Furthermore, the FTS amplifiers have been added.

	Inertia elements $[\mathbf{kgm^2}]$			**Mass [kg]**
Right toe	$[I_{xx,1}\ I_{yy,1}\ I_{zz,1}] = [0.00033$	0.00015	$0.00047]$	$m_1 = 0.172$
Right foot	$[I_{xx,2}\ I_{yy,2}\ I_{zz,2}] = [0.0062$	0.0130	$0.0114\ ]$	$m_2 = 2.321$
Right ankle cross axle	$[I_{xx,3}\ I_{yy,3}\ I_{zz,3}] = [0.00017$	0.00034	$0.00040]$	$m_3 = 0.470$
Right shin	$[I_{xx,4}\ I_{yy,4}\ I_{zz,4}] = [0.0727$	0.0725	$0.0129\ ]$	$m_4 = 5.275$
Right thigh	$[I_{xx,5}\ I_{yy,5}\ I_{zz,5}] = [0.0910$	0.0926	$0.0238\ ]$	$m_5 = 6.656$
Right hip cross axle	$[I_{xx,6}\ I_{yy,6}\ I_{zz,6}] = [0.00043$	0.00032	$0.00061]$	$m_6 = 0.640$
Right hip bracket	$[I_{xx,7}\ I_{yy,7}\ I_{zz,7}] = [0.0158$	0.0120	$0.0088\ ]$	$m_7 = 3.052$
Pelvis	$[I_{xx,8}\ I_{yy,8}\ I_{zz,8}] = [0.0525$	0.0122	$0.0629\ ]$	$m_8 = 4.689$
Left hip bracket	$[I_{xx,9}\ I_{yy,9}\ I_{zz,9}] = [0.0158$	0.0120	$0.0088\ ]$	$m_9 = 3.052$
Left hip cross axle	$[I_{xx,10}\ I_{yy,10}\ I_{zz,10}] = [0.00043$	0.00032	$0.00061]$	$m_{10} = 0.640$
Left thigh	$[I_{xx,11}\ I_{yy,11}\ I_{zz,11}] = [0.0910$	0.0926	$0.0238\ ]$	$m_{11} = 6.656$
Left shin	$[I_{xx,12}\ I_{yy,12}\ I_{zz,12}] = [0.0729$	0.0727	$0.0129\ ]$	$m_{12} = 5.279$
Left ankle cross axle	$[I_{xx,13}\ I_{yy,13}\ I_{zz,13}] = [0.00017$	0.00034	$0.00040]$	$m_{13} = 0.470$
Left foot	$[I_{xx,14}\ I_{yy,14}\ I_{zz,14}] = [0.0062$	0.0130	$0.0114\ ]$	$m_{14} = 2.321$
Left toe	$[I_{xx,15}\ I_{yy,15}\ I_{zz,15}] = [0.00033$	0.00015	$0.00047]$	$m_{15} = 0.172$
Waist bracket	$[I_{xx,16}\ I_{yy,16}\ I_{zz,16}] = [0.0094$	0.0066	$0.0055\ ]$	$m_{16} = 2.167$
Waist cross axle	$[I_{xx,17}\ I_{yy,17}\ I_{zz,17}] = [0.00034$	0.00094	$0.00119]$	$m_{17} = 0.754$
Torso	$[I_{xx,18}\ I_{yy,18}\ I_{zz,18}] = [0.6378$	0.3674	$0.3748\ ]$	$m_{18} = 11.681$
Right arm	$[I_{xx,19}\ I_{yy,19}\ I_{zz,19}] = [0.0133$	0.0291	$0.0161\ ]$	$m_{19} = 0.906$
Left arm	$[I_{xx,20}\ I_{yy,20}\ I_{zz,20}] = [0.0133$	0.0291	$0.0161\ ]$	$m_{20} = 0.906$

Table 5.4: *The diagonal elements of the inertia tensors and the mass of each link extracted from the drawing of* **AAU-BOT1** *made in* `Solidworks`.

The inertia tensor elements and the masses retrieved from the `Solidworks` model and presented in Table 5.4 are not the actual values used in the model, as a weighing of the actual robot showed a weight larger than that of the `Solidworks` model. This weight increase is as expected because wires and other small components are not included in the `Solidworks` model. The total weight of the `Solidworks` model is 58.275 kg and the weighing of the physical robot showed 64.2 kg. Due to the fact that most of the wires and additional small parts are placed at the thighs, shins, and torso only the mass and inertia of these are changed. The additional mass is distributed relative to the current weight of the affected parts. The inertia of the affected parts is increased by the same factor as the mass, assuming equal distribution of the additional mass.

Now, the Jacobian is used in order to map Eq. (5.10) to each of the actuators [Craig, 2005, p. 186]:

$$\begin{aligned}\vec{\tau} &= \boldsymbol{J}^T(\vec{\theta})\mathcal{F} \\ &= \boldsymbol{J}^T(\vec{\theta})\left(\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\vec{q}}}\right) - \frac{\partial \mathcal{L}}{\partial \vec{q}}\right)\end{aligned} \tag{5.15}$$

where:

$\boldsymbol{J}(\vec{\theta})$ is the Jacobian of **AAU-BOT1**

$\vec{\tau}$ is the sum of torques acting on the joint [Nm]

The Jacobian is calculated using the following equation [Craig, 2005, p. 149]. Note that before calculating the derivatives, the kinematic model must be inserted in place of q_i.

$$\begin{aligned} \boldsymbol{J}(\vec{\theta}) &= \frac{\partial \vec{q}}{\partial \vec{\theta}} \\ &= \begin{bmatrix} \frac{\partial q_1}{\partial \theta_1} & \cdots & \frac{\partial q_1}{\partial \theta_{19}} \\ \vdots & \ddots & \vdots \\ \frac{\partial q_{79}}{\partial \theta_1} & \cdots & \frac{\partial q_{79}}{\partial \theta_{19}} \end{bmatrix} \end{aligned} \tag{5.16}$$

The dynamic equation in Eq. (5.15) can, when it has been derived, be written on the general form [Craig, 2005, pp. 185, 180]:

$$\vec{\tau} = \boldsymbol{M}(\vec{\theta})\ddot{\vec{\theta}} + \vec{V}(\vec{\theta}, \dot{\vec{\theta}}) + \vec{G}(\vec{\theta}) \tag{5.17}$$

where:

$\boldsymbol{M}(\vec{\theta})$ is the mass matrix of the robot
$\vec{V}(\vec{\theta}, \dot{\vec{\theta}})$ is the velocity vector which contains centrifugal and Coriolis terms
$\vec{G}(\vec{\theta})$ is the vector containing all gravity terms

Lastly, the angular acceleration vector, $\ddot{\vec{\theta}}$, is isolated to obtain the following equation:

$$\ddot{\vec{\theta}} = \boldsymbol{M}^{-1}(\vec{\theta}) \left(\vec{\tau} - \vec{V}(\vec{\theta}, \dot{\vec{\theta}}) - \vec{G}(\vec{\theta}) \right) \tag{5.18}$$

It should be noted that the way the $\boldsymbol{M}$ matrix is formed ensures that it is invertible as long as all links have a mass and inertia contribution.

5.3.2 Friction Force Model

To get a realistic behavior of the model frictions must be included, meaning the exerted torque, τ, in Eq. (5.18) needs to be expressed as explained here. First the load torque is defined as stated in Eq. (5.19) and the exerted torque is expressed as in Eq. (5.20).

$$\vec{\tau}_\mathrm{l} = \vec{V}(\vec{\theta}, \dot{\vec{\theta}}) + \vec{G}(\vec{\theta}) \tag{5.19}$$

$$\vec{\tau} = \vec{\tau}_\mathrm{j} - \vec{\tau}_\mathrm{f} \tag{5.20}$$

where:

$\vec{\tau}_\mathrm{f}$ is the friction torque in each joint [Nm]
$\vec{\tau}_\mathrm{l}$ is the load torque in each joint [Nm]
$\vec{\tau}_\mathrm{j}$ is the torques applied to the joint by the DC motors and gears [Nm]

The friction forces, $\vec{\tau}_\mathrm{f}$, can be calculated according to the equation below, if the cross couplings can be neglected [Andersen and Pedersen, 2007, p. 10]. Note that the equation is set up for a single joint, as this has to be performed individually for each joint.

$$\tau_\mathrm{f} = \begin{cases} \dot{\theta}\mu + \mathrm{sign}(\dot{\theta})\tau_\mathrm{c} & \text{if } \dot{\theta} \neq 0 \\ \tau_\mathrm{j} - \tau_\mathrm{l} & \text{if } \dot{\theta} = 0 \wedge \mathrm{abs}(\tau_\mathrm{j} - \tau_\mathrm{l}) \leq \tau_\mathrm{c} + \tau_\mathrm{s} \\ \mathrm{sign}(\tau_\mathrm{j} - \tau_\mathrm{l})(\tau_\mathrm{c} + \tau_\mathrm{s}) & \text{if } \dot{\theta} = 0 \wedge \mathrm{abs}(\tau_\mathrm{j} - \tau_\mathrm{l}) > \tau_\mathrm{c} + \tau_\mathrm{s} \end{cases} \tag{5.21}$$

where:

μ is the viscous friction [Nms]
τ_c is the Coulomb friction [Nm]
τ_s is the stiction [Nm]

However, the cross couplings in **AAU-BOT1** have very significant influence on the system, meaning that a more advanced procedure need to be followed in order to calculate the friction forces.

The utilized procedure is proposed by this project group and is best illustrated by an example. Assume that the following conditions are present at some sample time, k, for a simple system with only 3 joints:

$$\dot{\vec{\theta}}(k-1) = [\ 0.05 \quad 0.01 \quad 0\ ]^T \quad \dot{\vec{\theta}}(k) = [\ 0.1 \quad -0.01 \quad 0\ ]^T$$
$$\vec{\tau}_\mathrm{j}(k) = [\ 4.5 \quad 0 \quad 0\ ]^T \qquad \vec{\tau}_\mathrm{l}(k) = [\ 1 \quad 1 \quad 1\ ]^T \qquad \tau_\mathrm{c} = 3 \quad \mu = 1$$
$$\boldsymbol{M} = \begin{bmatrix} 1 & 0.5 & -0.5 \\ 0.5 & 1 & -0.5 \\ -0.5 & -0.5 & 1 \end{bmatrix} \qquad \boldsymbol{M}^{-1} = \begin{bmatrix} 1.5 & -0.5 & 0.5 \\ -0.5 & 1.5 & 0.5 \\ 0.5 & 0.5 & 1.5 \end{bmatrix}$$

This situation indicates that the first joint is currently applied a torque from the motor and is accelerating, while the second joint has just changed running direction and the third joint is not moving. The rest of the parameters are chosen just to give some easy numbers in the example.

In the following pseudo code example, the parts to the right of the brackets are the results of the parts to the left of the brackets.

The first point is to determine the friction contribution in the joints that are moving and have not changed direction since the previous sample. This is because it is certain that these joints should continue to move while the status of the rest cannot be determined initially.

```
if θ̇_i(k) ≠ 0
  if sign(θ̇_i(k)) == sign(θ̇_i(k−1))
    τ_f,i = sign(θ̇_i(k))τ_c + θ̇_i(k)μ
  else
    τ_f,i = 0
  end
else
  τ_f,i = 0
end
```
$\Big\} \vec{\tau}_\mathrm{f} = [\ 3.1 \quad 0 \quad 0\ ]^T$

The second part is to calculate the angular acceleration contribution of the currently known torque. The currently known torque is the load torque, the torque exerted by the motors in the joints, and the friction contribution in moving joints with no direction change.

$$\ddot{\vec{\theta}}_\mathrm{t} = \boldsymbol{M}^{-1}\left(\vec{\tau}_\mathrm{j}(k) - \vec{\tau}_\mathrm{l}(k) - \vec{\tau}_\mathrm{f}\right) \Big\} \ddot{\vec{\theta}}_\mathrm{t} = [\ 0.6 \quad -2.2 \quad -1.8\ ]^T$$

Thirdly, the torque contribution required to negate the angular acceleration contribution in non-moving joints and joints with direction change. This is calculated by using the inverse of the part of $\boldsymbol{M}^{-1}$ which corresponds to the rows and columns of these joints. In the example this is the bottom-right part of the matrix.

$$\vec{\tau}_\mathrm{f,hold}[2:3] = (\boldsymbol{M}^{-1}[2:3, 2:3])^{-1}\ddot{\vec{\theta}}_\mathrm{t}[2:3] \Big\} \vec{\tau}_\mathrm{f,hold} = [\ * \quad -1.2 \quad -0.8\ ]^T$$

Next, if the contribution calculated to hold the joint at a stand still is larger than the coulomb friction plus the stiction, then the joint should continue to move if it changed direction and start to move if it was standing still. Hence, the friction contribution is set to fit the situation. Otherwise, the friction contribution is set to the value calculated to hold the joint.

```
if abs(τ_f,hold,i) > (τ_c + τ_s)
  if sign(θ̇_i(k)) == 0
    τ_f,i = sign(θ̇_i(k))(τ_c + τ_s)
  else
    τ_f,i = sign(θ̇_i(k))τ_c + θ̇_i(k)μ
  end
else
  τ⃗_f,i = τ_f,hold,i
end
```
$\Big\} \tau_\mathrm{f} = [\ 3.1 \quad -1.2 \quad -0.8\ ]^T$

Lastly, the angular acceleration of all joints are calculated based on the torques determined.

$$\left.\ddot{\vec{\theta}} = \boldsymbol{M}^{-1}(\vec{\tau}_{\mathrm{j}}(k) - \vec{\tau}_{\mathrm{l}}(k) - \vec{\tau}_{\mathrm{f}}) \right\} \ddot{\vec{\theta}} = [\ 0.4 \quad 0 \quad 0\ ]^T$$

The proposed algorithm has proven to be effective in all tested situations.

5.3.3 Ground Contact Model

To have a correct model of the robot it is necessary to model the interaction with the floor as this affects the movement of the entire robot. To get the most suitable model the robot is modeled as a free body which is affected by a proper number of springs when it is in contact with the floor as this is the method which has proven to be most effective. The actual modeling of ground reaction forces using springs is to some degree similar to the method utilized in the simulation of the biped robot Johnnie [Buschmann et al., 2006].

When modeling the robot as a free body with springs in the feet, the model is able to handle both single support and double support. Other approaches to model the robot in double support has been tried out, these approaches are described in Appendix D on page 126. The tested methods all showed not to be as suitable for modeling the robot as the method described here.

The modeling set-up of the robot, including the modeling of the ground reaction forces, is shown in Figure 5.7.

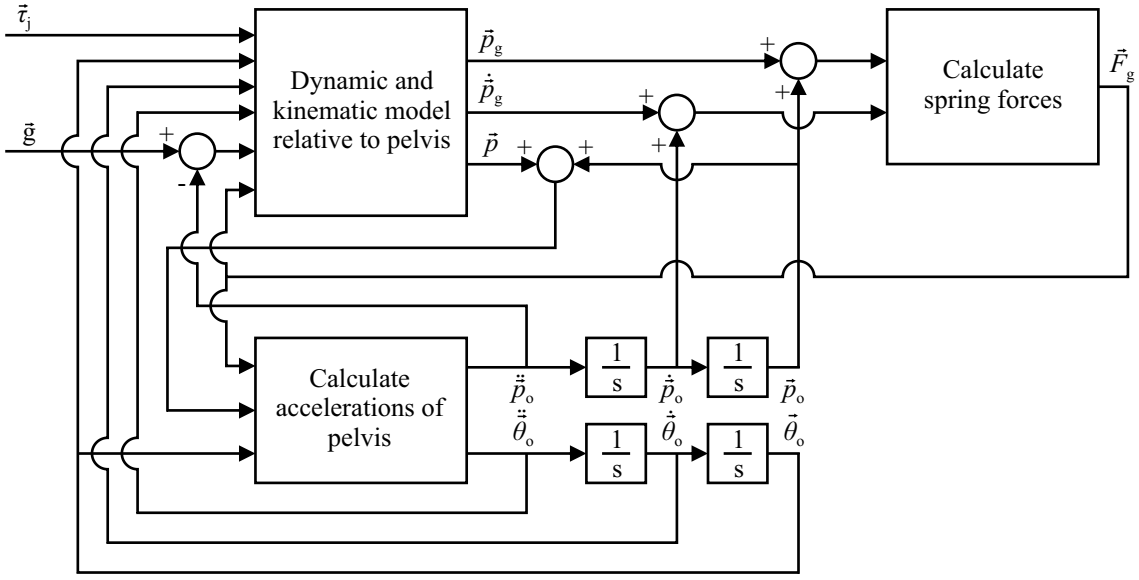


Figure 5.7: *The set-up used to model the robot. Note that it goes for all the outputs of the dynamic and kinematic model block that there are multiple point vectors and to all of these the summations add the same vector, e.g.* $\forall i, \vec{p}_{\mathrm{g},i} + \vec{p}_{\mathrm{o}}$.

The dynamic model set up in Section 5.3 on page 42 has the pelvis as origin, which means that using the model without change the pelvis is stationary and all other parts of the robot moves relative to it. To obtain a free body model and get the actual absolute positions, velocities, and accelerations of all model points the movement of the pelvis must also be taken into consideration. To do this, the accelerations and positions calculated by the relative model are corrected by the addition of the acceleration and position of the pelvis. Furthermore, to set up an expression for the potential energy of the different links of the robot, as used in Section 5.3 on page 42, it is also necessary to include the accelerations of the pelvis. Considering some link of the robot it is not only affected by the gravitational force as originally set up in the dynamic model. The potential energy of a link is instead affected by the gravitational acceleration minus the acceleration of the pelvis in all directions, which changes the equation for the potential energy of the system into Eq. (5.26).

The acceleration of the pelvis is calculated separately as:

$$\ddot{\vec{p}}_{\mathrm{o}} = \frac{\sum_i \vec{F}_{\mathrm{g},i}}{\sum_j m_j} + \vec{\mathrm{g}} \tag{5.22}$$

where:

$\vec{F}_{\mathrm{g},i}$ is the ith ground reaction force [N]
$\vec{\mathrm{g}}$ is the gravitational vector, defined as $[0\ 0\ -9.82]^{\mathrm{T}}$ [m/s^2]
m_j is the mass of the jth link [kg]
$\ddot{\vec{p}}_{\mathrm{o}}$ is the linear acceleration of the point of origin [m/s^2]

Additionally, extra joints for x-, y-, and z-rotation of the pelvis are added to the dynamic model to enable the pelvis to rotate around all axes. The addition of these axes in the dynamic model ensures that the gravitational force acts in the correct direction.

The angular acceleration of the pelvis is modeled by calculating the torque around all three axes at the CoM of the entire robot and then using the moment of inertia tensor of the entire robot to convert the torques into accelerations. These calculations are given by the equations below.

$$\ddot{\vec{\theta}}_{\mathrm{o}} = \boldsymbol{I}_{\mathrm{t}}^{-1} \vec{\tau}_{\mathrm{o}} \tag{5.23}$$

$$\boldsymbol{I}_{\mathrm{t}} = \sum_i \left({}^{0}_{i-1}\boldsymbol{R}\, \boldsymbol{I}_i + m_i \left((\vec{r}_i^{\mathrm{T}} \vec{r}_i)\, \mathbf{I} - \vec{r}_i \vec{r}_i^{\mathrm{T}} \right) \right) \tag{5.24}$$

$$\vec{\tau}_{\mathrm{o}} = \sum_i \vec{r}_{\mathrm{g},i} \times \vec{F}_{\mathrm{g},i} \tag{5.25}$$

where:

$\boldsymbol{I}_i$ is the inertia tensor of the ith link [kgm^2]
$\boldsymbol{I}_{\mathrm{t}}$ is the inertia tensor of the entire robot at the CoM [kgm^2]
$\vec{\tau}_{\mathrm{o}}$ is the resulting torque around the CoM of the entire robot [Nm]
$\vec{r}_{\mathrm{g},i}$ is a vector originating from the global CoM pointing at the point of application of the ith force [m]
$\vec{r}_i$ is a vector originating from the global CoM pointing at the CoM of link i [m]
${}^{0}_{i-1}\boldsymbol{R}$ is the rotation matrix from joint $i-1$ into the global frame
$\ddot{\vec{\theta}}_{\mathrm{o}}$ is the angular acceleration of the point of origin [rad/s^2]

The ground reaction forces are modeled by sets of springs in the directions of all three main axes. The vertical ground reaction forces are modeled by springs in each corner of the foot, i.e. 2 springs in each toe and 2 in each heel. The horizontal forces are likewise modeled by a spring in the x-direction and on in the y-direction placed in each corner of the foot. The distribution of springs and dampers is illustrated in Figure 5.8.

To include the ground reaction forces into the model, the generalized coordinate vector defined in Eq. (5.1) on page 37 is extended to include the coordinates of the spring attachment points. Then Eq. (5.12) on page 43, describing the potential energy in the system, is modified to include the forces and the accelerations of the origin, so it becomes as shown below:

$$E_{\mathrm{pot}} = \sum_{i=1}^{n} \left(m_i (\vec{\mathrm{g}} - \ddot{\vec{p}}_{\mathrm{o}})^T \vec{p}_i \right) + k_{\mathrm{t}} (\theta_1 + \theta_{14}) - E_{\mathrm{grf}} \tag{5.26}$$

$$E_{\mathrm{grf}} = \sum_j \vec{F}_{\mathrm{g},j}\, z_{\mathrm{g},j} \tag{5.27}$$

where:

$z_{\mathrm{g},j}$ is the z-coordinate of the jth spring attachment point [m]
E_{grf} is the potential energy contribution from the ground reaction forces [J]

Note that the value of $z_{\mathrm{g},j}$ is not of any importance in this formula, but the symbol is needed in order to get the correct relations in the formulas based upon the potential energy. After these modifications, the formulas presented in Subsection 5.3.1 on page 43 can be applied directly.

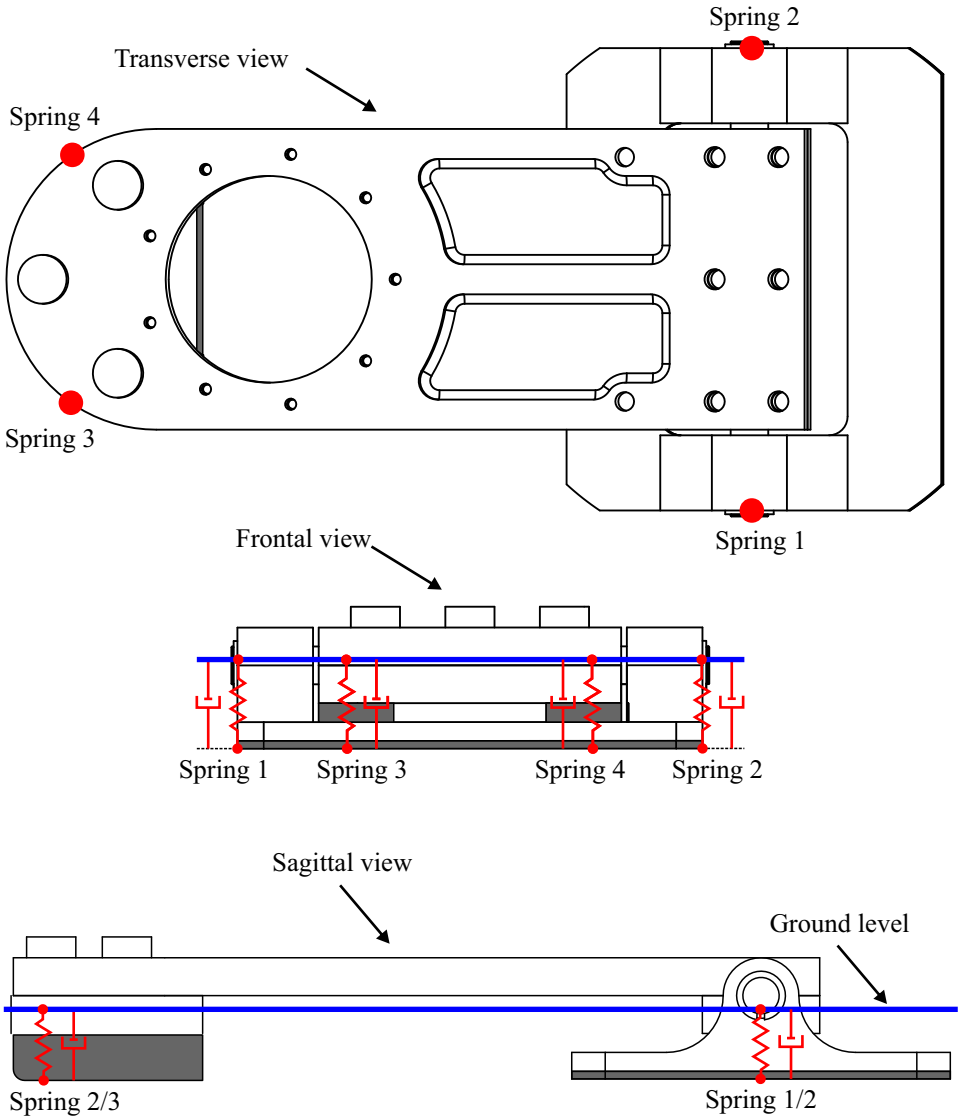


Figure 5.8: *Location of springs used to model the vertical part of the ground reaction forces. The ground level, illustrated by the blue line in the bottom part of the figure, is raised significantly compared to the actual level for illustrative purposes. The red dots indicate the points of attachment of both the springs and dampers.*

5.3.4 Summary of Dynamic Models

In the project two different versions of the dynamic model are utilized. One model is used in the simulation of **AAU-BOT1**, while the second is used in the control system.

Simulation Model

The simulation model is implemented using all 19 joints and the springs described in the previous subsection to model the ground reaction forces. This model has its origin in the center of the pelvis, where three additional virtual joints have been added to make it possible for the origin to rotate in all directions. Also, the origin is able to move because of the equation given in Eq. (5.22).

In this model the utilized mass matrix is calculated based on the individual mass matrixes determined for SSP-L and SSP-R. These are weighted based on the current posture of the robot as shown in Eq. (5.28). The weight factor is based on the forces acting on the feet as described in Eq. (5.29).

$$\boldsymbol{M}_{\text{sim}} = w_{\text{sim}}\boldsymbol{M}_{\text{r}} + (1 - w_{\text{sim}})\boldsymbol{M}_{\text{l}} \tag{5.28}$$

$$w_{\text{sim}} = \frac{F_{\text{z,r}}}{F_{\text{z,r}} + F_{\text{z,l}}} \tag{5.29}$$

where:

$F_{\text{z,l}}$ and $F_{\text{z,r}}$	are the sum of all forces in the z-direction acting on the left and right foot, respectively [N]
$\boldsymbol{M}_{\text{l}}$ and $\boldsymbol{M}_{\text{r}}$	are the mass matrices calculated from the left and right toe, respectively
$\boldsymbol{M}_{\text{sim}}$	is the mass matrix utilized in the dynamic simulation model
w_{sim}	is the weight factor between the two feet [·]

Simplified Model for Controller

For the controller a simplified version of the model is used. This model anticipates that either one or both feet are standing flat and stationary on the ground at all times. Furthermore, as one or both feet are unable to move as seen from this model the spring forces are not included. When the feet are always flat when in contact with the ground, there is no need to include the toe joints. Furthermore, the toe joints cannot be actuated. Hence these are removed from the model, leaving only the 17 joints that are actuated in the model.

Similar to the simulation model a weighting between the two single support models is used to obtain a double support model. However, the weight factor is different from the simulation model as another approach has shown to be better suited in the control model. The weight factor is further discussed in Subsection 6.3.2 on page 67.

5.4 Gear Model

In this section the gears of the robot are modeled. The gears convert the rotational speed of the motor into a slower rotational speed in the joints. The inputs and outputs from the gear model are illustrated in Figure 5.9.

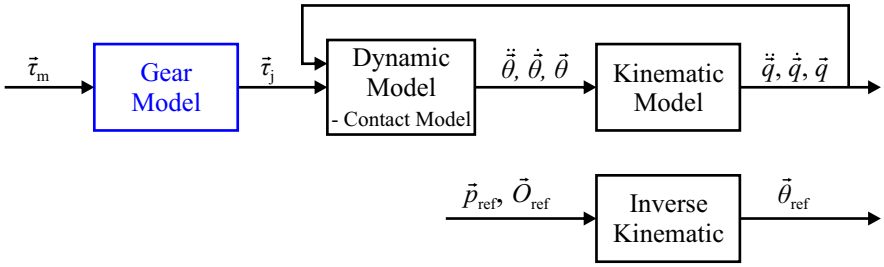


Figure 5.9: *The gear model takes as input: the torques applied by the DC motors. The output is: the resulting torques in each joint.*

The gears are simplified by dividing them into a high-speed shaft, a low-speed shaft, gearing, and a torsion spring. The simplified model is shown in Figure 5.10. Each shaft is modeled as a moment of inertia and a friction coefficient. The shafts are linked together by a gear ratio and gear torsion and slack is modeled by a torsion spring, which includes the flexibility of the gears.

At first glance the flexibility of the gears might seem negligible, but a further inspection reveals that it is a really important aspect. The reason being that the measurement of the angle in the joints are performed on the motor side of the gears, meaning that a small gear slack is measured as a change in the actual joint angle even though the joint is not even moved, if the gear slack is not taken into consideration. If the angle in a joint had been measured on the low-speed side of the gear, where the actual joint is, the flexibility of the gears would not have been as significant as the real joint angle could be used in the control loop. The problem and the solution to it are further discussed in Chapter 8 on page 87.

To simplify, the model is set up for a single gear, while the exact same procedure is actually used for all of the joints individually.

It is chosen to model the gears using a two-mass model, since this catches the behavior of torsion in the gears sufficiently well, as experienced in an earlier project [Esbensen et al., 2008, p. 30]. The two-mass model results in a single torsional mode.

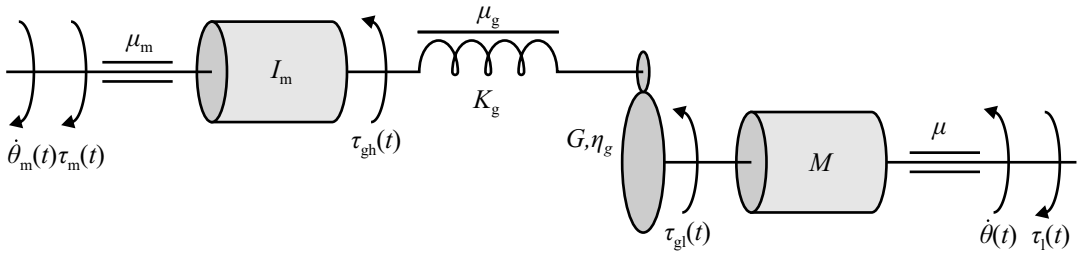


Figure 5.10: *A single gear modeled as a two-mass model.*

The moment of inertia of the high-speed motor shaft, I_m, is set equal to the inertia of the motor plus the inertia of the gears. The inertia of the motor is extracted from the datasheet [maxon motor, 2008b], and the inertia of the Harmonic Drive gears are also found from a datasheet [Harmonic Drive AG, 2005]. Furthermore the inertia of the belts and pulleys are included, and the values for these are found in [Pedersen et al., 2007, App. N, p. 59], however it should be noted that some of these numbers might be incorrect as other parameters in the table have been found to be erroneous. The total inertia of the motor shaft can be calculated using the following equation, utilizing the parameters in Table 5.5.

$$I_m = I_{motor} + I_{belt} + \frac{I_{CPU}}{G_{belt}^2} \tag{5.30}$$

where:

I_{belt} is the moment of inertia of the belt gears $\left[\text{kgm}^2\right]$
I_{CPU} is the moment of inertia of the Harmonic Drive gears $\left[\text{kgm}^2\right]$
I_m is the total moment of inertia of the high-speed motor shaft $\left[\text{kgm}^2\right]$
I_{motor} is the moment of inertia of the motor $\left[\text{kgm}^2\right]$
G_{belt} is the gear ratio of the belt gears, provided in Table 3.1 on page 22 $[\cdot]$

Joint	**Motor**	**Belt gear**	**Harmonic drive gear**	**Total**
	I_{motor}	I_{belt}	I_{CPU}	I_m
Shoulder pitch	34.5e-7 kgm^2	5.1e-7 kgm^2	25e-7 kgm^2	55.6e-7 kgm^2
Waist roll	64.7e-7 kgm^2	291.1e-7 kgm^2	25e-7 kgm^2	358.6e-7 kgm^2
Waist pitch	138e-7 kgm^2	317.1e-7 kgm^2	59e-7 kgm^2	461.7e-7 kgm^2
Waist yaw	34.5e-7 kgm^2	108e-7 kgm^2	59e-7 kgm^2	149.6e-7 kgm^2
Hip yaw	34.5e-7 kgm^2	51e-7 kgm^2	59e-7 kgm^2	98.8e-7 kgm^2
Hip roll	$2 \cdot$ 138e-7 kgm^2	557.6e-7 kgm^2	137e-7 kgm^2	859.5e-7 kgm^2
Hip pitch	138e-7 kgm^2	239.7e-7 kgm^2	137e-7 kgm^2	464.8e-7 kgm^2
Knee pitch	$2 \cdot$ 138e-7 kgm^2	211.1e-7 kgm^2	59e-7 kgm^2	520.3e-7 kgm^2
Ankle pitch	$2 \cdot$ 138e-7 kgm^2	42.1e-7 kgm^2	137e-7 kgm^2	354.3e-7 kgm^2
Ankle roll	64.7e-7 kgm^2	78.9e-7 kgm^2	25e-7 kgm^2	149.8e-7 kgm^2

Table 5.5: *The moment of inertia contributions of the motor shaft.*

The friction coefficient includes the friction in the motor bearings and the first part of the gear.

The dynamics of the high-speed shaft is:

$$I_{\mathrm{m}}\ddot{\theta}_{\mathrm{m}} = \tau_{\mathrm{m}} - \tau_{\mathrm{gh}} - \mu_{\mathrm{m}}\dot{\theta}_{\mathrm{m}} \tag{5.31}$$

where:

μ_{m} is the viscous friction of the high-speed motor shaft [Nm/(rad/s)]
τ_{gh} is the torque acting on the high-speed shaft of the gear [Nm]
τ_{m} is the torque delivered by the motor [Nm]
$\ddot{\theta}_{\mathrm{m}}$ is the angular acceleration of the motor and high-speed shaft [rad/s^2]

The moment of inertia of the low-speed shaft is actually more than just a single number. This is a matrix which includes the cross couplings between the joints, hence it cannot be specified for a single joint individually. To ease the understanding of this section it will however be represented as a single number as the gear model is set up for an individual joint. This means that in this section the symbol M is a simplification of the actual mass matrix $\boldsymbol{M}$ presented in Eq. (5.17) on page 45. The dynamics of the low-speed shaft is:

$$M\ddot{\theta} = \tau_{\mathrm{gl}} - \tau_{\mathrm{l}} - \mu\dot{\theta} \tag{5.32}$$

where:

M is a simple representation of the moment of inertia of the joint $\left[\mathrm{kgm}^2\right]$
μ is the viscous friction of the low-speed shaft [Nm/(rad/s)]
τ_{l} is the loading torque from the gravitational effects on the robot [Nm]
τ_{gl} is the torque acting on the low-speed shaft of the gear [Nm]
$\ddot{\theta}$ is the angular acceleration of the low-speed shaft [rad/s^2]

The coefficient η_{g} in Eq. (5.33) is included to count in the mechanical loss in the gear.

$$\tau_{\mathrm{gl}} = \eta_{\mathrm{g}} G \tau_{\mathrm{gh}} \tag{5.33}$$

where:

G is the joint gear ratio [·]
η_{g} is the efficiency of the gear [·]

A torsion spring and a damping coefficient model the gear slack and torsion of the gear, which is described according to:

$$\tau_{\mathrm{gh}} = K_{\mathrm{g}}\theta_{\Delta} + \mu_{\mathrm{g}}\dot{\theta}_{\Delta} \tag{5.34}$$
$$\theta_{\Delta} = \theta_{\mathrm{m}} - G\theta \tag{5.35}$$

where:

K_{g} is the stiffness/torsion coefficient of the gear [Nm/rad]
μ_{g} is the gear slack damp coefficient [Nm/(rad/s)]
θ_{Δ} is the torsion angle of the gear [rad]

In the following a state space model of the drive train is pursued. The states of the sub-model are $\dot{\theta}_{\mathrm{m}}$, $\dot{\theta}$, and θ_{Δ}. First, Eq. (5.35) is substituted into Eq. (5.34) to obtain:

$$\tau_{\mathrm{gh}} = K_{\mathrm{g}}\theta_{\Delta} + \mu_{\mathrm{g}}\left(\dot{\theta}_{\mathrm{m}} - G\dot{\theta}\right) \tag{5.36}$$

Substituting Eq. (5.36) into Eq. (5.31) results in Eq. (5.37). A similar approach is used to derive Eq. (5.38); however, in this case Eq. (5.36) first has to be substituted into Eq. (5.33) before inserting in Eq. (5.32). Lastly, Eq. (5.35) is differentiated to obtain Eq. (5.39), which is described in terms of the states of the sub-model. Hence, the three first order differential equations to describe the gear sub-model are derived.

$$I_{\mathrm{m}}\ddot{\theta}_{\mathrm{m}} = \tau_{\mathrm{m}} - K_{\mathrm{g}}\theta_{\Delta} - (\mu_{\mathrm{g}} + \mu_{\mathrm{m}})\dot{\theta}_{\mathrm{m}} + \mu_{\mathrm{g}} G\dot{\theta} \tag{5.37}$$
$$M\ddot{\theta} = \eta_{\mathrm{g}} G K_{\mathrm{g}}\theta_{\Delta} + \eta_{\mathrm{g}} G\mu_{\mathrm{g}}\dot{\theta}_{\mathrm{m}} - \left(\eta_{\mathrm{g}} G^2\mu_{\mathrm{g}} + \mu\right)\dot{\theta} - \tau_{\mathrm{l}} = \tau_{\mathrm{j}} - \mu\dot{\theta} - \tau_{\mathrm{l}} \tag{5.38}$$
$$\dot{\theta}_{\Delta} = \dot{\theta}_{\mathrm{m}} - G\dot{\theta} \tag{5.39}$$

The parameters of the gear sub-model are determined from experiments fitting the model to actual measurement data. The friction of the high- and low-speed shafts are determined together in Appendix C on page 124, and is then distributed equally among the two, while taking the gear ratio and the efficiency of the gear into consideration. All of the parameters for the gear model are shown in Table 5.6.

K_g	1 Nm/rad
η_g	0.82
μ_g	12 mNm/(rad/s)

Table 5.6: *Parameters for the gear model. The efficiency is set to the approximate maximum efficiency at* 25 °C *found in [Harmonic Drive AG, 2005]. The remaining two parameters are determined experimentally.*

Three first order differential equations have been derived in this section in order to describe the behavior of the gear sub-model.

5.5 Inverse Kinematic Model

The kinematic model set up in Section 5.2 on page 37 is used to determine the position of all the links in some global Cartesian space based on all the joint angles. The purpose of the inverse kinematics on the other hand is to determine the joint angles based on requirements to the position and orientation of some of the joints. The inputs and outputs from the inverse kinematic model are illustrated in Figure 5.11.

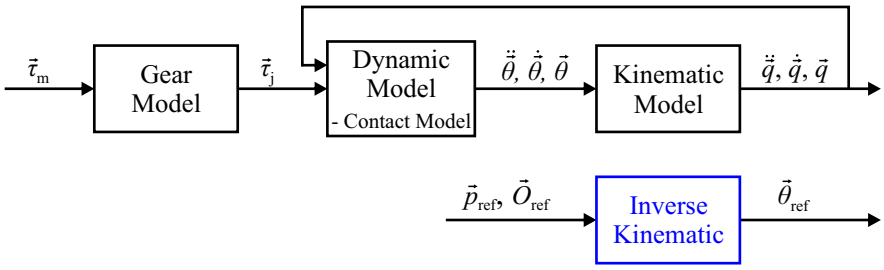


Figure 5.11: *The inverse kinematic model takes as input: the references to the positions and orientations. The output is: the references to the joint angles.*

The inputs to the inverse kinematics are further separated into groups, illustrated in Figure 5.12. Depending on if the robot is standing on the right foot or the left foot it is different points on the robot that has to be positioned. When standing on the right foot it is the position and orientation of the right hip, $\vec{p}_{rh}$ and o_{rh}, and the left foot, $\vec{p}_{lf}$ and o_{lf}, that should meet the requirements, and opposite when standing on the left foot. In both situations the orientations of the waist and arms, o_w, o_{ra}, and o_{la}, should meet the requirements.

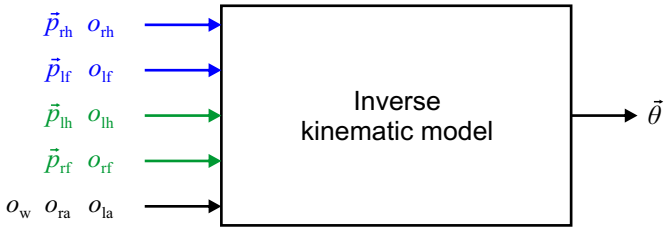


Figure 5.12: *The inputs and outputs of the inverse kinematic model. When in single support on the right foot it is the inputs marked with blue that are used, and the inputs marked with green are used when the robot is standing om the left foot.*

The inputs to the inverse kinematic are marked on an illustration of **AAU-BOT1** in Figure 5.13.

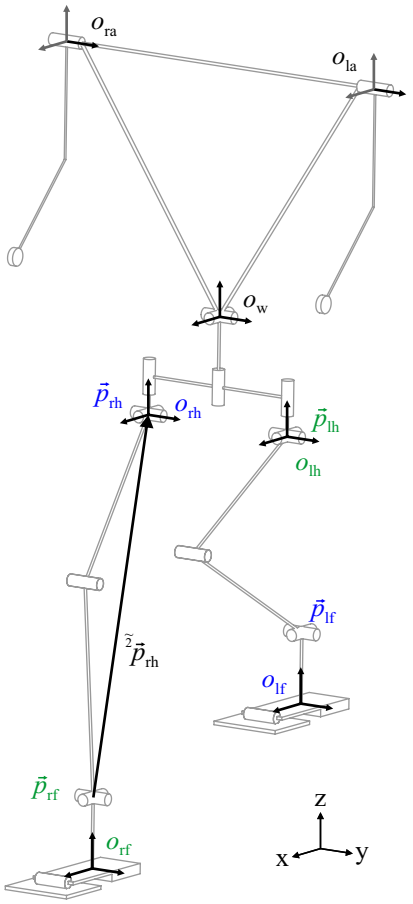


Figure 5.13: *The points and orientations used in the inverse kinematics.*

The inverse kinematic problem can be solved with two methods called the numerical solution and the closed form solution. The numerical solution is an iterative search for the joint angles that gives the smallest error between the required position and the achieved position. The closed form solution is based on analytic expressions to determine the exact angles and avoiding the iterative process [Craig, 2005, p. 106]. The numerical solution has been tested on a biped robot by [Christensen et al., 2007a], and it showed that the method is able to determine the joint angles, but is a heavy computational task, as it took 2.9 s on average to determine the angles in the leg.

The closed form solution is much faster to compute since the joint angles can be calculated directly using analytic expressions. But a closed form solution only exist when a robot manipulator has 5 DoF or less, or in some special cases of robot manipulators with 6 DoF. Therefore, robot manipulators are often designed in such a way that a closed form solution to the inverse kinematic exist. With some considerations and assumptions this method can be applied to **AAU-BOT1**.

To achieve a closed form solution for a manipulator with 6 DoF, three neighboring joints axes must intersect at a point [Craig, 2005, p. 106]. Considering the legs of **AAU-BOT1**, they consist of 7 rotatable joints where the last three joints in the hip are intersecting, if the z-component of 3 mm in the link vector a_7 is disregarded. The link vector is shown in Table 5.1 on page 39. The first joint in the leg is the toe joint which is unactuated, and is affected by the movement of the leg. So by considering the leg from the ankle to the hip it has 6 DoF where the last three joints intersects, by which a closed form solution can be derived for the standing leg to achieve the desired position and orientation of the hip. The orientation of the torso is determined by a z-x-y rotation, and the arms can only be rotated around the y-axis.

The method for a manipulator with 6 DoF where the last three joints intersect is called Pieper's solution [Craig, 2005, p. 114]. This method will be used to solve the inverse kinematic for the right

leg in the following. Note that in the following, the notation c_i and s_i will be used to represent $\cos(\theta_i)$ and $\sin(\theta_i)$.

5.5.1 Inverse Kinematics of the Right Leg

By assuming that the robot is standing on the right foot the inverse kinematics of the right leg can be split into two parts. The first part consists of determining the first three angles to achieve the desired position of the right hip, $\vec{P}_{\text{rh}}$. In the second part the last three angles are determined to get the orientation of the right hip, O_{rh}, which also determine the orientation of the pelvis. Note that through the calculation of the inverse kinematic, the zero components in the link vectors, shown in Table 5.1 on page 39, are taken into consideration to simplify the calculations. The position of the right hip, illustrated in Figure 5.13, can be expressed as shown in Eq. (5.40).

$$ {}^{\tilde{2}}\vec{p}_{\text{rh}} = {}^{\tilde{2}}_{2}\boldsymbol{T}\ {}^{2}_{3}\boldsymbol{T}\ {}^{3}_{4}\boldsymbol{T}\ {}^{4}\vec{p}_{\text{rh}} = \begin{bmatrix} x_{\text{rh}} \\ y_{\text{rh}} \\ z_{\text{rh}} \\ 1 \end{bmatrix} \tag{5.40} $$

where:

- ${}^{\tilde{2}}_{2}\boldsymbol{T}$ is a transformation matrix from frame 2 to the frame located in the zero orientation of frame 2
- ${}^{i-1}_{i}\boldsymbol{T}$ is a transformation matrix from frame i to $i-1$
- ${}^{\tilde{2}}\vec{p}_{\text{rh}}$ is the position of the right hip seen in frame 2 with its zero orientation [m]

The transformation matrix describes the position and orientation of a frame relative to a reference frame, and consists of a rotation matrix and a link vector as shown in Eq. (5.41).

$$ {}^{i-1}_{i}\boldsymbol{T} = \left[\begin{array}{c|c} {}^{i-1}_{i}\boldsymbol{R} & \vec{a}_i \\ \hline 0 & 1 \end{array}\right] \tag{5.41} $$

By using Pieper's solution Eq. (5.40) is rewritten into Eq. (5.43).

$$ {}^{\tilde{2}}\vec{p}_{\text{rh}} = {}^{\tilde{2}}_{2}\boldsymbol{T}\ {}^{2}_{3}\boldsymbol{T}\ \vec{f}(\theta_4) \tag{5.42} $$

$$ \vec{f}(\theta_4) = \begin{bmatrix} f_1(\theta_4) \\ f_2(\theta_4) \\ f_3(\theta_4) \\ 1 \end{bmatrix} = {}^{3}_{4}\boldsymbol{T}\ {}^{4}\vec{p}_{\text{rh}} = \begin{bmatrix} s_4 a_{5\text{z}} \\ a_{5\text{y}} \\ c_4 a_{5\text{z}} + a_{4\text{z}} \\ 1 \end{bmatrix} \tag{5.43} $$

where:

- $\vec{f}(\theta_4)$ is a function mapping the point $\vec{p}_{\text{rh}}$ from frame 4 to frame 3

Since the function $\vec{f}(\theta_4)$ only contains the angle θ_4 it can be used together with the squared magnitude of ${}^{\tilde{2}}\vec{p}_{\text{rh}}$ to determine θ_4. The squared magnitude is defined as shown in Eq. (5.44) and it can be shown that it is equal to Eq. (5.45).

$$ r^2 = x_{\text{rh}}^2 + y_{\text{rh}}^2 + z_{\text{rh}}^2 \tag{5.44} $$

$$ = f_1^2 + f_2^2 + f_3^2 \tag{5.45} $$

where:

- r is the magnitude of ${}^{\tilde{2}}\vec{p}_{\text{rh}}$ [m]

By inserting Eq. (5.43) into Eq. (5.45), θ_4 can be determined as shown in Eq. (5.46).

$$ \theta_4 = \arccos\left(\frac{r^2 - a_{5\text{z}}^2 - a_{4\text{z}}^2 - a_{5\text{y}}^2}{2 a_{5\text{z}} a_{4\text{z}}}\right) \tag{5.46} $$

By writing out the expression for the position of the right hip, shown in Eq. (5.40), the expression shown in Eq. (5.47) is obtained.

$$ {}^{\tilde{2}}\vec{p}_{\text{rh}} = \begin{bmatrix} x_{\text{rh}} \\ y_{\text{rh}} \\ z_{\text{rh}} \\ 1 \end{bmatrix} = \begin{bmatrix} c_3 f_1 + s_3 f_3 \\ c_2 f_2 + s_2 s_3 f_1 - c_3 s_2 f_3 \\ s_2 f_2 - c_2 s_3 f_1 + c_2 c_3 f_3 \\ 1 \end{bmatrix} \tag{5.47} $$

From this it is clear that the x-component only depends on θ_3 and θ_4. Since θ_4 is known, θ_3 can be determined as shown in Eq. (5.48) [Craig, 2005, p. 377].

$$ \theta_3 = \text{atan2}(f_3, f_1) \pm \text{atan2}\left(\sqrt{f_1^2 + f_3^2 - x_{\text{rh}}^2}, x_{\text{rh}}\right) \tag{5.48} $$

Now θ_2 is the only unknown and can be determined from the y-component as shown in Eq. (5.49).

$$ \theta_2 = \text{atan2}\left(s_3 f_1 - c_3 f_3, f_2\right) \pm \text{atan2}\left(\sqrt{f_2^2 + \left(s_3 f_1 - c_3 f_3\right)^2 - y_{\text{rh}}^2}, y_{\text{rh}}\right) \tag{5.49} $$

The desired position of the right hip is now achieved and the last part for the right leg is the orientation of the right hip, which is described in the following.

Right Hip Orientation

The orientation is determined by the last three joints in the leg, which is mounted as an y-x-z rotation. The orientation can be expressed using the rotation matrices as shown in Eq. (5.50).

$$ {}^{1}_{7}\boldsymbol{R} = {}^{1}_{4}\boldsymbol{R}\ {}^{4}_{7}\boldsymbol{R} \tag{5.50} $$

$$ {}^{4}_{7}\boldsymbol{R} = {}^{1}_{4}\boldsymbol{R}^{-1}\ {}^{1}_{7}\boldsymbol{R} \tag{5.51} $$

where:

${}^{1}_{7}\boldsymbol{R}$ represent the desired orientation of the right hip relative to the body frame
${}^{1}_{4}\boldsymbol{R}$ is the rotation of the first three angles in the leg
${}^{4}_{7}\boldsymbol{R}$ is the y-x-z rotation of the last three joints in the leg

By rewriting Eq. (5.50) into Eq. (5.51), the left side of the equation contains the unknown angles θ_5, θ_6, and θ_7, and the right side is known as the rotation of the first three angles and the desired orientation of the right hip are all known. In Eq. (5.52), Eq. (5.51) is written out.

$$ \begin{bmatrix} c_5 c_7 + s_5 s_6 s_7 & -c_5 s_7 + s_5 s_6 c_7 & s_5 c_6 \\ c_6 s_7 & c_6 c_7 & -s_6 \\ -s_5 c_7 + c_5 s_6 s_7 & s_5 s_7 + c_5 s_6 c_7 & c_5 c_6 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{5.52} $$

From this it is clear that θ_6 can be determined as shown in Eq. (5.53).

$$ \theta_6 = -\arcsin(r_{23}) \tag{5.53} $$

Since θ_6 now is known the solution to θ_7 is given as shown in Eq. (5.54).

$$ \begin{aligned} \cos(\theta_7) &= \frac{r_{22}}{c_6} \\ \sin(\theta_7) &= \frac{r_{21}}{c_6} \\ \theta_7 &= \text{atan2}\left(\frac{r_{21}}{c_6}, \frac{r_{22}}{c_6}\right) \end{aligned} \tag{5.54} $$

In the same way θ_5 is found as shown in Eq. (5.55).

$$ \theta_5 = \text{atan2}\left(\frac{r_{13}}{c_6}, \frac{r_{33}}{c_6}\right) \tag{5.55} $$

All the angles in the right leg are now found based on a specified position and orientation of the right hip.

5.5.2 Inverse Kinematics of the Left Leg

The inverse kinematics for the left leg can be calculated in the same way as the right leg. Instead of calculating the inverse kinematics for the kinematic chain from the left hip to the left foot, it is calculated from the foot to the hip. Using the vector ${}^{\tilde{1}2}\vec{p}_9$, illustrated in Figure 5.14, Pieper's solution is used in the same way as for the right leg.

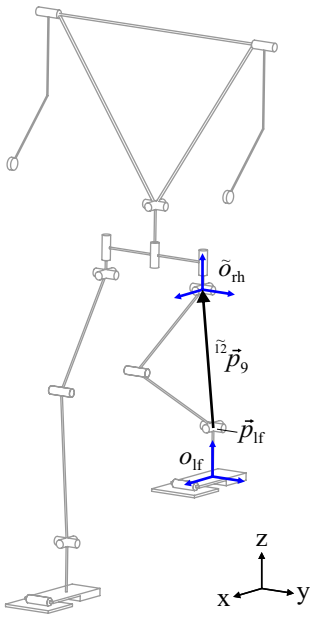


Figure 5.14: *Illustration of how the calculations of the inverse kinematics for the left leg is done.*

Since the joint angles θ_2 to θ_7 are known, the position of joint 9, see Figure 5.3 on page 38, can be determined as shown in Eq. (5.57) by disregarding the z-component of a_9. Afterwards, the point vector used in Pieper's solution is found as shown in Eq. (5.56).

$$ {}^{\tilde{1}2}\vec{p}_{\text{j9}} = o_{\text{lf}} \left(\vec{p}_{\text{j9}} - \vec{p}_{\text{lf}} \right) \tag{5.56} $$

$$ \begin{bmatrix} \vec{p}_{\text{j9}} \\ 1 \end{bmatrix} = {}^{0}_{7}\boldsymbol{T} \begin{bmatrix} \vec{a}_8 + \vec{a}_9 \\ 1 \end{bmatrix} \tag{5.57} $$

where:

- o_{lf} is the desired orientation of the left foot
- $\vec{p}_{\text{j9}}$ is the position of joint 9 in the body frame [m]
- ${}^{\tilde{1}2}\vec{p}_{\text{j9}}$ is the point vector from the left ankle to the left hip [m]
- $\vec{p}_{\text{lf}}$ is the desired position of the left foot [m]

For the left leg the orientation of the foot is set as a requirement, therefore before using Pieper's solution the orientation of the foot is used as the base orientation. To calculate the joint angles in the left hip, the orientation of the left hip is the same as the right hip but including the base orientation as shown in Eq. (5.58).

$$ \tilde{o}_{\text{rh}} = o_{\text{lf}}\, o_{\text{rh}} \tag{5.58} $$

where:

- $\tilde{o}_{\text{rh}}$ is the orientation of the left hip seen from the left foot, to be used in Pieper's solution

Now the inverse kinematic has been set up enabling the determination of the joint angles to achieve a specific position and orientation of the robot.

5.5.3 Verification of Inverse Kinematics

The inverse kinematics is verified by use of the kinematic model and then comparing the input to the inverse kinematics with the output from the kinematic model, as shown in Figure 5.15.

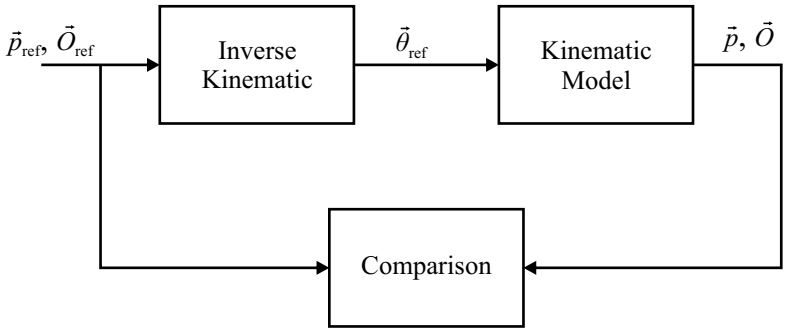


Figure 5.15: *Block diagram illustrating the verification of the inverse kinematics based on the kinematic model.*

The verification showed that the inverse kinematics is able to determine the joint angles that produce the requested positions and orientation. This means that the inverse kinematic in fact is the inverse of the kinematic model, and by assuming that the kinematic model is correct, the inverse kinematic model is also correct.

Having set up a complete model of **AAU-BOT1** including the kinematics, dynamics, and the flexibility of the gears. The model is used both for simulation and for the design of the control system, which is presented in next part starting with the design of the trajectory for **AAU-BOT1**.

III

Control System Design

6 Controller Design

In this chapter the design of a control system for the robot is described. The control system consist of a posture controller, a balance controller, a phase estimator, a ZMP estimator, and the Kalman estimator designed in Chapter 8 on page 87, along with the trajectory generation in Chapter 7 on page 76. The implementation of the control system together with the model can be seen in the `Matlab Simulink` model located on the DVD in [`Model_with_Control_System/`].

Contents

6.1 Control Strategy

So far a complete model of the robot has been set up, to be able to simulate the behavior of the robot both in single support and double support. The purpose of the control system is to track a trajectory that ensures static balanced gait. The trajectory includes basically two parts, one part is the trajectory for each joint that ensures that the robot is walking forward, the second is a trajectory of the ZMP that ensures that the robot is in balance during the walk.

The overall control system consists of two control goals. The first goal is to control the posture of the robot in such a way that the robot follows the trajectory. The other control goal is to ensure that the robot always is in balance. The posture control can be designed in different ways, one is to use the absolute position, in Cartesian coordinates, of the robot and then control the joint angles to obtain the desired absolute position. This has the advantages that it is possible to ensure that the robot always is in the correct position and thereby making sure that the robot is in balance. To use the absolute position though require a form of online trajectory generation and special measurement equipment as described in Section 7.1 on page 76. Another strategy for the posture control is to use the angle trajectory for each joint which means that there is no control on the absolute position of the robot. This approach is more strait forward and does not require special measurement equipment, which makes it suitable to implement on **AAU-BOT1**. Using only the angle trajectories require an additional outer control loop to ensure balance since disturbances and model uncertainty can entail that the balance can be compromised. Therefore, the control strategy illustrated in Figure 6.1 will be used where the two control loops control the posture and the balance of the robot, respectively.

The posture control should be designed to track the joint angle trajectories in a way that ensures the correct position of the robot during the gait cycle and also provides a smooth gait such that the balance is not compromised. The choice of reference signal for the posture controller should be taken based on achieving these goals. To track the position based on the joint angle trajectories the most obvious choice of reference signal would be the joint angles, $\vec{\theta}$, since the trajectory gives the angular positions during the gait cycle. But to get a smooth gait it could be beneficial to use the angular velocities, $\dot{\vec{\theta}}$. When the angle position is used as reference it could entail fluctuations on the angular velocity, which will be reduced with use of the angular velocity as reference, hence a smoother gait is

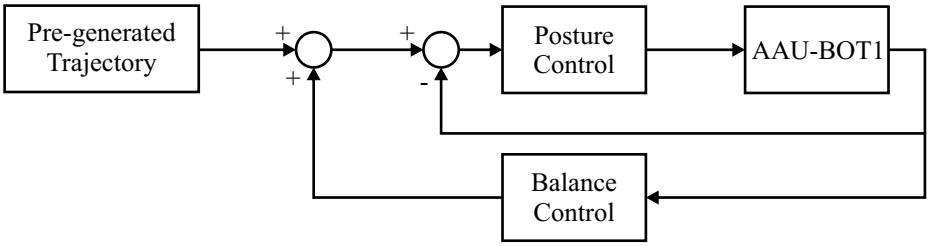


Figure 6.1: *The overall structure of the general control system.*

obtained. However, the consequence can be that a large error on the position is build up over time. This can happen if the controller is unable to track the angular velocity completely for a period of time; then, the error will also entail an error on the position, but even if the controller later obtains the correct angular velocity, there will be a constant error on the position. The consequence is illustrated in Figure 6.2 where the velocity is used as reference and, e.g. caused by friction, the velocity gets an error in the beginning but reaches the reference afterwards. The error then entails a constant error on the position as illustrated.

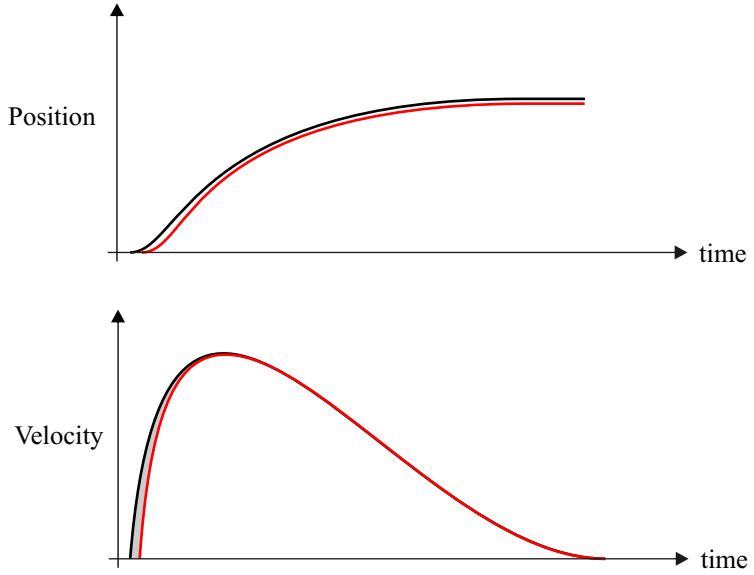


Figure 6.2: *Illustration of the consequence by using the velocity as reference. The black graph illustrate the reference and red illustrate the obtained.*

Since it is rather important that the robot track the position specified by the trajectory during a gait cycle, the angular position is used as reference signal to the posture controller. Even though it is not pursued in this thesis it may be beneficial for a later project group to investigate the possibility of using both position and velocity references.

Having described the control strategy, the following section presents the structure of the control system and describes the different parts that constitute the control system.

6.2 Control Structure

The structure of the control system and the connection between the different parts are shown in Figure 6.3. Each part of the control system is shortly described in the following.

- **Trajectory**
  An important step in making a biped robot walk is to design a gait pattern which the robot should follow. This gait pattern is designed as a position trajectory for different parts of the

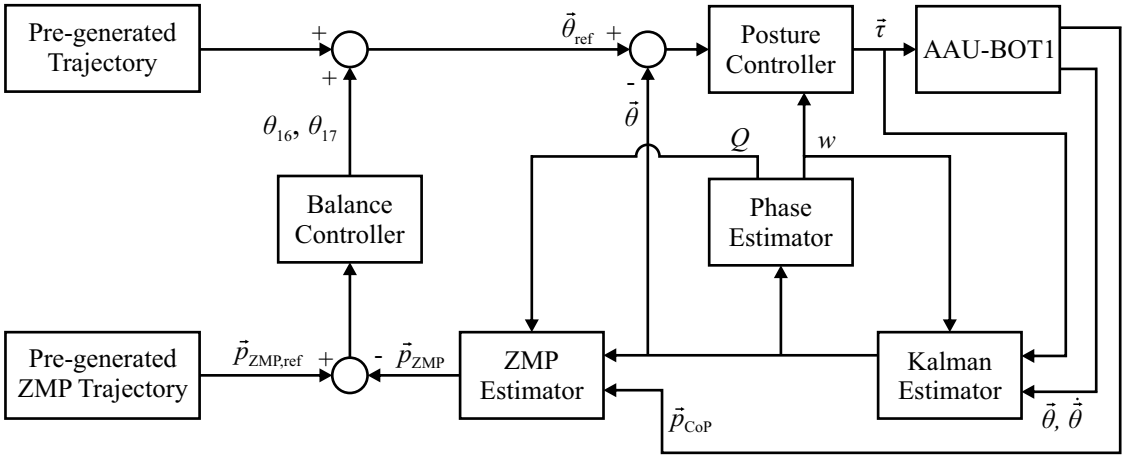


Figure 6.3: *Block diagram illustrating the control structure.*

robot, which is transformed into joint angle trajectories for all joints using the inverse kinematic model. The trajectory is designed to include the physical limitations of the robot and to ensure stability through the gait cycle. The posture controller uses the joint angle trajectories to ensure that the robot achieve the desired gait pattern, while the balance controller uses the resultant ZMP trajectory to keep the balance. The design of the trajectory is further described in Chapter 7 on page 76.

- **Posture Controller**
  The posture controller is the main part of the control system with the purpose of controlling the angular position of each joint such that a static gait is achieved. This is done by using the joint angles as reference signals and the motor currents as control signals which are proportional to the motor torques. The posture controller has been designed using two different strategies, where one strategy was to design a simple control system with the purpose of studying the possibility of making the robot walk with simple control. The weaknesses and disadvantages of the simple control strategy necessitate a model based control strategy which has shown to be more suitable.

- **Kalman Estimator**
  The measurements available from the robot are the angles and angular velocities of the motor and, as described in Section 5.4 on page 50, the flexibility of the belts and gears entail that the use of the measurements without any handling can cause unwanted vibrations. To get the optimal control performance the angles and angular velocities of the joints should be used instead of those of the motor, but since these measurements are not available a Kalman estimator is designed to estimate the unmeasured states. The Kalman estimator takes the measured motor angles and angular velocities, along with the control signal to estimate the joint angles and joint angular velocities. The Kalman estimator is described and designed in Chapter 8 on page 87.

- **Balance Controller**
  To get a balanced gait the robot must keep the ZMP specified by the trajectory. Ideally, the balanced gait is achieved only by tracking the joint angle trajectory since the generated trajectory is designed to keep the robot in balance at all times. But due to disturbances, model uncertainty, and the fact that the trajectory is not tracked perfectly, the balance controller should be designed to track the trajectory of the ZMP. The balance is controlled by use of the torso since a significant part of the total mass is located in the torso. The torso joints θ_{16} and θ_{17} are used to move the ZMP in the y- and x-direction, respectively. The changes to the two angles are added as compensations to the joint angle trajectories, as illustrated in Figure 6.3.

- **ZMP Estimator**
  The input to the balance controller is the CoP measured by the FTS which are identical to

the calculated ZMP during balance gait. But since the FTS in the left ankle has shown to be defective, only the CoP from the right ankle is available. This means that when the robot is standing on the left leg it is not possible to measure the balance directly; instead, the model is used to determine the balance. But the calculation of the ZMP includes the angular accelerations, which are not available and the ZMP can therefore not be calculated based on the measurements. Instead, the GCoM is used in the balance controller since the ZMP and the GCoM are assumed to be coincident during static gait. The ZMP estimator consists of a switching mechanism that provides the measured CoP when the robot is in single support right and double support and when the robot is in single support left the model is used to calculate the GCoM.

- **Phase Estimator**
  During walk it is important to know which phase the robot is in, i.e. single support right, single support left, or double support, since the calculation of the GCoM in the ZMP estimator is done by the kinematic model and thereby is depending on the phase. Also the model used by both the posture controller and the Kalman filter is depending on which phase the robot is in. The phase estimator takes the estimated states and determines which phase the robot is in.

Having presented the structure of the control system the different parts will be designed in the following sections, except for the Kalman estimator which is designed in Chapter 8 on page 87.

6.3 Posture Control

To make the robot walk, the posture control should control the angular position of each joint, which can be achieved in many ways. In this project, two different control strategies are studied, one being a simple strategy with independent controller for each joint and the other a model based approach.

To study the possibility of making a robot walk with a simple control strategy an approach is taken with simple independent controllers for each joint. This has the advantage that it is more strait forward to implement on a real robot. The structure of the simple control strategy is illustrated in Figure 6.4 and the design of the controller is described in details in Appendix E on page 132. Different tests have shown the weaknesses of the simple control which primarily is the cross couplings in the system and the highly non-linear load in the joints. The movements of each joint affect the others and thereby the individual controller can result in instability. This is particularly an issue for the legs where the three pitch joints are placed one after another and are affected by a high load. The effect of the cross couplings gets more significant the faster the movements are, due to the fact that the load on each joint is dependent on the angular velocity. Also the high load and its non-linearity can entail that the controllers in some situations are unable to keep the position, which will lead to the robot falling over.

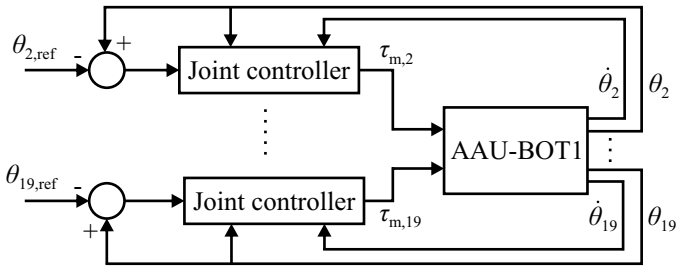


Figure 6.4: *The structure of the joint controllers with an independent controller for each joint.*

The relatively slow static trajectory that is designed in this project signifies that the weaknesses of the simple control strategy are less distinguished, and it will therefore probably be able to control the robot in a slow static walk. This will though require different controllers depending on whether the robot is in single support right, single support left, or double support due to the changes in the dynamic behaviors, and hence a switching mechanism between the controllers will be needed. It is therefore decided to use a control strategy that is based on the dynamic model and is therefore

able to handle the cross couplings and the non-linear load. Using this approach the handling of the different phases of the robot becomes more strait forward. This strategy is based on the control method called computed-torque control and is presented in the following section.

6.3.1 Computed-Torque Control

The dynamics of the robot is influenced by cross couplings between the joints, meaning that the movements of each joint are affected by each other. These cross couplings have shown to make it difficult to design individual controllers for each joint. Therefore a method for decoupling and linearization will be described and designed in this section.

The method is called computed-torque control and uses the model of the robot to compute the necessary torque to achieve a desired motion [Siciliano and Khatib, 2008, p. 143]. The computed-torque control is a well known and tested method to control the motion of biped robots [Caux and Zapata, 1999], [Löffler et al., 2004].

By using the model to compute the motor torque, based on a desirable angular acceleration, as shown in Eq. (6.1), a system that is decoupled and feedback linearized is obtained. As shown in Figure 6.5 the computed torque consists of two loops, the inner non-linear compensation which compensate for the load caused by centrifugal, Coriolis, gravity, and friction terms. The second loop takes the desired angular acceleration acting as an exogenous control signal and decouple the system through the mass matrix, $\boldsymbol{M}(\vec{\theta})$, to compute the needed torque.

$$\vec{\tau}_{\mathrm{m}} = \boldsymbol{M}(\vec{\theta})\vec{v} + \vec{V}(\vec{\theta}, \dot{\vec{\theta}}) + \vec{G}(\vec{\theta}) + \vec{F}(\vec{\theta}, \dot{\vec{\theta}}) \tag{6.1}$$

where:

$\vec{F}(\vec{\theta}, \dot{\vec{\theta}})$	is the friction forces, including viscous friction, Coulomb friction, and stiction [Nm]
$\vec{G}(\vec{\theta})$	is the vector containing all gravity terms
$\boldsymbol{M}(\vec{\theta})$	is the mass matrix of the robot
$\vec{V}(\vec{\theta}, \dot{\vec{\theta}})$	is the velocity vector which contains centrifugal and Coriolis terms
$\vec{v}$	is the control signal [rad/s^2]

When the system is decoupled and linearized, a controller that operates on the desired angular acceleration can be added, as shown in Figure 6.5.

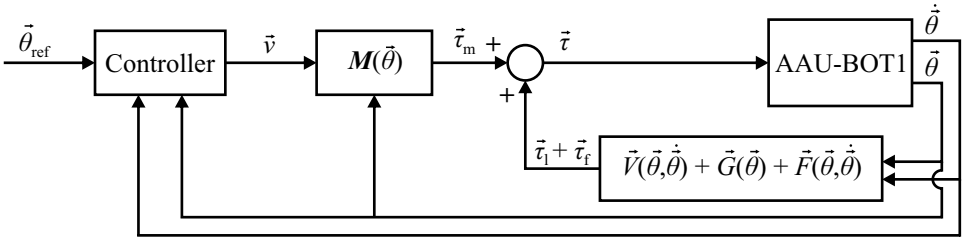


Figure 6.5: *Block diagram showing the structure of the computed-torque control method.*

Controller

The controller designed for the computed-torque is an LQR with the structure shown in Figure 6.6. Besides the standard LQR a feedforward gain is added from the reference signal to reduce the response time. To ensure a steady state error of zero, an integrator and the corresponding integral gain is included in the controller. To include the integral gain in the LQR design, the integrator is

added to the model in the following way:

$$\dot{\vec{x}}_{\mathrm{e}} = \boldsymbol{A}_{\mathrm{e}}\vec{x}_{\mathrm{e}} + \boldsymbol{B}_{\mathrm{e}}\vec{u} \tag{6.2}$$

$$\begin{bmatrix}\dot{\vec{x}}\\ \dot{\vec{x}}_{\mathrm{i}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{A} & \mathbf{0}\\ \boldsymbol{C} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\vec{x}\\ \vec{x}_{\mathrm{i}}\end{bmatrix} + \begin{bmatrix}\boldsymbol{B}\\ \mathbf{0}\end{bmatrix}\vec{u} - \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\vec{x}_{\mathrm{ref}}$$

$$\begin{aligned}\vec{y} &= \boldsymbol{C}_{\mathrm{e}}\vec{x}_{\mathrm{e}} \\ &= \begin{bmatrix}\boldsymbol{C} & \mathbf{0}\end{bmatrix}\vec{x}_{\mathrm{e}}\end{aligned} \tag{6.3}$$

The state feedback gain and the integral gain is calculated using the `MATLAB` function `lqr`, based on the state weight matrix $\boldsymbol{Q}$ and the control signal weight matrix $\boldsymbol{R}$. The weight matrices is set to be diagonal matrices and used as tuning parameters. The calculated gains are used as follows:

$$\begin{aligned}\vec{u} &= \boldsymbol{K}_{\mathrm{e}}\vec{x}_{\mathrm{e}} \\ &= \begin{bmatrix}\boldsymbol{K} & \boldsymbol{K}_{\mathrm{i}}\end{bmatrix}\begin{bmatrix}\vec{x}\\ \vec{x}_{\mathrm{i}}\end{bmatrix}\end{aligned} \tag{6.4}$$

However, the system used to design the LQR becomes rather simple. By assuming a perfect decoupling and feedback linearization of the model the system is reduced to the following:

$$\dot{\vec{x}} = \boldsymbol{A}\vec{x} + \boldsymbol{B}\vec{v} \tag{6.5}$$

$$\begin{bmatrix}\dot{\vec{\theta}}\\ \ddot{\vec{\theta}}\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\vec{\theta}\\ \dot{\vec{\theta}}\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\vec{v}$$

$$\begin{aligned}\vec{y} &= \boldsymbol{C}\vec{x} \\ &= \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\vec{\theta}\\ \dot{\vec{\theta}}\end{bmatrix}\end{aligned} \tag{6.6}$$

As it can be seen the system becomes a double integrator from input to output, which is a rather simple system to control. This model is valid for the entire operational area of the robot, meaning that the actual controller is the same whether the robot is in single support right, single support left, or in double support. This is a great advantage since there is no need for switching between different controllers as required with the individual joint controllers presented in Appendix E on page 132.

The complete structure of the computed-torque control with an LQR is illustrated in Figure 6.6.

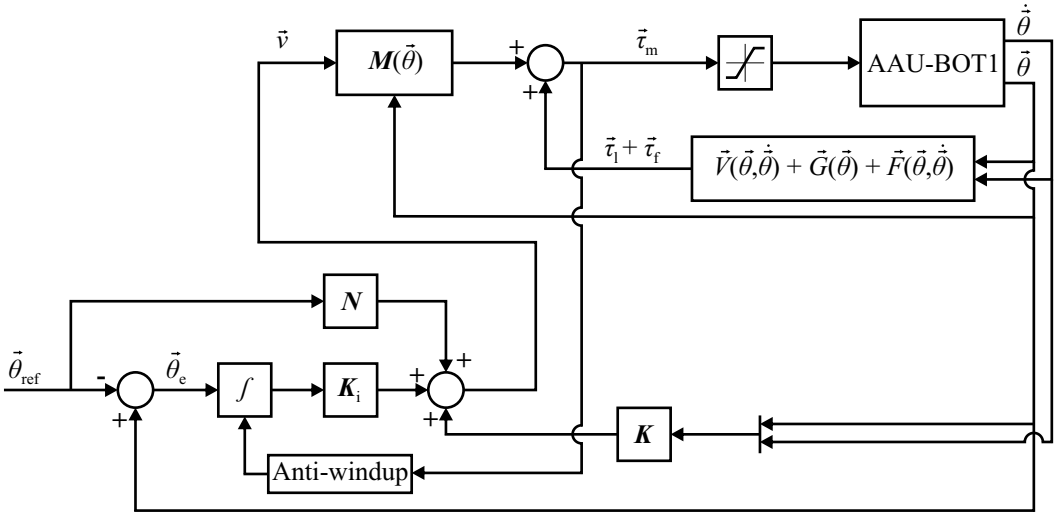


Figure 6.6: *The complete structure of the computed-torque control with an LQR.*

Having set up the computed-torque control method some remarks about the method will be discussed in the following.

Remarks of the Computed-Torque Method

In theory the computed-torque control method is an optimal way to control a robot since all the dynamic effects are included in the control loop, but in practice some issues exist, which could result in the method not being usable. The typical issues are model uncertainty and computational effort.

- **Model uncertainty**
  Since the method uses a model of the robot to compute the needed torque it is clear that an accurate model is needed. If the model differs from the actual robot, the method may induce unstable control due too erroneous decoupling and compensation. The model uncertainty will typically be caused by parameter uncertainty, i.e. the knowledge about the mass, inertia, and kinematics of the robot. In the considered case it is assumed that the parameters are well known since the complete technical drawings that were used to produce the robot are available in `Solidworks`. This means that the kinematics is well known, and the mass and inertia can be calculated from the drawings which should result in accurate parameters.

  The effect of model uncertainty becomes evident when the control signal is inserted into the system model as in Eq. (6.7), resulting in the closed-loop system. Note that the gear model is left out in the shown closed loop system as it is assumed that the angles and angular velocities of the joints are estimated precisely, which means that the effect of the gears become negligible.

$$\begin{bmatrix} \dot{\vec{\theta}} \\ \ddot{\vec{\theta}} \\ \vec{\theta}_{\mathrm{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \boldsymbol{M}^{-1}\hat{\boldsymbol{M}}\boldsymbol{K}_{\theta} & \boldsymbol{M}^{-1}\hat{\boldsymbol{M}}\boldsymbol{K}_{\dot{\theta}} & \boldsymbol{M}^{-1}\hat{\boldsymbol{M}}\boldsymbol{K}_{\mathrm{i}} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \vec{\theta} \\ \dot{\vec{\theta}} \\ \int \vec{\theta}_{\mathrm{e}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{N} \\ \mathbf{-I} \end{bmatrix} \vec{\theta}_{\mathrm{ref}} + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{M}^{-1}\left(\hat{\vec{\tau}}_{\mathrm{l}} + \hat{\vec{\tau}}_{\mathrm{f}} - \vec{\tau}_{\mathrm{l}} - \vec{\tau}_{\mathrm{f}}\right) \\ \mathbf{0} \end{bmatrix} \tag{6.7}$$

  where:

$\hat{\boldsymbol{M}}$	is the mass matrix utilized in the system model, hence in the control system
$\boldsymbol{M}$	is the real mass matrix of the physical system
$\vec{\theta}_{\mathrm{e}}$	is the angular position error [rad]

  The equation clearly shows where a model error can have impact on the system. If the model is close to the correct system, the matrix given by $\boldsymbol{M}^{-1}\hat{\boldsymbol{M}}$ will be close to a diagonal matrix, which means that the system is stable if the contribution for the load and friction compensation are also close to the correct values.

- **Computational effort**
  Another disadvantage by the computed-torque method is that it requires a lot of computational effort. For each sample time the entire dynamic model of the robot must be calculated in order to set the next control input. Therefore, before using this method it is tested that the on-board computer is able to process that amount of data in each sample time. The tests showed that the on-board computer is able to compute the model, which means that the computed-torque controller can be implemented on the computer.

6.3.2 Control in Double Support

The dynamic behavior of the robot changes significantly when the robot switches from single support to double support and vice versa, since the total weight of the robot changes from being held by one foot to being distributed on both feet. This means that the load on each joint in the legs changes significantly. When the dynamics of a system changes, it is also necessary to change the control scheme.

The computed-torque control method uses the dynamic model in a feedback linearization and decoupling which means that the actual controller does not have to change when the robot goes from single support to double support. Due to the feedback linearization and decoupling the system seen

from the controller is unchanged during the walking cycle. This leads to the fact that the feedback linearization and decoupling must be valid in the entire range of operation meaning that the dynamic model used must change when the robot goes from single support to double support.

The dynamic model of the robot is set up based on single support, meaning that two models are set up based on single support right and single support left, respectively, as shown in Eq. (6.8) and Eq. (6.9). In double support the dynamic behavior can be seen as a combination of the two single support models [Christensen et al., 2007a, p. 84].

$$\vec{\tau}_{\mathrm{m,r}} = \boldsymbol{M}_{\mathrm{r}}(\vec{\theta})\ddot{\vec{\theta}}_{\mathrm{c}} + \vec{V}_{\mathrm{r}}(\vec{\theta},\dot{\vec{\theta}}) + \vec{G}_{\mathrm{r}}(\vec{\theta}) + \vec{F}(\vec{\theta},\dot{\vec{\theta}}) \tag{6.8}$$

$$\vec{\tau}_{\mathrm{m,l}} = \boldsymbol{M}_{\mathrm{l}}(\vec{\theta})\ddot{\vec{\theta}}_{\mathrm{c}} + \vec{V}_{\mathrm{l}}(\vec{\theta},\dot{\vec{\theta}}) + \vec{G}_{\mathrm{l}}(\vec{\theta}) + \vec{F}(\vec{\theta},\dot{\vec{\theta}}) \tag{6.9}$$

where:

$\vec{\tau}_{\mathrm{m,r}}$, $\vec{\tau}_{\mathrm{m,l}}$ is the motor torque calculated from the right and left foot model, respectively [Nm]

Then, the two models are weighted in such a way that the more weight that is on the right foot the more the robot is in single support right. The computed torque when in double support then becomes as shown in Eq. (6.10) with the weight factor w that has a value between 0 and 1.

$$\vec{\tau}_{\mathrm{m}} = w\vec{\tau}_{\mathrm{m,r}} + (1-w)\vec{\tau}_{\mathrm{m,l}} \tag{6.10}$$

where:

w is the weight factor between the single support models used in the control [·]

The weight factor is calculated by the phase estimator and is described in Subsection 6.5.2 on page 72.

6.3.3 Friction Compensation

The joints in the robot are affected by high Coulomb friction, thus, to improve the performance of the control system, it would be beneficial to add a friction compensation to the motor current such that the response time of the controller is reduced. The friction compensation is calculated by use of the joint angular velocities, as shown in Eq. (6.11). When the velocity is zero but is about to start, due to a reference change, it is important that the friction compensation has the sign that corresponds to the direction that joint is about to move. Therefore, the error between the joint angle reference and the joint angle is used, in this way the friction is always added in the direction that joint should start to move.

$$\tau_{\mathrm{f}} = \begin{cases} \mathrm{sign}(\dot{\theta})\tau_{\mathrm{c}} + \mu\dot{\theta} & \text{if } \mathrm{abs}(\dot{\theta}) > \epsilon_1 \\ -\mathrm{sign}(\theta_{\mathrm{e}})\tau_{\mathrm{c}} & \text{if } \mathrm{abs}(\dot{\theta}) \leq \epsilon_1 \wedge \mathrm{abs}(\theta_{\mathrm{e}}) > \epsilon_2 \\ 0 & \text{otherwise} \end{cases} \tag{6.11}$$

where:

θ_{e} is the error between the reference angle the actual angle [rad]

ϵ_1 and ϵ_2 are thresholds determined experimentally [·]

Due to noise and fluctuations on the estimated values, the friction compensation is implemented with some thresholds to ensure that the friction compensation is not changing too frequently.

6.4 Balance Controller

In this section the balance controller will be described and tested. The purpose of the balance controller is to track the pre-generated ZMP trajectory and thereby keep the robot in balance. To move the ZMP of the robot a part of the robot must be moved, and the larger the mass that is moved the more the ZMP can be moved. This indicates that there are different ways to control the balance of the robot. First of all the trajectory generated for the gait of the robot is designed to keep the robot in balance at all time, which means that ideally the robot is always in balance only by tracking the generated joint trajectories. In practice there will be small tracking errors

from the trajectory, and the model used for designing the trajectory may differ from the real robot. Additional the robot can be subjected to disturbances which may influence the balance.

Modifying the pre-generated trajectory the balance can be controlled by changing the position of pelvis in the x- and y- direction and thereby moving the entire mass of the robot to achieve balance. This is however not a suitable solution in practice due to the physical limitations in the joint angles, described in Chapter 7 on page 76. Since the trajectory already is close to the limitations it will only be possible to change the position of pelvis in a very small degree. Instead, the balance can be controlled by moving torso.

The weight of the torso is about 13 kg and the balance of the robot is therefore significantly dependent on the position of the torso. The position of torso can be changed by only two joints which mean that the limitations of the joint angles are not exceeded. In Figure 6.7 it is illustrated how the CoM can be moved by use of the torso. Note that the ZMP is also dependent of the velocity and acceleration, but for static gait CoM and ZMP are assumed to be equal.

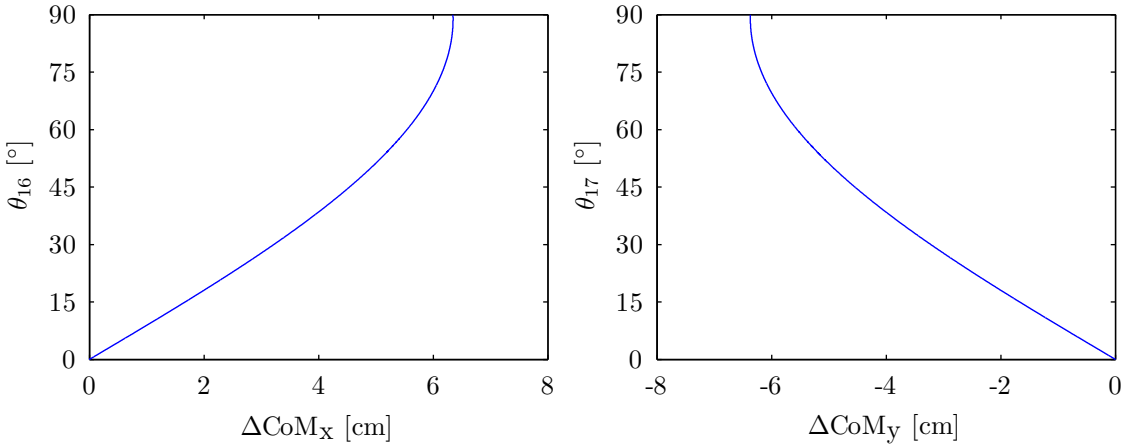


Figure 6.7: *Graph showing how the CoM is displaced by tilting torso using joint angle 16 and 17.*

It is clear that using the torso, the ZMP can be moved significantly and it is therefore possible to use torso to control the balance of the robot. The balance controller is added in an outer control-loop to control the position of the torso using θ_{16} and θ_{17}. The output from the balance controller acts as a compensation, which is added to the joint angle reference from the pre-generated trajectory. The structure of the balance controller is illustrated in Figure 6.8. Since the x- and y-direction of the ZMP depends on θ_{16} and θ_{17}, respectively, the balance controller is designed as two PI-controllers which act on the x- and y-directions independently.

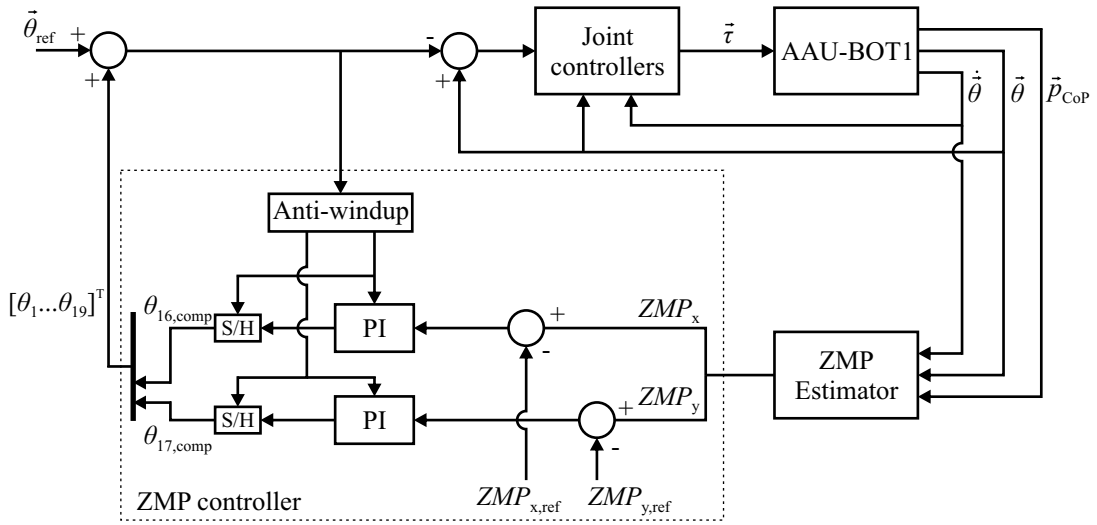


Figure 6.8: *The structure of the balance controller.*

The ZMP is determined by the ZMP estimator which utilizes the measurement from the FTS and the model of the robot, along with the angle and angular velocity measurements. Since the robot only has a functional FTS in the right ankle the ZMP will be measured when the robot is standing on the right foot. When standing on the left foot the ZMP will be determined from the model of the robot and the measurement of the angles and angular velocities.

To limit the angles in torso and avoid very large tilting, the compensation from the balance controller is limited. When the reference to the joint controller becomes larger than some value for θ_{16} and θ_{17}, the output from the balance controller is held constant and anti-windup is added to the integrator.

To illustrate the performance of the balance controller a step is given on the reference while the robot is at a standstill. The result of the test for the x-direction and the y-direction is shown in Figure 6.9 and Figure 6.10, respectively. Note that since the balance controller is in the outer loop the performance is also dependent on the joint controllers.

As the tests show the ZMP gives very high jumps in the beginning which occurs when the torso joint begins to move. This is caused by a step in acceleration that occur when a joint changes from a standstill and begins to move, since the ZMP is dependent on the acceleration. This behavior of the ZMP entail that the balance controller causes undesirable movement of the torso. Therefore, in this test the compensation output of the balance controller is filtered with a low-pass filter to remove the jumps cause by the ZMP. Note that this low-pass filter is not intended to be implemented on the real robot, since it is not the ZMP that is measured and used in the balance controller, but it is the CoP that is measured by the FTS. While the robot is in balance the ZMP and CoP are assumed to be alike, but the fast movements of the ZMP does not occur at the CoP.

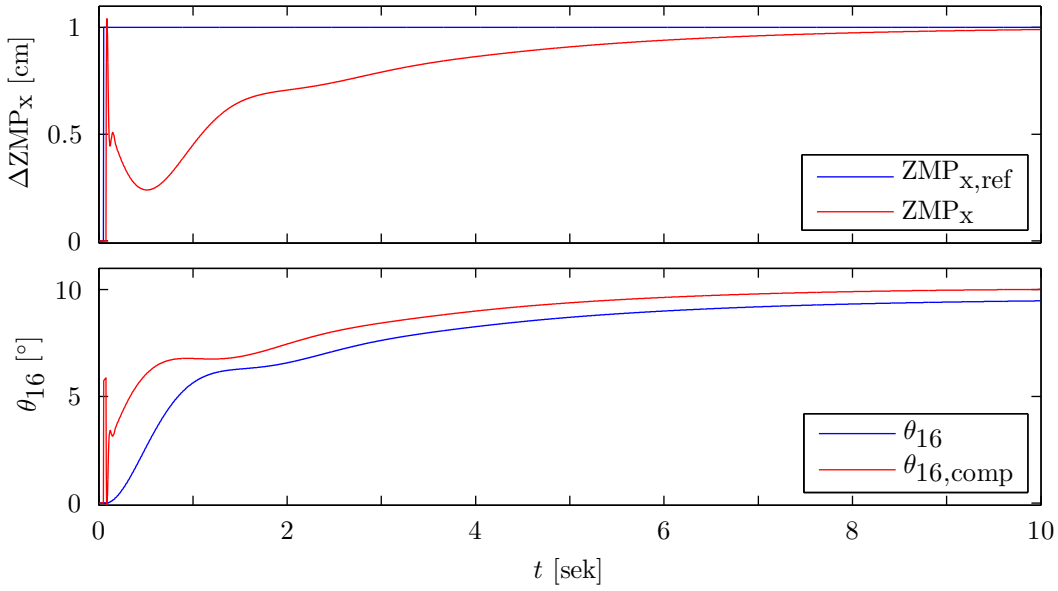


Figure 6.9: *The performance of the balance controller in the x-direction with a step on the reference. The top plot shows the reference and achieved ZMP, while the bottom plot shows the angle compensation from the balance controller and the achieved angle.*

From the test it is clear that the balance controller is able move the ZMP to the reference. It takes approximately 10 s to achieve the reference but after just 3 s the ZMP has reached two thirds of the reference.

6.5 Phase Estimator

The gait pattern for the human gait is described in Section 2.3 on page 14, and consists of different phases depending on the position of the legs. The control model of the robot depends on which foot is in contact with the ground, i.e. if the robot is standing on one foot or both feet. Therefore,

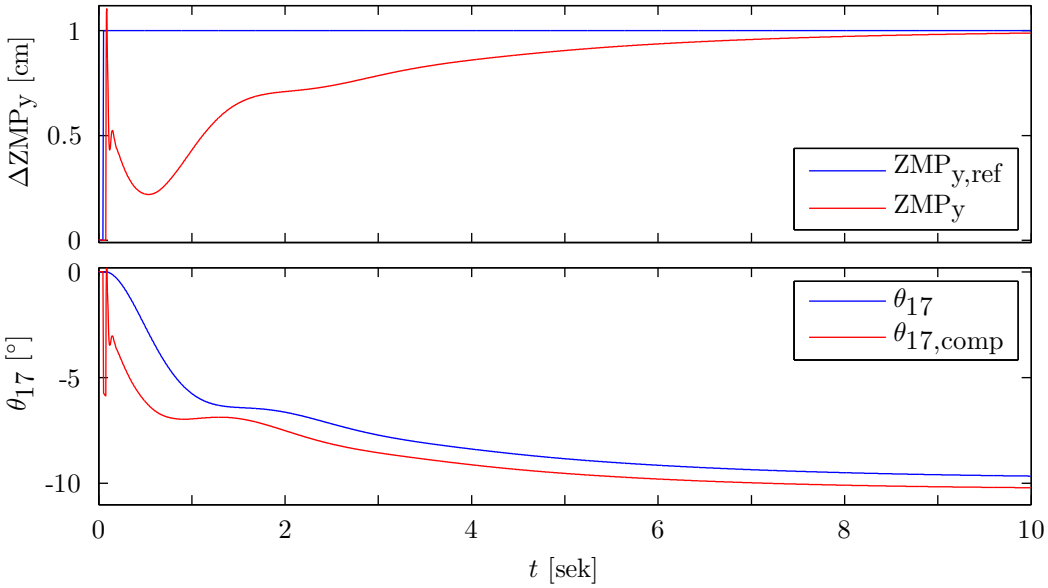


Figure 6.10: *The performance of the balance controller in the y-direction with a step on the reference. The top plot shows the reference and achieved ZMP, while the bottom plot shows the angle compensation from the balance controller and the achieved angle.*

the purpose of the phase estimator is to monitor the robot and determine which phase the robot is in.

6.5.1 Support Phases

The phases of the robot are defined in such a way that it should be able to perform human like gait. The phases are defined based on the walking method proposed by [Takahashi and Kawamura, 2002] and will be referred to as the Ken-method. The Ken-method make use of toe support which should enable a faster and more human like natural gait compared to more traditional human like dynamic gait methods. The Ken-method results in 8 different phases and is defined in Eq. (6.12) together with the initial double support phase; the phases are illustrated in Figure 6.11. It should be noted that for static gait the robot only attains three different phases, namely q_1, q_2, and q_5.

$$Q = \begin{cases} q_1 \equiv \text{SSP-L} & : \text{Single Support Phase Left} \\ q_2 \equiv \text{SSP-R} & : \text{Single Support Phase Right} \\ q_3 \equiv \text{SSP-LT} & : \text{Single Support Phase Left Toe} \\ q_4 \equiv \text{SSP-RT} & : \text{Single Support Phase Right Toe} \\ q_5 \equiv \text{DSP} & : \text{Double Support Phase} \\ q_6 \equiv \text{DSP-LT} & : \text{Double Support Phase Left Toe} \\ q_7 \equiv \text{DSP-RT} & : \text{Double Support Phase Right Toe} \\ q_8 \equiv \text{DSP-LT-RH} & : \text{Double Support Phase Left Toe and Right Heel} \\ q_9 \equiv \text{DSP-RT-LH} & : \text{Double Support Phase Right Toe and Left Heel} \end{cases} \tag{6.12}$$

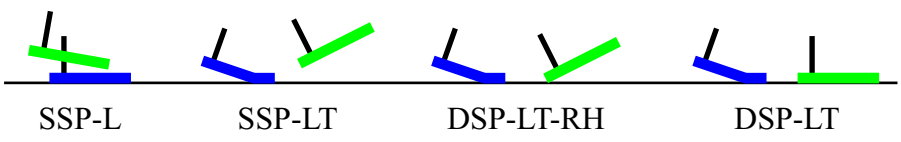


Figure 6.11: *The support phases of a half gait cycle of the Ken-method. The blue color illustrates the left foot, and the green the right foot.*

To determine which phase the robot is in, the heights of the toe and the heel are used. The height of the toe is determined from the toe joint and an extra point is defined on the heel to determine the height of the heel. From the toe and the heel height it is fairly simple to determine the phase as shown in Eq. (6.13).

$$Q = \begin{cases} q_1 & \text{if} \quad z_{\mathrm{r,t}} > \epsilon_1 \ \wedge \ z_{\mathrm{r,h}} > \epsilon_2 \ \wedge \ z_{\mathrm{l,t}} < \epsilon_3 \ \wedge \ z_{\mathrm{l,h}} < \epsilon_4 \\ q_2 & \text{if} \quad z_{\mathrm{r,t}} < \epsilon_1 \ \wedge \ z_{\mathrm{r,h}} < \epsilon_2 \ \wedge \ z_{\mathrm{l,t}} > \epsilon_3 \ \wedge \ z_{\mathrm{l,h}} > \epsilon_4 \\ q_3 & \text{if} \quad z_{\mathrm{r,t}} > \epsilon_1 \ \wedge \ z_{\mathrm{r,h}} > \epsilon_2 \ \wedge \ z_{\mathrm{l,t}} < \epsilon_3 \ \wedge \ z_{\mathrm{l,h}} > \epsilon_4 \\ q_4 & \text{if} \quad z_{\mathrm{r,t}} < \epsilon_1 \ \wedge \ z_{\mathrm{r,h}} > \epsilon_2 \ \wedge \ z_{\mathrm{l,t}} > \epsilon_3 \ \wedge \ z_{\mathrm{l,h}} > \epsilon_4 \\ q_5 & \text{if} \quad z_{\mathrm{r,t}} < \epsilon_1 \ \wedge \ z_{\mathrm{r,h}} < \epsilon_2 \ \wedge \ z_{\mathrm{l,t}} < \epsilon_3 \ \wedge \ z_{\mathrm{l,h}} < \epsilon_4 \\ q_6 & \text{if} \quad z_{\mathrm{r,t}} < \epsilon_1 \ \wedge \ z_{\mathrm{r,h}} < \epsilon_2 \ \wedge \ z_{\mathrm{l,t}} < \epsilon_3 \ \wedge \ z_{\mathrm{l,h}} > \epsilon_4 \\ q_7 & \text{if} \quad z_{\mathrm{r,t}} < \epsilon_1 \ \wedge \ z_{\mathrm{r,h}} > \epsilon_2 \ \wedge \ z_{\mathrm{l,t}} < \epsilon_3 \ \wedge \ z_{\mathrm{l,h}} < \epsilon_4 \\ q_8 & \text{if} \quad z_{\mathrm{r,t}} > \epsilon_1 \ \wedge \ z_{\mathrm{r,h}} < \epsilon_2 \ \wedge \ z_{\mathrm{l,t}} < \epsilon_3 \ \wedge \ z_{\mathrm{l,h}} > \epsilon_4 \\ q_9 & \text{if} \quad z_{\mathrm{r,t}} < \epsilon_1 \ \wedge \ z_{\mathrm{r,h}} > \epsilon_2 \ \wedge \ z_{\mathrm{l,t}} > \epsilon_3 \ \wedge \ z_{\mathrm{l,h}} < \epsilon_4 \end{cases} \tag{6.13}$$

where:

ϵ_i is a threshold determined experimentally [m]
$z_{\mathrm{r,t}}$ is the z-coordinate of the right toe [m]
$z_{\mathrm{r,h}}$ is the z-coordinate of the right heel [m]
$z_{\mathrm{l,t}}$ is the z-coordinate of the left toe [m]
$z_{\mathrm{l,h}}$ is the z-coordinate of the left heel [m]

In the implementation the condition for determining the phase is changed. This is done since the robot only is able perform static walk in this project and therefore it has shown to be more suitable to use the forces on the feet to determine the phase. Therefore the phase of the robot is determined as shown in Eq. (6.14).

$$Q = \begin{cases} q_1 & \text{if} \quad F_{\mathrm{z,l}} > \epsilon_1 \ \wedge \ F_{\mathrm{z,r}} < \epsilon_2 \\ q_2 & \text{if} \quad F_{\mathrm{z,l}} < \epsilon_1 \ \wedge \ F_{\mathrm{z,r}} > \epsilon_2 \\ q_5 & \text{if} \quad F_{\mathrm{z,l}} > \epsilon_1 \ \wedge \ F_{\mathrm{z,r}} > \epsilon_2 \end{cases} \tag{6.14}$$

where:

ϵ_i is a threshold determined experimentally [N]
$F_{\mathrm{z,l}}$ is the resultant vertical force on the left foot [N]
$F_{\mathrm{z,r}}$ is the resultant vertical force on the right foot [N]

To prevent fast changing between the phases during impact and lift off, a hysteresis effect is added.

6.5.2 Double Support Weight Factor

The phase estimator also returns a weight factor which is used in model utilized in the control system.

The weight factor should be determined based on the ratio between how much weight that is on each foot. This can be done in different ways, e.g. using the measured forces, the location of the CoM, or the location of the ZMP. One of the important aspect of the weight factor is that it must be 0 or 1 in the end of the double support phase, i.e. when the robot begins to lift the left or right foot. This is due to the fact that the inertia experienced by e.g. the angle joints is very high for the standing leg and very low for the swinging leg. Hence, if the left leg is about to be lifted and the left model is weighted just a little, the computed torque applied to the robot will become too large and result in an undesirable large angular acceleration caused by the large inertia.

The measured forces have some fluctuations due to noise and the movement of the robot. These result in unwanted fluctuations on the weight factor and since fluctuations on the weight factor will have direct impact on the torque applied to the robot, the measured forces are not used to calculate the weight factor.

Instead, the position of the GCoM is used to calculate the weight factor since there is very little fluctuations on the GCoM. Furthermore the set-up is made so that when the GCoM is located

inside the support area of one of the feet, the weight factor will become 0 or 1. Thereby, when the robot is in double support with the balance on one of the foot the dynamic behavior is assumed to be the same as when the robot is only standing on one leg. Since the GCoM and the ZMP are assumed to be identical during static gait the GCoM is chosen since the ZMP will have more small fluctuations.

The weight factor is calculated as shown in Eq. (6.15) using the shortest distances between the GCoM and the feet, as illustrated in Figure 6.12.

$$w = \begin{cases} 0 & \text{if GCoM is inside the left foot} \\ 1 & \text{if GCoM is inside the right foot} \\ \frac{d_{\text{left}}}{d_{\text{left}}+d_{\text{right}}} & \text{otherwise} \end{cases} \tag{6.15}$$

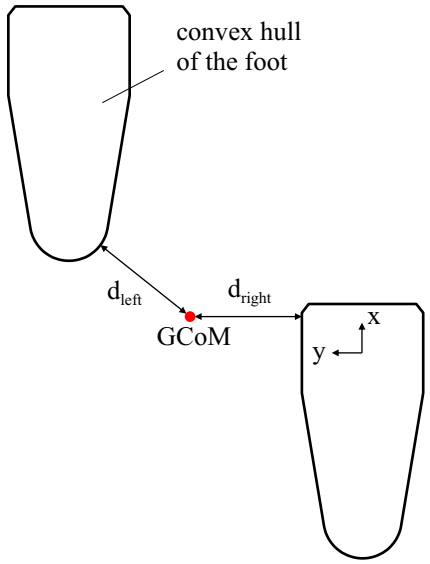


Figure 6.12: *The distances between the feet and the GCoM is used to weight the control input in double support.*

6.6 Motor Saturation

When a control system is implemented on **AAU-BOT1** it is important to ensure that the limitations of the DC-motors are obeyed. The DC-motors has limitations on how much torque they can produce, i.e. how high the motor current can be. If the motor current gets too high the DC-motor can sustain permanent damage, caused by the high internal temperature of the DC-motor. Therefore, the maximum current is dependent on the duration of the applied current.

For continues operation the motor current should be limited by the nominal value specified by the manufacturer to ensure that the motor does not overheat. But the nominal current can be exceeded for a limited time dependent on the subsequent cooling period. In Figure 6.13 it is illustrated how many times the nominal current the motor is allowed to be overloaded as function of the on time.

From the figure it is clear that the shorter time the motor is turned on the more current it is allowed to consume. During a walking cycle of the robot the load on each motor varies and the motors are therefore allowed to be overloaded during the peak load. The maximum allowed overload is four times the nominal [Pedersen et al., 2007, p. 54]:

$$i_{\text{m,max}} < 4i_{\text{N}} \tag{6.16}$$

At four times the nominal value the duration of the overload time is approximately 8 %, and the experimental work done by [Pedersen et al., 2007, p. 54] showed that the overload duration for

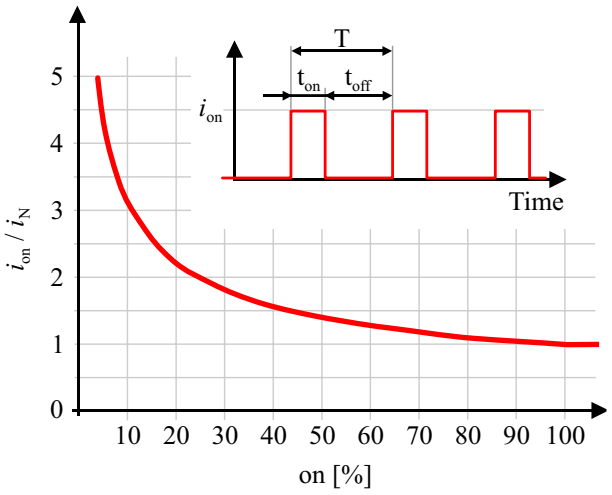


Figure 6.13: *Allowed overloading of the Maxon DC-motor as a function of the turned on time. [maxon motor, 2008b, p. 38]*

humanlike gait is between 5 % and 15 % depending on the joint. In this project the control of the robot is firstly designed to perform relatively slow static gait and therefore the motor saturation is set to two times the nominal value:

$$i_{\mathrm{m}} < 2i_{\mathrm{N}} \tag{6.17}$$

In Figure 6.14 the used motor current during a simulation of the static walking cycle is shown, along with the nominal current values. As it can be seen all the joints are below the nominal value or only shortly exceeds the nominal value.

Having established the control strategy, control structure, and designed the posture and balance controller as part of the control system for **AAU-BOT1**, the next chapter set up a trajectory for static walk.

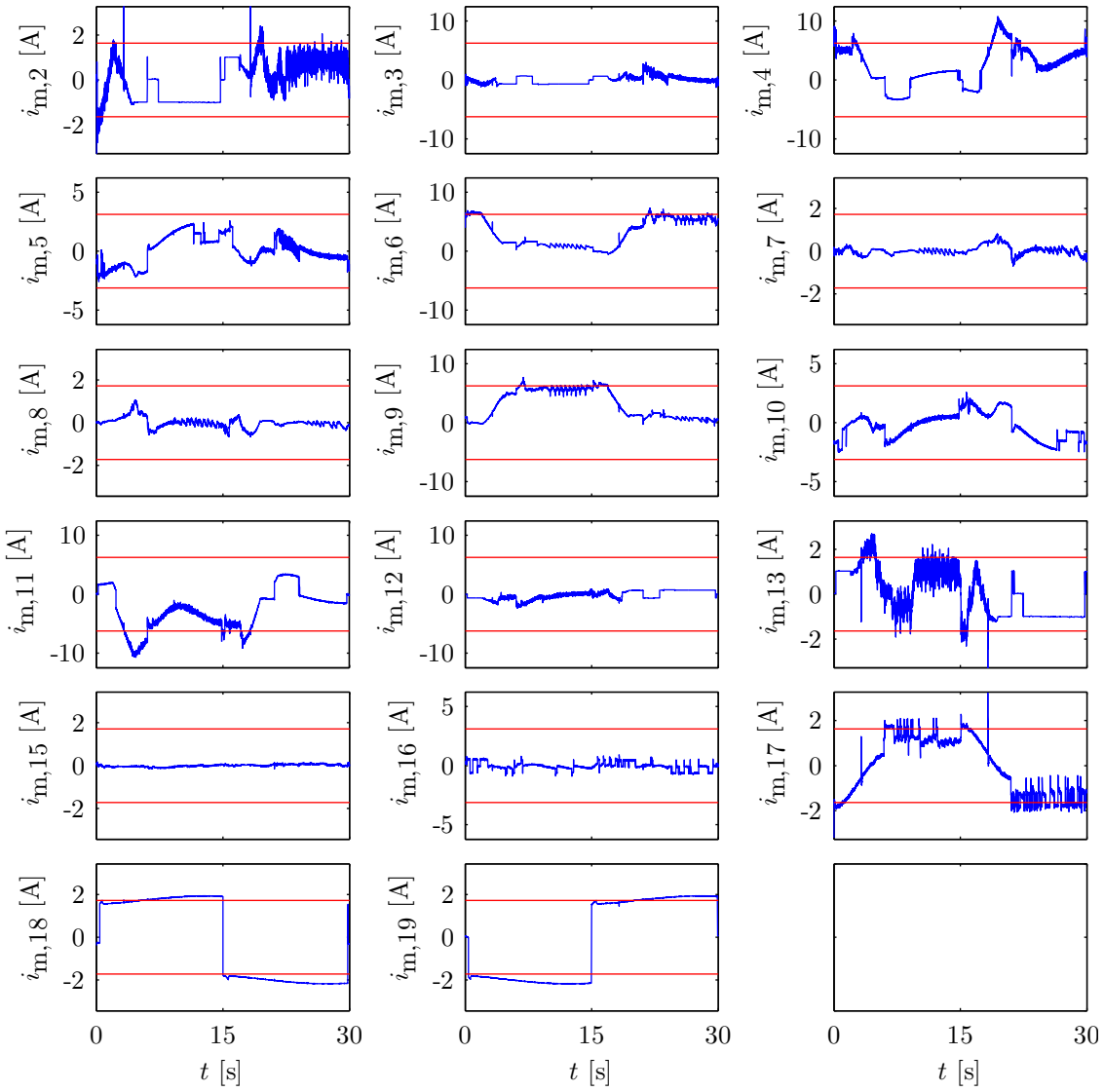


Figure 6.14: *The simulated motor current used for the static gait cycle. The red horizontal lines indicate the nominal current values for each joint.*

7 Trajectory Generation

As part of the entire control system this chapter handles the determination of a suitable trajectory for **AAU-BOT1** in order to perform static gait. The trajectory is designed in accordance with the physical limitations of the robot.

The design and simulation of the trajectory can also be seen on the attached DVD in [`Trajectory_Generation/`].

Contents

7.1 Introduction to Trajectory Generation

There are many challenges in making a biped robot walk, one of them being to ensure a reasonable walking pattern. To obtain a good walking pattern it is necessary to plan ahead to avoid unfortunate situations where smooth progress would be impossible.

One of the important aspects in trajectory generation is to investigate and decide whether the generation is going to happen online, offline, or a combination of both. Some advantages and disadvantages of the different strategies are given in the following bullet point list.

- **Online Trajectory Generation**
  The purpose of online trajectory generation is to calculate the optimal trajectory at all times dependent on the current posture of the robot and its surroundings. A change in the terrain or backlash in any joint would cause the optimum trajectory to change. If online trajectory generation is used such changes would not be a problem as the new optimum trajectory would be calculated and the robot would react in correspondence to it.

  True human walking indeed requires the use of online trajectory generation as humans naturally use all senses to adapt to the current situation. The problem of online trajectory generation is however that it usually requires rather high computational power. As concluded by [Jensen et al., 2008, p. 98] online trajectory generation is not an option for **AAU-BOT1** as it would require additional sensors to gain knowledge about the surroundings in order to adapt to them. A sensor that could be used if online trajectory generation were to be used is a camera combined with image recognition; this could give knowledge about obstacles or changes in the surface that might occur. Also radar could be useful to sense the progress of the surroundings.

  In addition to the missing sensors **AAU-BOT1** lacks the required computational power required to benefit from online trajectory generation. This is evident when considering the field trip taken by the project group behind [Jensen et al., 2008], where they gained knowledge about the DARPA Urban Grand Challenge car from Massachusetts Institute of Technology (MIT) that utilizes a very complex sensor system and 10 quad-core computers to handle the sensor data and perform online trajectory generation.

- **Offline Trajectory Generation**
  The generation of trajectories offline means that the trajectories for some or all of the joints are performed before the actual walking. The main advantage of this is that it is possible to use advanced and time consuming simulations and/or algorithms to determine the optimal trajectory. On the other hand the disadvantage is that changes in the movements or the surroundings during walk will result in a non-optimal trajectory and maybe even instability.

Another advantage given by offline trajectory generation is that it is known beforehand at all times how the robot is supposed to move. This feature means that it is easier to evaluate the performance of the controllers right away and also to stop the robot as soon as an irregular situation occurs in order to avoid any damage.

- **Hybrid Trajectory Generation**
  The last possibility is a combination of both online and offline trajectory generation. This can be obtained in different ways; one is to have a trajectory generated offline which is adjusted online to obtain higher energy efficiency or maybe higher stability. Another hybrid approach is to have several offline generated trajectories intended for different situations and then an online mechanism which chooses the one best suited for the current situation.

To actually generate the trajectories several different approaches can be taken. The following list gives a short description of some of the considered methods.

- **Human Trajectories**
  First of all it might seem compelling to use a pre-recorded human trajectory as reference trajectory for a biped robot. This would be obvious as humans have stable walk which should be obtained, and at the same time humans are said to have an energy optimal walk. As a consequence of this reasoning, experiments were carried out with motion capture of human trajectories by [Pedersen et al., 2007, pp. 21-26,135]. These experiments however did not end out in the way they expected so they ended up by using a different approach to trajectory generation.

  The problem with the human motion capture method is that the human walk is very dependent on the mass and inertia distribution between the different body parts, which varies from person to person. In order to utilize a human trajectory directly in a biped robot it is required that the mass and inertia distribution is very much alike for the robot and the person whose motion is captured. Also, even if such a condition could be fulfilled it is not certain that the trajectory would be as energy optimal on the robot as on the human as the complex muscle and bone structure of a human is very different from the DC motors and metal in the robot.

- **Adoption of Passive Walker Trajectory**
  Another appealing approach could be to adopt a trajectory from a so called biped passive walker like the one presented in [Collins et al., 2001]. Such a passive walker is capable of walking down a slope without any supply of energy except for that provided by gravity. The thought of adapting such a trajectory comes from the fact that no external energy source is needed, hence the walking must be rather energy optimized.

  On the other hand the trajectory of passive walkers are generally not very human-like, as stated in the article the walking pattern is often more similar to duck-like walk.

- **ZMP Stability**
  The stability of the ZMP is often considered in the trajectory generation and control for biped robots. As a consequence the method has been investigated in many ways and several approaches can be taken. One of these approaches is to simulate several possible trajectories and then choose the one with the highest stability [Choi et al., 2004].

  The method of ZMP stability can also be used in the way that trajectories are pre-defined for the feet and then the hip and torso motions are adjusted in order to obtain the highest stability [Jensen et al., 2008, p. 101]. In this way several trajectories for the hips and torso are simulated while the trajectories for the feet remain the same. In this method the different trajectories are determined based on Cubic Spline Interpolation which ensures smooth trajectories without sudden accelerations that might affect the ZMP in an undesirable way.

- **Inverted Pendulum**
  In the inverted pendulum method the entire robot is basically modeled as a single mass, concentrated in the CoM of the robot. This mass is then modeled as a 3D inverted pendulum where a force makes it swing in a forward motion from side to side. This method is described in [Wollherr, 2005, pp. 59-63] and [Kajita et al., 2001]. The method is also utilized in 2 dimensions in the Johnnie project [Löffler et al., 2004]. Furthermore, the method was also

considered by the first project group working on **AAU-BOT1** [Pedersen et al., 2007, pp. 140-143].

Unfortunately it is required that more than 70 % of the entire mass of the robot is located in the upper body which is not the case for **AAU-BOT1**. Additionally, the 3D inverted pendulum method assumes that no torque is applied in the ankle joints, which is definitely necessary on **AAU-BOT1** in order to overcome the friction in the joints.

As a first try in making **AAU-BOT1** walk the offline trajectory generation approach will be taken, using the ZMP stability method to obtain a suitable trajectory. The next section describes the trajectory generation.

7.2 ZMP Stability

As stated in the previous section it is decided to utilize an offline trajectory generation based on ZMP stability to determine an appropriate trajectory for **AAU-BOT1**. This decision is based on the fact that the on-board computer does not have enough computational power to perform online trajectory generation.

Roughly, two different approaches can be taken towards the ZMP stability method. One approach is to choose an appropriate ZMP trajectory to follow and then use inverse formulas to determine the joint positions that result in the desired ZMP trajectory. This approach is however rather complex to implement and may not always be effective as it can easily result in inappropriately large accelerations of specific joints to obtain the desired ZMP trajectory [Huang et al., 2001].

Another approach to the ZMP stability method is more experimental. The first step is to define joint trajectories for the feet during a walking cycle and then perform numerous simulations with small variations in the trajectories for the hips and torso. Then, from the simulation data, the obtained ZMP trajectory can be calculated for each simulation, and the best trajectory can be chosen.

The best trajectory can be defined in many ways depending on which parameters that are considered most important. If the absolutely most optimal trajectory is to be chosen it means finding the trajectory with the largest stability margin while minimizing the energy consumption. Also other parameters might be considered, e.g. small changes in the angles might be desired as that would result in more fluent trajectories. For the first attempt of making **AAU-BOT1** walk the energy consumption will not be considered, only the stability margin is considered. This is chosen to obtain the trajectory where the stability margin is as large as possible to have a large margin for errors, e.g. to make it easier for the controller to cope with modeling inaccuracies.

For this project the method based on multiple simulations of trajectory variations will be used to determine the best trajectory. The comparison of the simulated trajectories is performed by comparing the stability margin for each of the trajectories. The stability margin is defined as the minimum distance from the ZMP to the convex hull defining the SA, as shown in Figure 7.1. If the ZMP is located outside the SA, then it means that the situation is not stable and the stability margin will be a negative number.

The procedure of the trajectory simulation is illustrated in Figure 7.2. The illustrated procedure is utilized to get a result from each of the trajectory variations, which means it is a single iteration of the entire search for the best trajectory. Note that in static gait the ZMP and GCoM are assumed to be identical and therefore the GCoM will be used in the further design of the trajectory due to the easier computation.

The input to the system is the time series of parameters defining the posture to be tested, i.e. the positions and orientations of feet, hip, torso, and arms. In the figure V_l indicates the validity of the leg length, i.e. if both legs are long enough to reach the desired positions. The symbol V_a is symbolizing if all angle constraints are met, while V_sm indicates if the stability margin is a positive number, which is true when the GCoM is inside the SA. When all validity conditions are fulfilled it results in V being true; this is one of the outputs from the simulations. The other output is the stability margin d_sm.

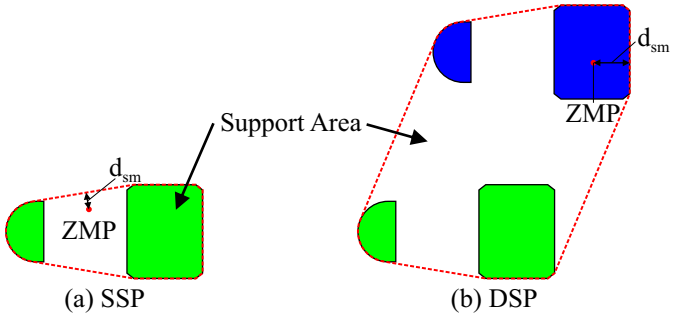


Figure 7.1: *Definition of stability margin, d_{sm}. The minimum distance from the ZMP to the support area (SA) is the stability margin, i.e. the larger the number the better. If the ZMP is outside the SA it indicates that the situation is unstable.*

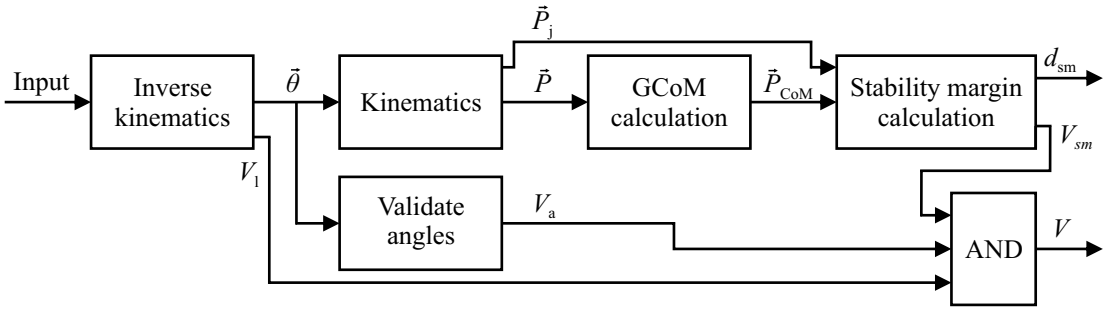


Figure 7.2: *The procedure for a single one of the trajectory simulations. The procedure here is repeated for each of the simulations.*

7.3 Definition of Trajectories

To define the trajectories to be simulated, a number of parameters are used to describe them. As written in the previous section the trajectories for both feet are specified first. It is decided to walk with a constant distance between the feet and keep both feet flat at all times.

The parameters in Table 7.1 are used to define the gait pattern in static gait. The parameters where multiple numbers are stated are those that are varied during the different simulations. Multiple iterations are performed to find the best trajectory; the parameters that are varied in the first search are denoted search A in the table, while the parameter that are varied in the final search are denoted search B.

It is decided to use Cubic Spline Interpolation to actually create the trajectories for the joints. This technique creates a line which passes through a specified set of points and is defined by a set of piecewise fourth-order polynomials. To make the Cubic Spline Interpolation the `MATLAB` function `spline` is used.

The gait parameters stated in Table 7.1 are shown in the series of pictures in Figure 7.3 which shows a single forward step.

Parameter	Symbol	Value(s)
Fixed parameters		
Average speed	v_{avg}	0.01 m/s
Step length	l_{step}	0.30 m
Gait cycle time	t_{cycle}	$\frac{l_{\text{step}}}{v_{\text{avg}}} = 30$ s
Step time	t_{step}	$0.5t_{\text{cycle}} = 15$ s
SSP time	t_{SSP}	$0.6t_{\text{step}} = 9$ s
DSP time	t_{DSP}	$0.4t_{\text{step}} = 6$ s
Max. ankle height	$h_{\text{a,max}}$	0.133 m
Max. ankle height length	$l_{\text{a,max}}$	$0.5l_{\text{step}} = 0.15$ m
Max. ankle height time	$t_{\text{a,max}}$	$0.6t_{\text{step}} = 9$ s
Distance between feet	y_{fd}	$\sum_{i=1}^{13} a_{i,\text{z}} - 0.07$ m
Min. arm pitch rotation	$\theta_{18/19,\text{min}}$	$-\frac{\pi}{6}$ rad
Max. arm pitch rotation	$\theta_{18/19,\text{max}}$	$\frac{\pi}{6}$ rad
Varied parameters in search A		
Min. pelvis height	$h_{\text{p,min}}$	$[0.71, 0.72, ..., 0.79]$ m
Max. pelvis height	$h_{\text{p,max}}$	$h_{\text{p,min}} + 0.03$ m
Max. pelvis x-movement	$x_{\text{p,max}}$	$[0.06, 0.075, ..., 0.18]$ m
Max. pelvis y-movement	$y_{\text{p,max}}$	$0.5y_{\text{fd}} \cdot [0.8, 0.85, ..., 1.3]$ m
Max. torso roll rotation	$\theta_{17,\text{max}}$	$\frac{\pi}{14} \cdot [0, 0.4, ..., 2.4]$ rad
Varied parameters in search B		
Min. pelvis height	$h_{\text{p,min}}$	0.78 m
Max. pelvis height	$h_{\text{p,max}}$	$h_{\text{p,min}} + [0.025, 0.03, 0.035]$ m
Max. pelvis x-movement	$x_{\text{p,max}}$	$[0.135, 0.138, ..., 0.165]$ m
Max. pelvis y-movement	$y_{\text{p,max}}$	$0.5y_{\text{fd}} \cdot [1.04, 1.06, ..., 1.16]$ m
Max. torso roll rotation	$\theta_{17,\text{max}}$	$\frac{\pi}{14} \cdot [1.6, 1.8, 2.0]$ rad

Table 7.1: *Parameters defining the static gait trajectories. The parameters are illustrated in* Figure 7.3.

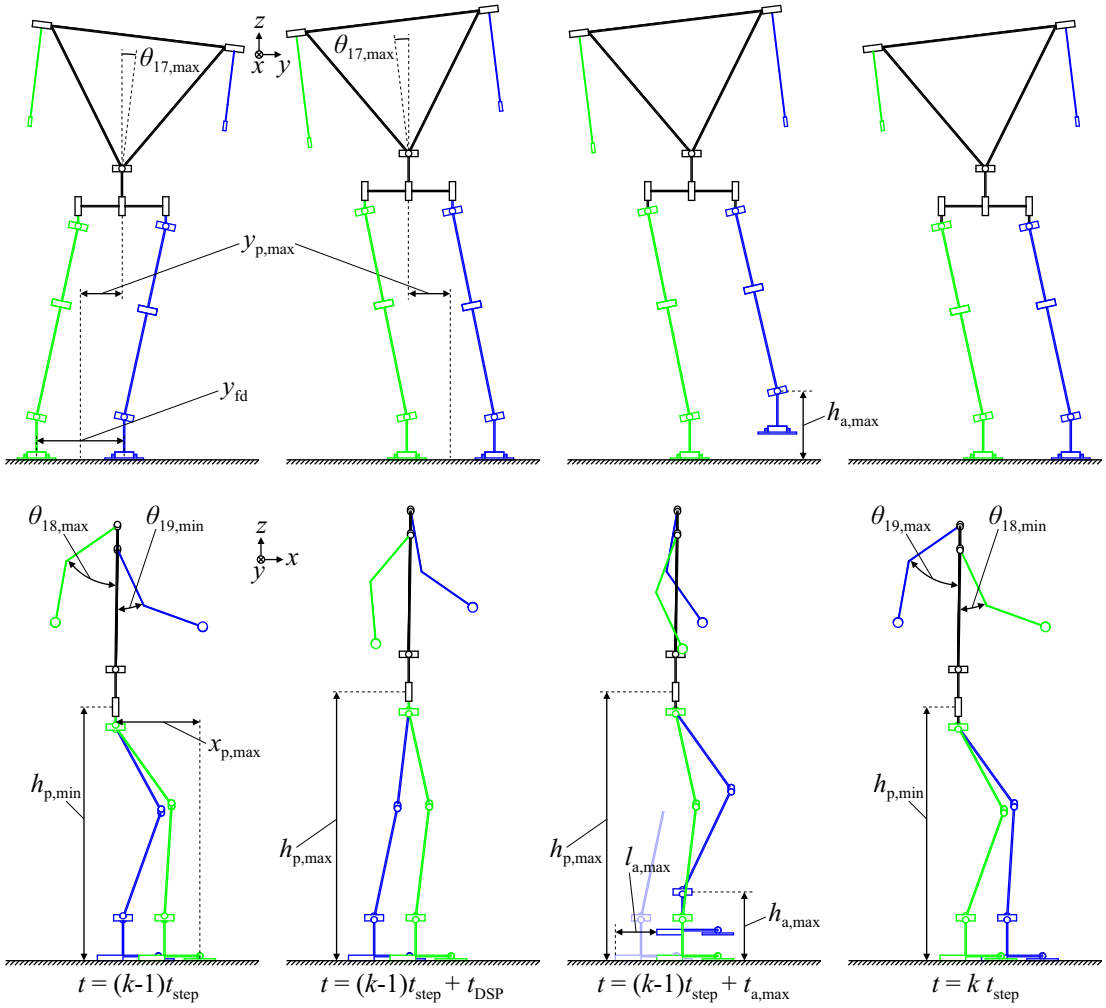


Figure 7.3: *The four defined situations which results in a single step. To complete the gait cycle each situation should just be mirrored in order to take a step with the other foot in front. The top row shows the robot in the frontal plane, while the bottom row shows the same situations seen in the sagittal plane. To ease the readability of the figure the right arm and leg are marked with green and the left arm and leg are marked with blue.*

7.4 Simulation of Trajectories

To determine the best of the trajectories they are simulated using the kinematic model described in Chapter 5 on page 36, according to the illustration in Figure 7.2 on page 79. The result of each of the simulations is a set of numbers defining the stability margin, d_{sm}, for each simulation time step. A plot of the stability margin for one of the trajectories is shown in Figure 7.7 on page 85. In the figure the most critical point is around 6 s where the margin is smallest, hence this is the number used to represent that specific trajectory. The trajectory with the highest stability margin at the most critical point is chosen as the best trajectory.

To obtain a useful trajectory it is necessary to include the physical limitations of the robot in the trajectory simulations. In order to do this a series of tests have been performed on the physical robot. The physical angular limitations were determined by moving the joints one at a time to their physical limit and then performing a reading of the angle. While doing so the safety circuits mounted on the joints were also adjusted to make them trigger at the correct angle. The determined limitations are listed in Table 7.2.

Joint name	**Angle symbol**	**Minimum angle**	**Maximum angle**
Right toe pitch	θ_1	-? °	? °
Right ankle roll	θ_2	-10.99 °	42.17 °
Right ankle pitch	θ_3	-24.95 °	25.67 °
Right knee pitch	θ_4	-67.52 °	5 °
Right hip pitch	θ_5	-15.17 °	37.18 °
Right hip roll	θ_6	-18.76 °	15.73 °
Right hip yaw	θ_7	-? °	? °
Left hip yaw	θ_8	-? °	? °
Left hip roll	θ_9	-18.76 °	15.73 °
Left hip pitch	θ_{10}	-37.18 °	15.17 °
Left knee pitch	θ_{11}	-5 °	67.52 °
Left ankle pitch	θ_{12}	-25.67 °	24.95 °
Left ankle roll	θ_{13}	-10.99 °	42.17 °
Left toe pitch	θ_{14}	-? °	? °
Waist yaw	θ_{15}	-? °	? °
Waist pitch	θ_{16}	-60 °	60 °
Waist roll	θ_{17}	-50 °	50 °
Right arm pitch	θ_{18}	–	–
Left arm pitch	θ_{19}	–	–

Table 7.2: *The physical limitations of* **AAU-BOT1**. *The angles where a ? is indicated are those which have not been determined as they are of no importance when the robot walks flat footed and with constant orientation of pelvis. The arms only have a – as they do not have any physical limitations.*

It is important that no angle in the used trajectory ever gets close to the limitations as the safety circuit in a joint will shutdown the corresponding EPOS immediately if the limitation is violated. This behavior will leave the robot in an unfortunate state until everything has been reset. As the controllers need some freedom to act the angular limitations used in the trajectory search are set stricter than the physical limits. The physical limitations are however rather strict in relation to static walk as the robot is only designed for dynamic walk. To make it possible to find a suitable trajectory within the limitations and still provide some regulation margin to the controller the limitations are set 2 degrees harder than the physical limitations listen in Table 7.2.

In addition to the angular limitations it is necessary to ensure that both legs are capable of reaching the desired positions when taking the desired position of the pelvis into consideration.

All the limitations are implemented in the trajectory search `MATLAB Simulink` model as Assertion blocks which ensures that as soon as a limitation is violated the simulation is ended and marked as invalid in the search. For all valid trajectories the minimum safety distance is saved as result and it is marked as valid.

The search for the best trajectory is performed using a set of for loops running through each of the parameter variations specified in Section 7.3 on page 79, through the procedure specified in Figure 7.2 on page 79. The result of the initial search (search A) for the best trajectory is shown in Figure 7.4. In order to find a suitable trajectory multiple searches have been performed with some changes in the trajectory specifications in the directions indicated by the previous search.

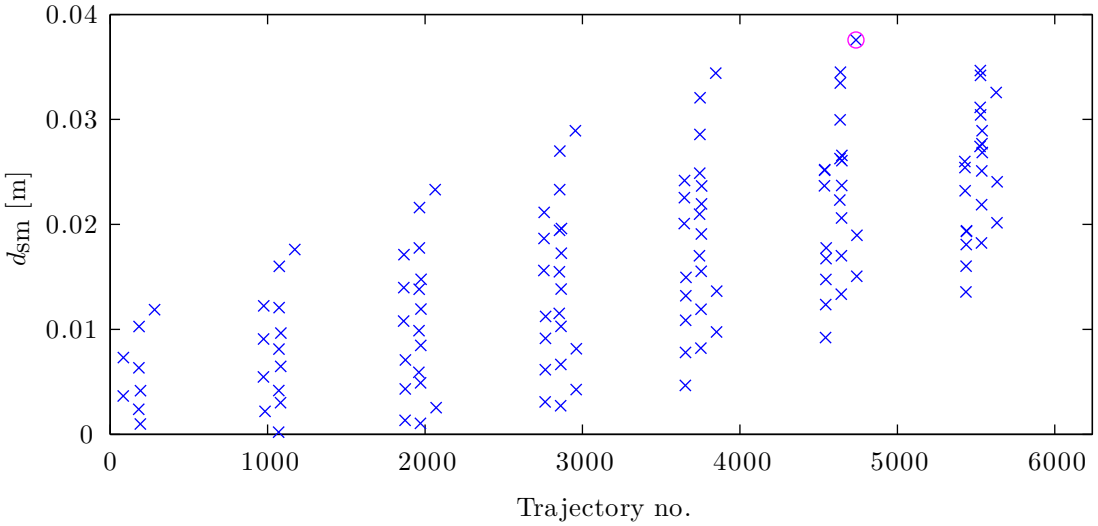


Figure 7.4: *The plot illustrates the trajectories simulated in search A for the best trajectory. All trajectories where one or more limitations are violated, i.e. angles and/or the physical reach of either leg, are hidden, which means that only valid trajectories are marked. The distance indicated is the shortest distance from the CoM to the boundary of the SA at any time through the simulation. The trajectory marked with a purple circle is the best trajectory in this search.*

The result of the last trajectory search performed (search B) is shown in Figure 7.5, where the best trajectory is marked with a circle.

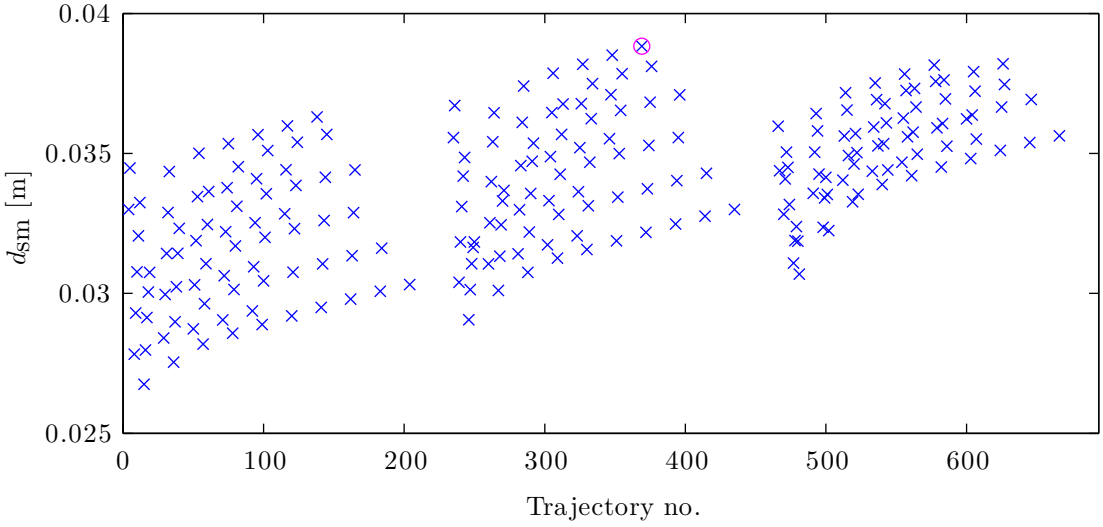


Figure 7.5: *The plot illustrates the trajectories simulated in search B for the best trajectory. All trajectories where one or more limitations are violated, i.e. angles and/or the physical reach of either leg, are hidden, which means that only valid trajectories are marked. The distance indicated is the shortest distance from the CoM to the boundary of the SA at any time through the simulation. The trajectory marked with a magenta circle is the best trajectory in this search.*

The parameters of the best found trajectory for static walk are shown in Table 7.3. To illustrate

the trajectory, all of the angles are plotted along with the limitations in Figure 7.6.

Parameter	**Symbol**	**Value(s)**
Min. pelvis height	$h_{\text{p,min}}$	0.78 m
Max. pelvis height	$h_{\text{p,max}}$	$h_{\text{p,min}} + 0.03$ m
Max. pelvis x-movement	$x_{\text{p,max}}$	0.147 m
Max. pelvis y-movement	$y_{\text{p,max}}$	$0.5 y_{\text{fd}} \cdot 1.12$ m
Max. torso roll rotation	$\theta_{17,\text{max}}$	$\frac{\pi}{14} \cdot 1.8 = 0.4039$ rad $= 23.14$ °

Table 7.3: *Parameters defining the best static gait trajectory found in the searches.*

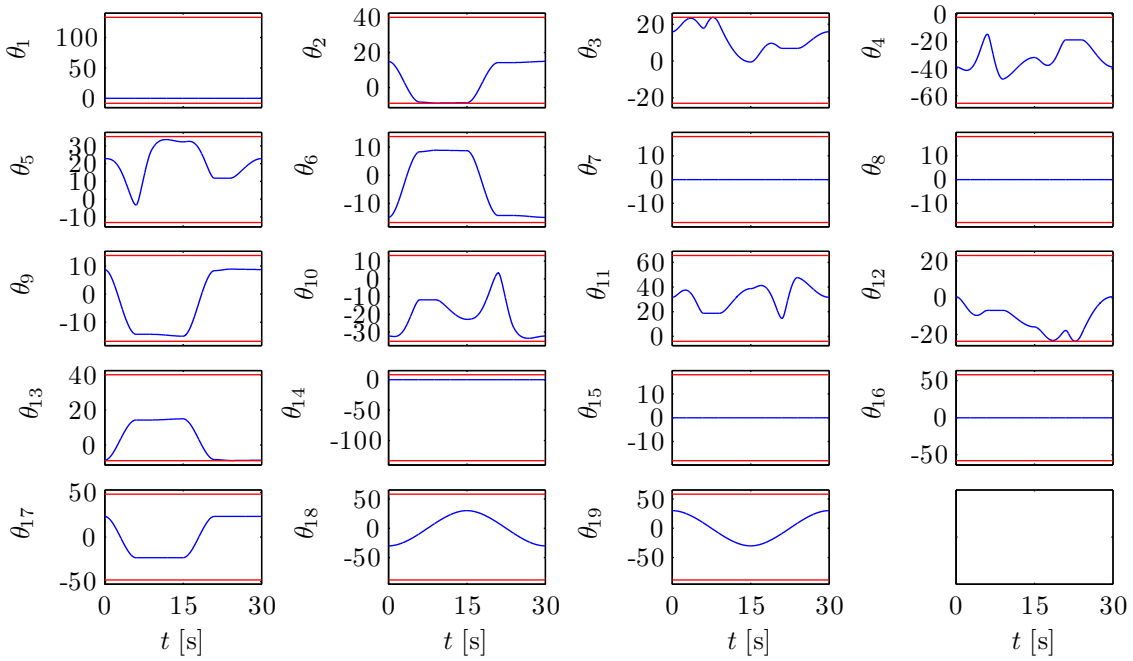


Figure 7.6: *The resulting angle trajectories for the best static gait trajectory. The red lines illustrate the limitations used in the trajectory search.*

Furthermore, to illustrate the stability of the trajectory the stability margin for a full gait cycle is shown in Figure 7.7, and the progress of the GCoM is shown in Figure 7.8.

To provide a better physical understanding of the movements of the robot during a step it is illustrated by 4 individual plots from different angles in Figure 7.9.

A static balanced gait trajectory has been designed which include the physical limitations of the joint angles. In the following chapter a Kalman estimator is designed to estimate the unmeasured joint angles and velocities.

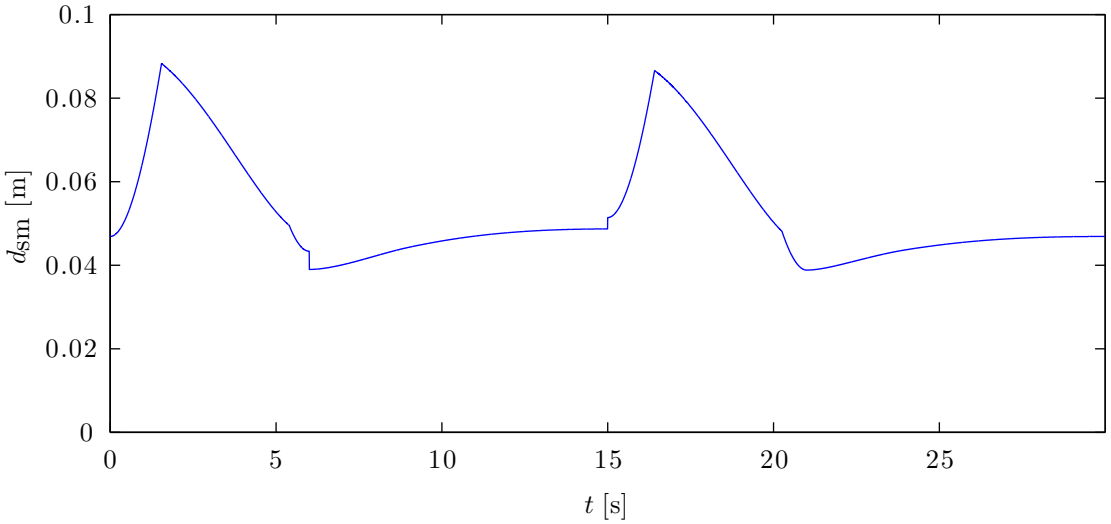


Figure 7.7: *The resulting stability margin for the best static gait trajectory.*

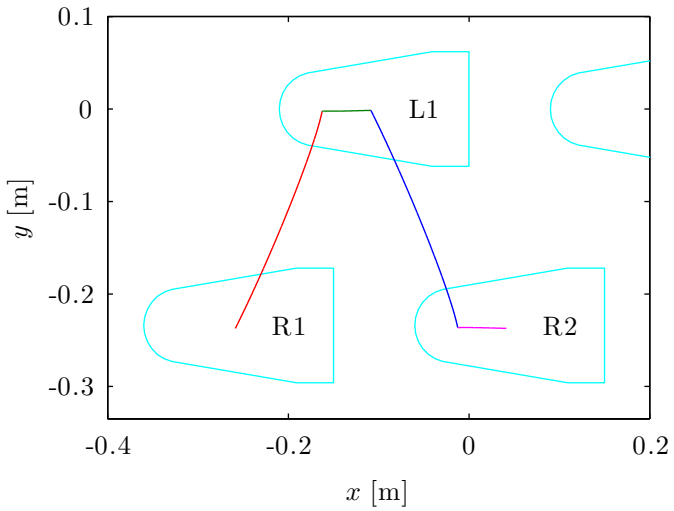
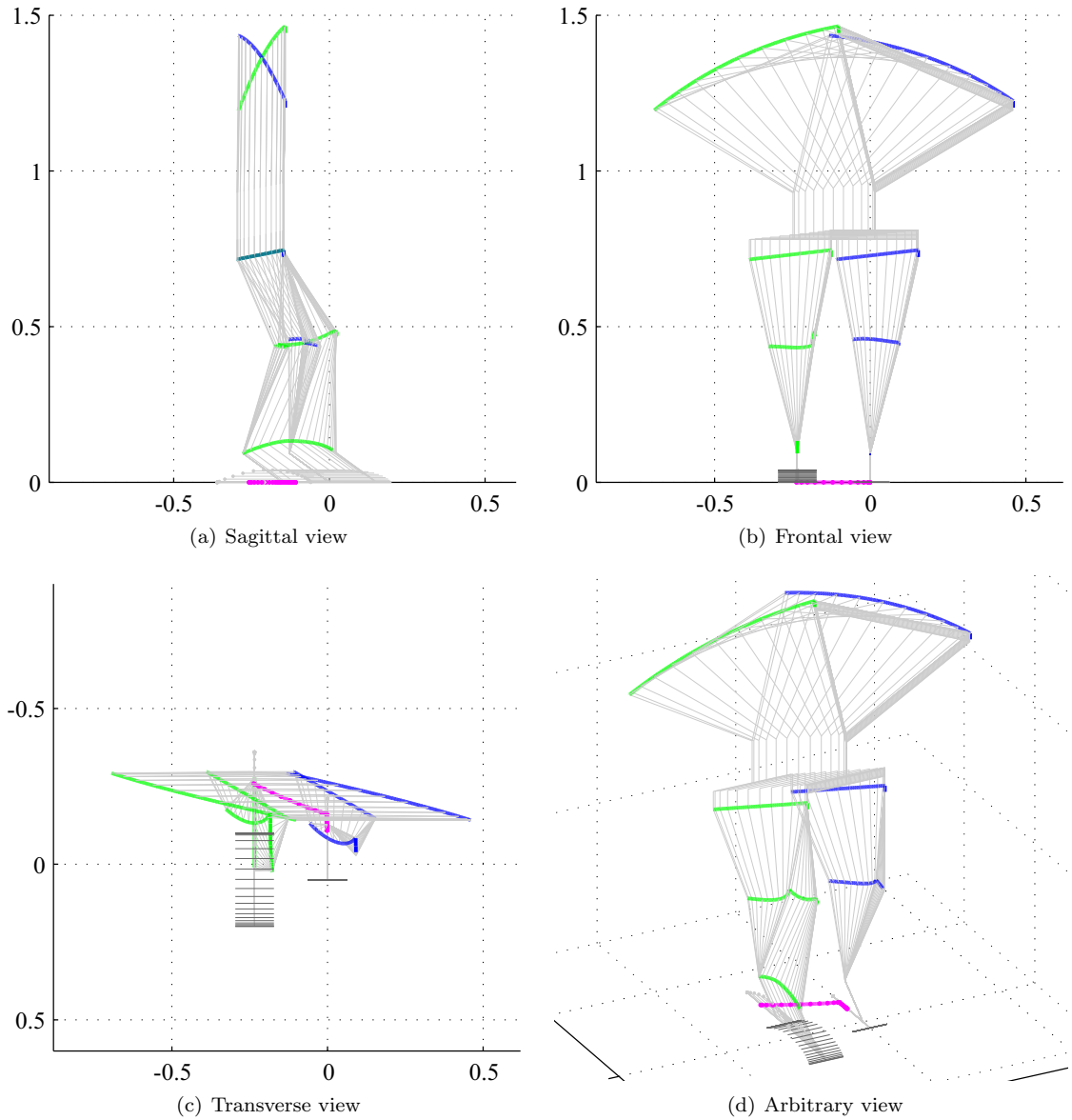


Figure 7.8: *The resulting CoM trajectory for the best static gait trajectory. The red line indicates the period where the robot is in the first double support phase with the feet in positions marked L1 and R1. The green line is the left single support phase. The blue line is the second double support phase with the feet in positions marked L1 and R2. The magenta line is the right single support phase.*



(a) Sagittal view

(b) Frontal view

(c) Transverse view

(d) Arbitrary view

Figure 7.9: *Illustration of the best trajectory seen from 4 different angles. The robot is plotted once for each second of a single step, i.e. half of the gait cycle. The green lines show the path of 4 different joints in the right side and just as the blue lines do it for the left side; this is done to make it easier to see how the different parts of the robot move. The magenta line shows the path of the GCoM.*

8 Estimator Design

This chapter describes the choice and design of an estimator for the system. The estimator must be able to provide an estimate of the necessary unmeasured system states. The implementation of the estimator together with the rest of the control system and the model can be seen in the `Matlab Simulink` model located on the DVD in [`Model_with_Control_System/`].

Contents

8.1 Introduction to Estimator Design

The purpose of the estimator is to provide an appropriate estimate of the actual joint angles, $\vec{\theta}$, and angular velocities, $\dot{\vec{\theta}}$, to be used in the feedback control loop. The problem is, as described in Section 5.4 on page 50, that the angle and angular velocity measurement provided at the robot are those of the motor axle and not of the joint axle, even though these are the ones that should be controlled.

It is illustrated in Figure 8.1 how the estimator is going to interact with the remaining system.

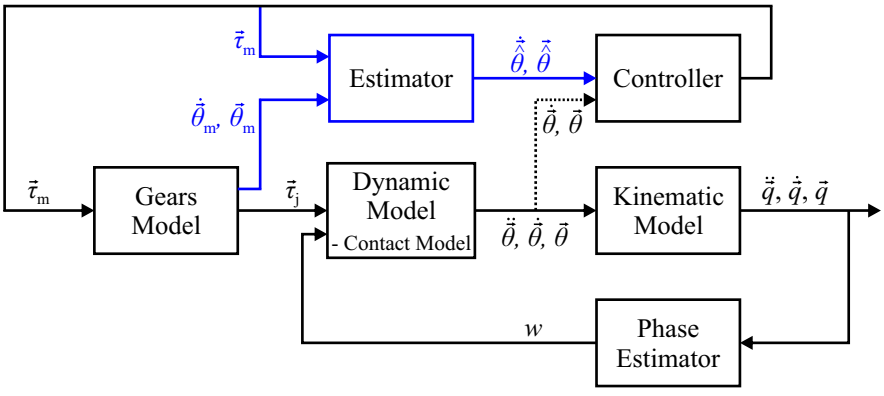


Figure 8.1: *The incorporation of the estimator in the remaining system. The dotted line is the desirable connection, but it is not possible to make that connection due to the available measurements at the robot.*

The problem of the missing measurements of $\dot{\vec{\theta}}$ and $\vec{\theta}$ is made clear with an experiment. The experiment is performed on the full simulation model of the robot described in Chapter 5 on page 36 because it can be done without any risk of damaging the robot, but the same behavior is observed

at the physical robot. Also, the second part of the experiment described below cannot be performed at the physical robot due to missing measurements.

Basically the experiment is two identical simulations with the only difference being the measurements utilized in the controller. Note that the controller used in the experiment is the one designed in Chapter 6 on page 61. In the first simulation the measurement input to the controller is the angle and angular velocity of the motor shaft, $\dot{\vec{\theta}}_{\mathrm{m}}$ and $\vec{\theta}_{\mathrm{m}}$, corrected by the gear ratio, $\frac{1}{\vec{G}}$, while in the second simulation the controller measurement input is the angle and angular velocity of the joint shaft, $\dot{\vec{\theta}}$ and $\vec{\theta}$. The angular velocity of both simulations is shown for a single joint in Figure 8.2. By

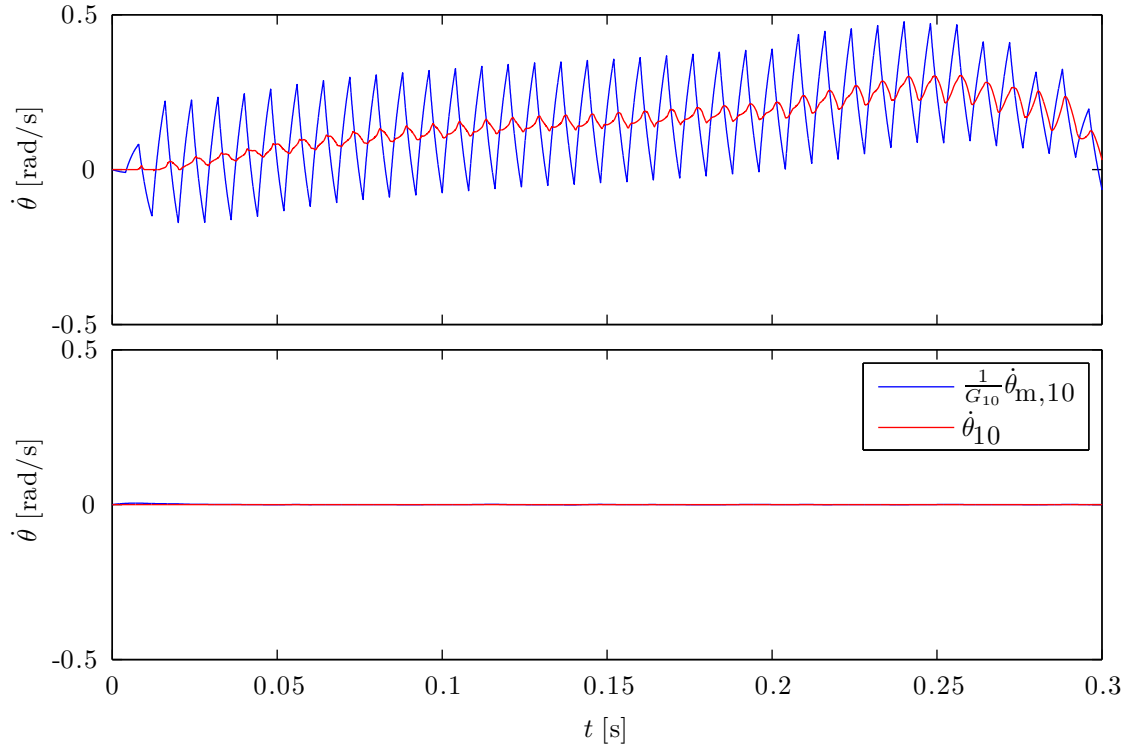


Figure 8.2: *The top plot shows the motor and joint angular velocities for a simulation with the motor angle and angular velocity as controller input and the bottom plot for a simulation with the joint angle and angular velocity as controller input. Note that the angular velocity of the motor shaft has been divided by the gear ratio to get equal scales for the two variables. The plots are taken from the beginning of a simulation of the robot following the designed static gait trajectory.*

looking at the figure it is clear that the situation at the top plot is very undesirable as the velocity is changed constantly and no smooth movements can be expected. It can also be observed that the problem is reduced to a negligible extent when the measurement of the joint shaft is utilized, which motivates the design of an estimator to provide estimates of the missing measurements.

8.2 Estimator Techniques

When designing an estimator several approaches can be taken. The purpose of this section is to provide a short description of the methods considered for this project, together with a reflection on some of the hurdles with the methods.

8.2.1 Linear Kalman Filter

The first method considered was a standard Kalman filter. The problem of using such a filter for the robot is that it operates on a linearized version of the considered system. On many linear and

slightly non-linear systems this is a useful approach, but unfortunately it is not very useful for the robot as this is a highly non-linear system.

When some joint on the robot is changed just a few degree it changes the dynamics of the robot completely, which means that the mass matrix, $\boldsymbol{M}$, is significantly different. At the same time the load in each joint is also changed significantly. These two factors together means that a linearization of the full model will only be valid in a narrow region around the linearization point. Hence, it is difficult to design a single useful Kalman estimator.

One solution to the problem could be to design multiple Kalman estimators for different linearization points of the robot and then switch between them according to the current situation. This approach will however require quite a large number of different Kalman estimators to be designed; hence, keeping track of these in a proper and useful way would not be an easy task. At the same time the switches between the Kalman estimators would be cumbersome as they are all valid in only a narrow region.

8.2.2 Extended Kalman Filter

When the system is highly non-linear and a single linearized model is inapplicable for the entire operating range, it might seem obvious to choose an extended Kalman filter to solve the problem. This is however not the case for **AAU-BOT1**. The problem in applying an extended Kalman filter to **AAU-BOT1** is that the first order derivative of the system is needed in order to do a first order Taylor approximation of the system in the current operating point at each sample time.

The robot model is indeed differentiable, but the system is so complex that it has not been possible to get access to any computer that had the capacity to do the first order differentiation of the entire system symbolically. Even if the model could be successfully differentiated, the resulting equations would most likely be of such an extent that it would be impossible to solve them online at each sample time as required to implement an extended Kalman filter.

8.2.3 Unscented Kalman Filter

The unscented Kalman filter works by simulating a number of sigma points through the entire non-linear model. The sigma points are determined from the mean value from the last sample time and the covariance matrix of the system states. The things needed to implement an unscented Kalman filter is the unmodified non-linear model and enough computational power to pass all sigma points through the model at each sample time.

Even though this approach might seem straight forward, that is not the case for the considered system. The system model has such high complexity that it is impossible to run it the required number of times each sample time, due to lack of computational power. Due to this problem some special precautions must be taken. Due to the relatively slow dynamics of the robot, as a consequence of the large moments of inertia, it is possible to use the same calculations for the mass matrix, $\boldsymbol{M}$, and the load in each joint, $\vec{\tau}_l$, in all of the sigma points each sample time. By doing so the demands to the computational power needed is significantly reduced, as the computational heavy part of the model only needs to be calculated ones for each sample step.

By taking the approach briefly described for the unscented Kalman filter it is possible to use such a filter to estimate the joint angles at **AAU-BOT1**.

8.3 Design of the Unscented Kalman Filter

As mentioned in the previous section the only thing needed to apply an unscented Kalman filter (UKF) to the system is the non-linear system model. The needed full system model is comprised of the dynamic model finalized in Subsection 5.3.4 on page 49 and the gear model set up in Section 5.4 on page 50. The unscented Kalman design presented here is mainly based on [Julier and Uhlmann, 1997], but is also inspired by [Thrun et al., 2005, pp. 65-71].

When the full model is combined, the full state vector consists of the angle and angular velocity of all motor shafts, the gear torsion angles, and the angle and angular velocity of all joints, i.e. it becomes:

$$\vec{x}(t) = \begin{bmatrix} \vec{\theta}_{\mathrm{m}} \\ \dot{\vec{\theta}}_{\mathrm{m}} \\ \vec{\theta}_{\Delta} \\ \vec{\theta} \\ \dot{\vec{\theta}} \end{bmatrix} \tag{8.1}$$

This makes a total of $17 \cdot 5 = 85$ states because the toe joints are not included as they are unactuated and the robot is intended to walk flat footed during static walk.

The way the unscented Kalman filter works is by using applied statistics together with the non-linear system model. The procedure is based on the intuition that "*it is easier to approximate a Gaussian distribution than it is to approximate an arbitrary nonlinear function or transformation*" [Julier and Uhlmann, 1997].

The approach uses a number of sigma points chosen so that their sample mean and sample covariance correspond to those estimated for the state vector. These points are transformed through the non-linear process model resulting in a cloud of transformed points with a new sample mean and covariance, which are used as a priori predictions of the new state vector. The process of selecting sigma points and transforming them is illustrated in Figure 8.3.

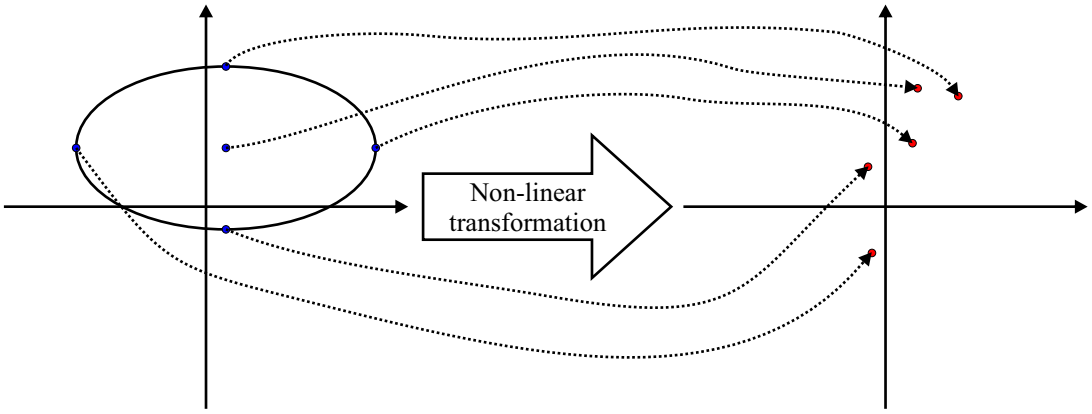


Figure 8.3: *The principle of the unscented transformation. The blue points are the sigma points including the mean, and the oval illustrates the covariance. The sigma points are propagated through the non-linear model to yield a cloud of transformed points. The figure is redrawn from a figure in [Julier and Uhlmann, 1997].*

Basically, each time step of the UKF algorithm consists of a number of steps that are listed below:

1. Create a number of sigma points based on the mean.
2. Propagate all sigma points through the process model.
3. Calculate the predicted mean of the state vector.
4. Run each prediction point through the observation model.
5. Calculate the predicted observation mean.
6. Compute the predicted state and innovation covariance matrices and the cross correlation matrix.
7. Determine the new Kalman gain matrix.
8. Calculate the state vector estimate.
9. Calculate the state covariance estimate.

Each of the steps in the UKF is explained in the following two subsections. The first subsection covers the steps related to the prediction of the current time step, while the second subsection describes the steps required to perform the update of the current state vector and its covariance. The numbers and the descriptions given in the list above are used to separate the different parts within the subsections.

8.3.1 Prediction Steps

The prediction steps in the Kalman filter creates a priori predictions of the needed variables based on the estimated mean of the state vector and its covariance matrix from the previous time step.

1. Create a number of sigma points based on the mean

The sigma points are chosen based on the scaled symmetric sigma point distribution to approximate the n-dimensional random variable representing the state vector. The random variable is approximated by $2n+1$ sigma points, according to the following equations:

$$\vec{\mathcal{X}}_i(k-1|k-1) = \begin{cases} \hat{\vec{x}}(k-1|k-1) & \text{for } i = 0 \\ \hat{\vec{x}}(k-1|k-1) + \left(\sqrt{(n+\lambda)\boldsymbol{P}_{\mathrm{xx}}(k-1|k-1)}\right)_i & \text{for } i = 1,...,n \\ \hat{\vec{x}}(k-1|k-1) - \left(\sqrt{(n+\lambda)\boldsymbol{P}_{\mathrm{xx}}(k-1|k-1)}\right)_{i-n} & \text{for } i = n+1,...,2n \end{cases} \tag{8.2}$$

where:

n	is the number of states in the state vector $[\cdot]$
$\boldsymbol{P}_{\mathrm{xx}}(k-1\|k-1)$	is the estimate of the state covariance matrix at time $k-1$
$\hat{\vec{x}}(k-1\|k-1)$	is the estimate of the state vector at time $k-1$
$\vec{\mathcal{X}}_i(k-1\|k-1)$	is the ith sigma point at time $k-1$
λ	is a parameter that determines how far the sigma points are spread from the mean; it is defined in Eq. (8.3) $[\cdot]$
$(\cdots)_i$	is the ith column of the matrix contained in the parenthesis

As the number of sigma points needed in the Kalman filter is dependent on the number of process states it is desirable to reduce the number of states to a minimum in order to save computational time. To reduce the number of states $\vec{\theta}_{\mathrm{m}}$ can easily be removed as there is a redundancy in the model as $\vec{\theta}_{\mathrm{m},i} = \vec{G}_i\vec{\theta}_i + \vec{\theta}_{\Delta,i}$.

$$\lambda = \alpha^2(n+\kappa) - n \tag{8.3}$$

where:

α and κ are tuning parameters

2. Propagate all sigma points through the process model

The non-linear process model is used to propagate each sigma point to the next time step to get prediction points. This step is illustrated by the dotted lines in Figure 8.3. The a priori predictions of the sigma points are generated using the following relation:

$$\vec{\mathcal{X}}_i(k|k-1) = f\left(\vec{\mathcal{X}}_i(k-1|k-1), \vec{u}(k-1)\right) \tag{8.4}$$

where:

$\vec{\mathcal{X}}_i(k\|k-1)$	is the ith sigma point propagated to time k
$\vec{u}(k-1)$	is the input vector at time $k-1$
$f\left(\vec{x}(k), \vec{u}(k)\right)$	is the discrete non-linear process model

3. Calculate the predicted mean of the state vector

The weighted mean of the state vector is calculated based on the a priori prediction points using Eq. (8.5) with the weight factors defined in Eq. (8.6).

$$\hat{\vec{x}}(k|k-1) = \sum_{i=0}^{2n} w_{\mathrm{m},i} \vec{\mathcal{X}}_i(k|k-1) \tag{8.5}$$

where:

$\hat{\vec{x}}(k|k-1)$ is the predicted mean of the state vector

Note that the weight factor for the 0th sigma point is not the same when calculating the mean and covariance matrix, contrary to the rest of the weight factors.

$$\begin{aligned} w_{\mathrm{m},0} &= \frac{\lambda}{n+\lambda} \\ w_{\mathrm{c},0} &= \frac{\lambda}{n+\lambda} + \left(1 - \alpha^2 + \beta\right) \\ w_{\mathrm{m},i} &= w_{\mathrm{c},i} = \frac{1}{2(n+\lambda)} \qquad \text{for } i = 1, .., 2n \end{aligned} \tag{8.6}$$

where:

$w_{\mathrm{c},i}$ is the weight of the ith sigma point when calculating the covariance of the Gaussian [·]

$w_{\mathrm{m},i}$ is the weight of the ith sigma point when the mean is calculated [·]

β is a tuning parameter which determines how much extra weight is put on the 0th sigma point [·]

4. Run each prediction point through the observation model

The predicted observation related to each of the propagated sigma points is determined using the observation model.

$$\vec{\mathcal{Z}}_i(k|k-1) = h\left(\vec{\mathcal{X}}_i(k|k-1), \vec{u}(k-1)\right) \tag{8.7}$$

where:

$h\left(\vec{x}(k), \vec{u}(k)\right)$ is the discrete non-linear observation model

$\vec{\mathcal{Z}}_i(k|k-1)$ is the predicted ith observation vector at time k corresponding to the ith sigma point

5. Calculate the predicted observation mean

The weighted mean of the predicted observations is calculated in the same manner as the mean of the state vector using the same weights.

$$\hat{\vec{z}}(k|k-1) = \sum_{i=0}^{2n} w_{\mathrm{m},i} \vec{\mathcal{Z}}_i(k|k-1) \tag{8.8}$$

where:

$\hat{\vec{z}}(k|k-1)$ is the predicted mean of the observation vector

6. Compute the predicted state and innovation covariance matrices and the cross correlation matrix

The covariance of the transformed sigma points represents the a priori covariance of the state vector and is calculated as follows:

$$\boldsymbol{P}_{\mathrm{xx}}(k|k-1) = \sum_{i=0}^{2n} w_{\mathrm{c},i} \left(\vec{\mathcal{X}}_i(k|k-1) - \hat{\vec{x}}(k|k-1)\right) \left(\vec{\mathcal{X}}_i(k|k-1) - \hat{\vec{x}}(k|k-1)\right)^T + \boldsymbol{Q} \tag{8.9}$$

where:

$\boldsymbol{P}_{\mathrm{xx}}(k|k-1)$ is the predicted state covariance matrix at time k
$\boldsymbol{Q}$ is the process noise covariance matrix

To compute the new Kalman gain the covariance of the innovation is needed along with the cross correlation between the sigma points and the observation points. The covariance of the innovation is calculated using Eq. (8.10) while the cross correlation is determined using Eq. (8.11).

$$\boldsymbol{P}_{\mathrm{zz}}(k|k-1) = \sum_{i=0}^{2n} w_{\mathrm{c},i} \left(\vec{\mathcal{Z}}_i(k|k-1) - \hat{\vec{z}}(k|k-1)\right) \left(\vec{\mathcal{Z}}_i(k|k-1) - \hat{\vec{z}}(k|k-1)\right)^T + \boldsymbol{R} \tag{8.10}$$

$$\boldsymbol{P}_{\mathrm{xz}}(k|k-1) = \sum_{i=0}^{2n} w_{\mathrm{c},i} \left(\vec{\mathcal{X}}_i(k|k-1) - \hat{\vec{x}}(k|k-1)\right) \left(\vec{\mathcal{Z}}_i(k|k-1) - \hat{\vec{z}}(k|k-1)\right)^T \tag{8.11}$$

where:

$\boldsymbol{P}_{\mathrm{xz}}(k|k-1)$ is the predicted cross correlation matrix between the states and the observations at time k
$\boldsymbol{P}_{\mathrm{zz}}(k|k-1)$ is the predicted innovation covariance matrix at time k
$\boldsymbol{R}$ is the observation noise covariance matrix

8.3.2 Update Steps

The purpose of the update steps is to use the observation of the system to update the a priori predictions determined during the prediction steps.

7. Determine the new Kalman gain matrix

The Kalman gain is determined from the innovation covariance and the cross correlation matrices determined in the prediction steps.

$$\boldsymbol{K}(k) = \boldsymbol{P}_{\mathrm{xz}}(k|k-1)\boldsymbol{P}_{\mathrm{zz}}^{-1}(k|k-1) \tag{8.12}$$

where:

$\boldsymbol{K}(k)$ is the Kalman gain at time k

8. Calculate the state vector estimate

The estimate of the state vector is determined from the a priori mean of the state vector, the Kalman gain, and the observation innovation, using the following equation:

$$\hat{\vec{x}}(k|k) = \hat{\vec{x}}(k|k-1) + \boldsymbol{K}(k) \left(\vec{z}(k) - \hat{\vec{z}}(k|k-1)\right) \tag{8.13}$$

where:

$\hat{\vec{x}}(k|k)$ is the estimate of the state vector at time k
$\vec{z}(k)$ is the observation vector at time k

9. Calculate the state covariance estimate

The state covariance matrix represents the certainty of the current state vector estimate and is determined using Eq. (8.14).

$$\boldsymbol{P}_{\mathrm{xx}}(k|k) = \boldsymbol{P}_{\mathrm{xx}}(k|k-1) - \boldsymbol{K}(k)\boldsymbol{P}_{\mathrm{zz}}(k|k-1)\left(\boldsymbol{K}(k)\right)^T \tag{8.14}$$

where:

$\boldsymbol{P}_{\mathrm{xx}}(k|k)$ is the estimate of the state covariance matrix at time k

8.3.3 Implementation of the Unscented Kalman Estimator

In the implementation of the unscented Kalman estimator it was discovered that handling the friction in the estimator in the same manner as described in Subsection 5.3.2 on page 45 did not perform satisfactory. The problem was that the estimator was not able to estimate a velocity of zero of a joint. This problem occurs because of the procedure where the sigma points are propagated through the model and the mean is determined. When the mean is calculated it will have a value different from zero if just a single sigma point results in a non-zero value. The problem is illustrated in Figure 8.4, where the estimate should be zero most of the time. In fact the only reason why the joint ever starts to move is because the estimate is non-zero which makes the controller compensate for it, and at some point the joint starts to move a bit.

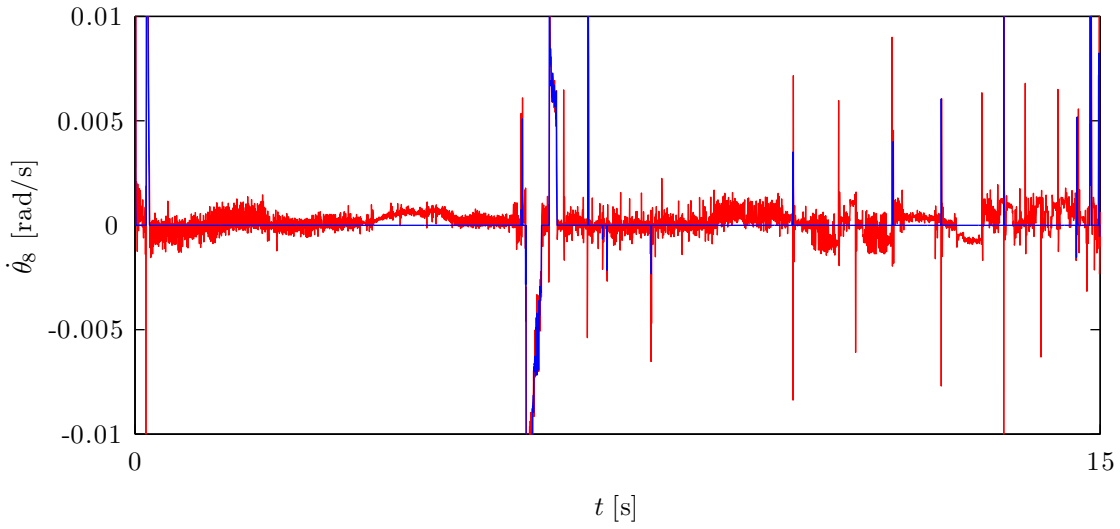


Figure 8.4: *Plot of the estimators ability to estimate zero before the change in implementation. The blue line is the correct value and the red line is the estimated value.*

A non-zero value of the angular velocity is very unfortunate as the previous value for the angular velocity is used to detect if the joint is running and the friction should be set accordingly. To handle the problem a small dead zone has been utilized in the estimator so that when the angular velocity is below a small epsilon value in the previous sample, then the joint is assumed to be non-moving.

When applying this method the estimator has improved in being able to estimate zero velocity. In the same situation as shown in Figure 8.4 the changed estimator is able to estimate a constant zero velocity.

8.3.4 Tuning of the Unscented Kalman Estimator

The tuning of an unscented Kalman estimator involves a number of parameters. First of all the sensor and process noise should be set to suitable values.

The sensor noise is often known as it is given by the datasheet of the sensors, or a good guess can be given based on measurements from the sensors. For the sensors in **AAU-BOT1** there is practically no noise as the measurements are given by digital incremental encoders which are unaffected by noise from the surroundings. The largest source of noise in the sensors is the result of the quantization; however, the quantization steps are so small that even this error is very small. For the angle measurements on the motor the quantization steps are $\frac{2\pi}{2048} \approx 0.0031$ rad and for the angular velocity measurements on the motor it is $\frac{2\pi}{60} \approx 0.1047$ rad/s. When looking at these numbers it is clear that the noise on the angle measurements is significantly smaller than on the angular velocities. As the sensors are unaffected by noise from the surroundings the assumption that the noise of each is independent of the others is easily justified, hence a diagonal matrix is used as covariance matrix.

For the process noise no values are known before hand; however, it is assumed that the model of the angles is good compared to the model of the angular velocities. As all joints are modeled equally

and should behave similar to each other the same value is used in all joints. To ease the tuning it is assumed that the process noise of all states is independent, resulting in a diagonal matrix as covariance matrix.

Also some parameters directly related to the sigma points of the unscented Kalman filter needs to be set properly. The parameters α and κ are parameters which determine how far from the mean value the sigma points should be located. α is a high order value while κ is for fine tuning of the spreading of the sigma points. β determines how much additional weight that should be put on the mean value in the calculation of the covariance of the transformed sigma points.

The resulting parameters of the tuning of the Kalman estimator are given below:

$$\boldsymbol{Q} = \text{diag}\left(\begin{array}{cccc} \mathbf{1}_{1\text{x}17} \cdot 10^{-6} & \mathbf{1}_{1\text{x}17} \cdot 10^{-6} & \mathbf{1}_{1\text{x}17} \cdot 5 \cdot 10^{-7} & \mathbf{1}_{1\text{x}17} \cdot 5 \cdot 10^{-5} \end{array} \right) \tag{8.15}$$

$$\boldsymbol{R} = \text{diag}\left(\begin{array}{cc} \mathbf{1}_{1\text{x}17} \cdot 10^{-5} & \mathbf{1}_{1\text{x}17} \cdot 5 \cdot 10^{-4} \end{array} \right) \tag{8.16}$$

$$\alpha = 10^{-2} \qquad \beta = 2 \qquad \kappa = 10^{-5} \tag{8.17}$$

Additionally, the initial state covariance matrix must be set. This matrix determines how much the initial values of the states are trusted. As a measurement can be used as initial value of most of the states (except for θ_Δ) the values are trusted a lot, hence a diagonal matrix like below is used:

$$\boldsymbol{P}_{\text{xx}}(0) = \text{diag}\left(\mathbf{1}_{1\text{x}68} \cdot 10^{-10}\right) \tag{8.18}$$

8.4 Evaluation of Estimator

The performance of the designed unscented Kalman estimator is evaluated by a number of tests performed on the simulation model. It is assumed that the worst situation for the estimator is a change in the stiffness coefficient of the gears as this means that the model between the motor shaft angle and the joint angle is changed. The reason this is assumed to be the worst case is that the joint angles and angular velocities are not measured but can only be estimated using the model.

In order to simulate the absolute worst case the stiffness coefficient is multiplied and divided by a factor of 10, in two tests, respectively. To show the performance of the estimator the correct joint angles and the estimated joint angles for the situation where the stiffness coefficient is divided by 10 is shown in Figure 8.5. Figure 8.6 shows the correct and the estimated angular velocities for the same test.

As it can be seen from the figures the estimate is following the correct value very close at all times. It is clear that the estimates of the angular velocities of the arms do not follow right at the beginning, but as the arm gets moving the estimate gets correct. This initial error is due to the initialization of the filter, combined with the dead zone in the friction model. Because the estimates are so close to the correct values it is hard to see the difference, therefore the estimate error is shown for the angles in Figure 8.7 on page 98.

In all tests performed, including the one where the stiffness coefficient is multiplied by 10, the estimator performed similarly to the test shown in the figures.

An unscented Kalman estimator for **AAU-BOT1** has been designed and tested to estimate the unmeasured joint angles and velocities. This completes the design of the entire control system, containing a posture controller, a balance controller, an unscented Kalman estimator, and a suitable trajectory. The next chapter presents the test of the control system, performed by simulations.

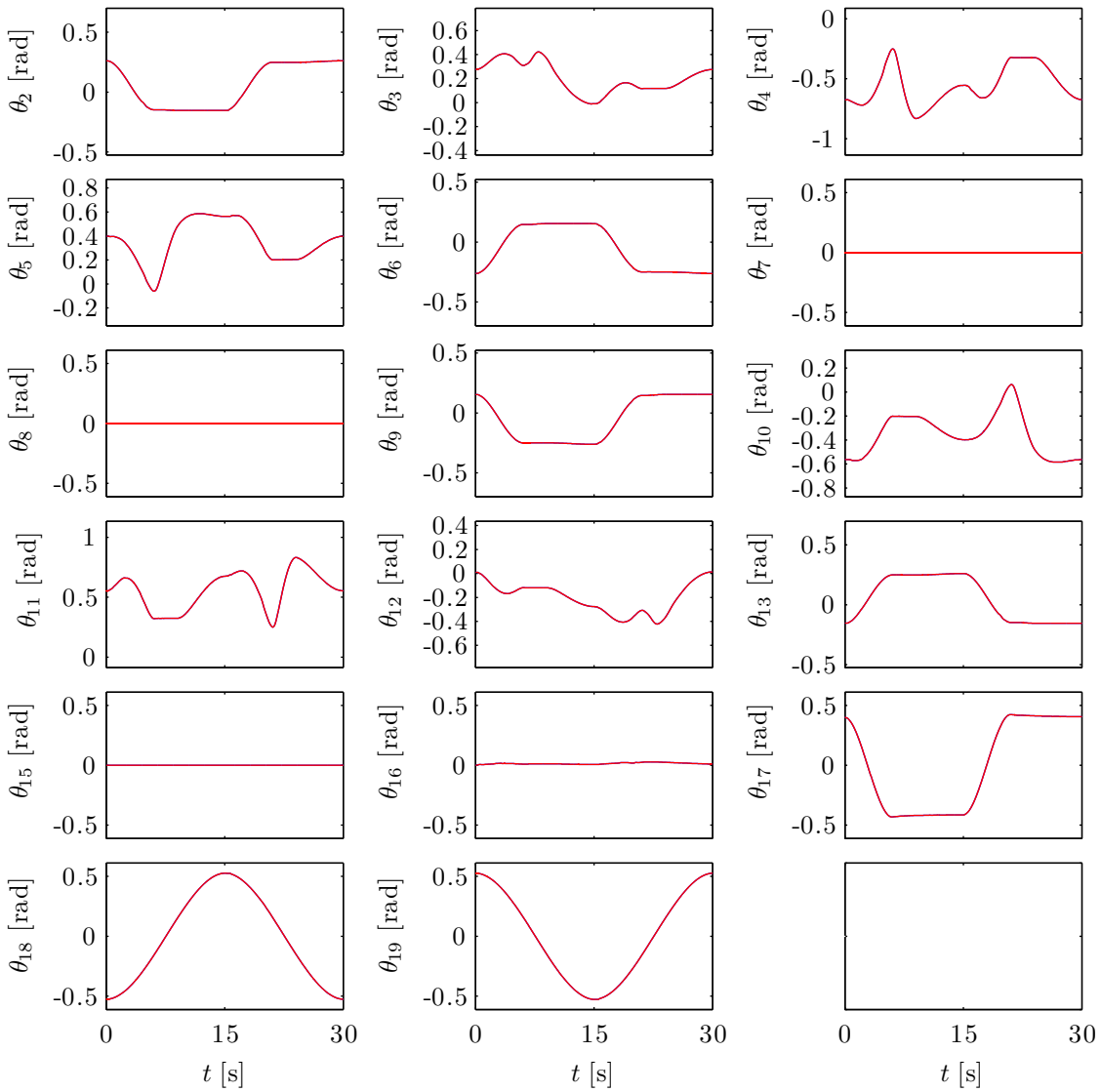


Figure 8.5: *Evaluation of the performance of the unscented Kalman filter. The plots show the correct joint angles as blue lines and the estimated angles as red lines. Note that the scaling of all subplots is equal even though they are offset differently.*

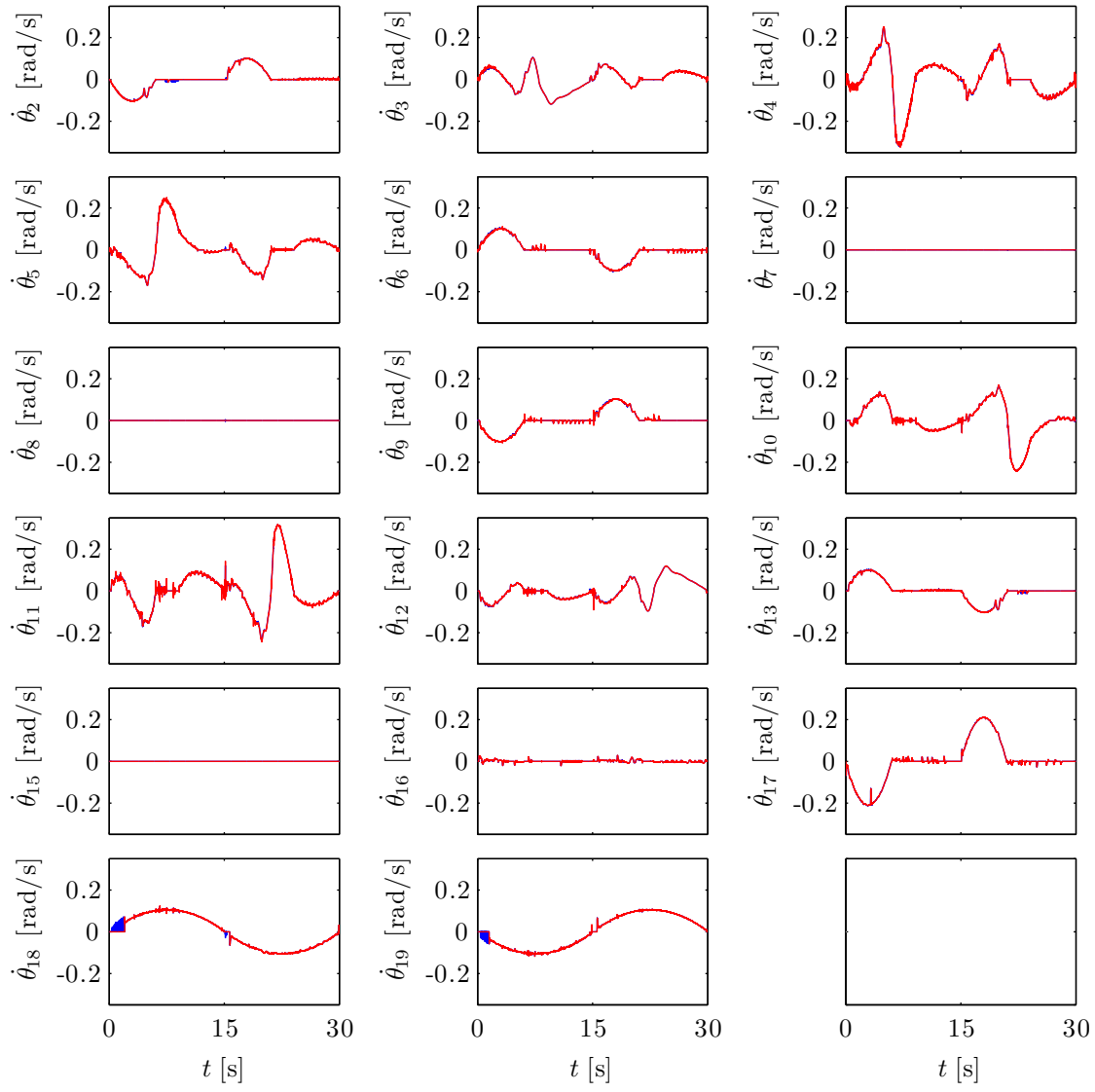


Figure 8.6: *Evaluation of the performance of the unscented Kalman filter. The plots show the correct joint angular velocities as blue lines and the estimated angular velocities as red lines.*

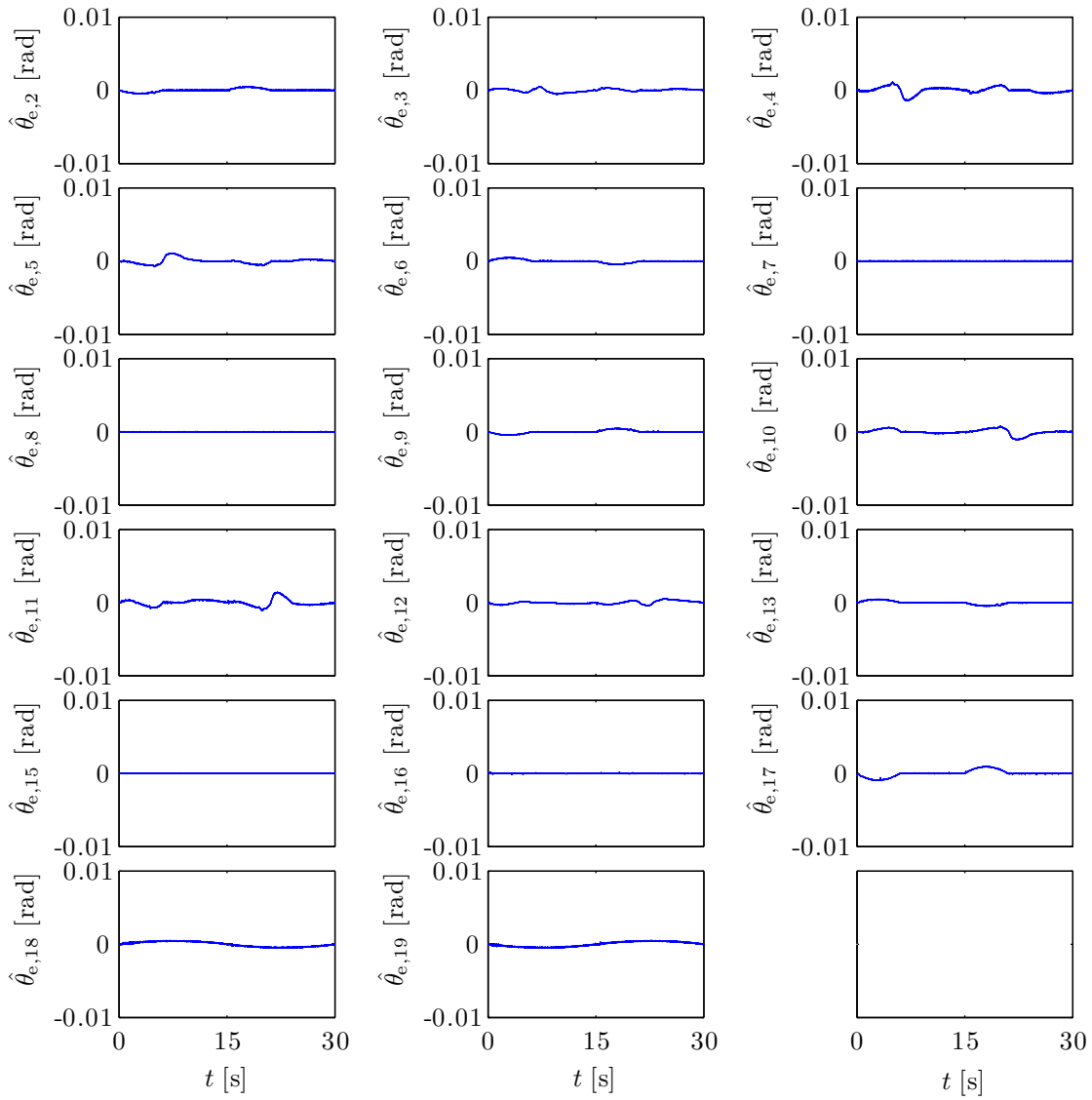


Figure 8.7: *Evaluation of the performance of the unscented Kalman filter. The plots shows the estimation error of the angle in all joints.*

IV

Control System Test and Conclusion

9 System Test

So far a complete model has been set up with the purpose of simulating the behavior of the robot and to use the model in the estimator and control design. A complete trajectory has been designed to make the robot perform static gait. To track the trajectory a Kalman estimator has been designed to estimate the unmeasured joint angles and velocities used in the controller based on the measured motor angles and angular velocities. The controller for tracking the joint angles has been designed utilizing feedback linearization and decoupling of the system.

In this chapter the complete control system is tested to study how it performs in preparation for making the robot accomplish static gait. The test is only carried out in the simulation of the robot, since it was not possible to test the control system on the actual **AAU-BOT1** before handing in this master's thesis. Conduction of the tests will however be attempted on the real **AAU-BOT1** before the exam. In the following section the specifications of the conducted tests are described, while the results are presented afterwards.

Contents

9.1 Test Specification

The control system is tested in different scenarios to simulate differences between the model used in the control system and the actual **AAU-BOT1**. In this way the robustness of the control system is in some degree tested. This is obtained by changing some parameters in the simulation model and then studying how the control system responds to the changes. The simulation model is modified in three different ways, involving the most significant model properties. The three modifications are listed and described in the following.

- **Changing the mass distribution**
  The mass distribution of the robot is one of the most important things for the dynamic behavior of the system and therefore it is important to test how the control system responds to changes in the distribution. In the control system the model is used to decouple the system, and by changing the mass distribution, the cross couplings changes. The mass of the robot is extracted from the technical drawings used to build the robot, as described in Section 5.3 on page 42, but this gives a difference in the total mass compared to the actual robot. This difference has been distributed to the legs and torso weighted by their mass. But this difference also indicates an uncertainty in the mass distribution and therefore the system is tested with this mass difference distributed in other ways. This test is made in two situations, first with the total mass difference placed entirely in the torso and then with the difference placed entirely in the legs. The total mass difference is approx 6 kg and when compared to the original model it means that the first test involves moving about 4 kg from the legs to the torso, while in the second test about 2 kg is moved from the torso to the legs

- **Changing parameters in the gear model**
  Another important part of the model is the gear model which is used in the Kalman estimator to estimate the joint angles and angular velocities. Since there are no measurements of the output of the gears it is important that the model of the gears is accurate and that the estimator is able to cope with uncertainties in it. Hence, tests are performed where the stiffness of the gears is varied in the model. Two tests are made with the stiffness 10 times

larger and smaller, respectively. The result of these tests are presented and evaluated in Section 8.4 on page 95.

- **Changing the Coulomb friction**
  The robot is affected by high friction in the joints which is why a friction compensation has been added as part of the control system. Therefore, a test is performed to evaluate the importance of the accuracy of the friction coefficients. The friction coefficients has been estimated for a single joint as described in Appendix C on page 124, but the friction can vary due to the load and be different in each joint. The most significant part of the friction is the Coulomb friction due to the high value and since the compensation for it is of great importance for the tracking ability for the control system. The Coulomb friction is changed in two tests, with 10 % up and down, respectively.

9.2 Test Results

In this section the results of the different tests are presented and evaluated. Firstly, the result of the simulation of the ideal case, meaning no parameters in the model has been changed, is presented for comparison. Afterwards, the results of the tests with changed parameters are presented. All the results of the system test can also be seen with additional plots on the attached DVD [`System_Test/`].

All tests are made for a complete gait cycle starting in double support with GCoM located on the right foot. Then at time 6 s the weight is moved to the left foot and the right foot is lifted to start the swing phase. At time 15 s the right foot makes contact with the ground and the second double support phase starts, where the weight starts to be moved to the right foot. Then at time 21 s the left foot is lifted to start the swing phase and at time 30 s it makes contact with the ground and the gait cycle is complete. The trajectory is illustrated by the plots in Section 7.4 on page 82.

9.2.1 Ideal Case

The simulation results for the ideal case are presented in this section. In Figure 9.1 the joint angle reference and the obtained joint angle is shown for each joint and it is clear that the robot follows the trajectory. To clarify the tracking ability the error between the reference and the obtained joint angles are shown in Figure 9.2. The errors are generally low for most of the joints through out the gait cycle but the pitch joints of each leg have larger errors in a short period of time. These larger errors are in connection with the lift off of the foot, i.e. around time 6 s for the right leg (θ_3, θ_4, and θ_5), and around time 21 s for the left leg (θ_{10}, θ_{11}, and θ_{12}). The larger errors are due to the weighting of the two models in the control used in double support described in Subsection 6.3.2 on page 67, where each model is weighted completely when the GCoM is located inside the given foot. This means that in the end of the double support phase the control system only uses one of the models even though both feet are in contact with the ground, which has shown to be a better compromise, as described in Subsection 6.3.2 on page 67. Overall the error is never lager than 2° and the joint angle tracking ability is therefore assumed to be sufficient. It should also be noted that the points where tracking error is largest is not at any time where the physical limitations are close to being violated.

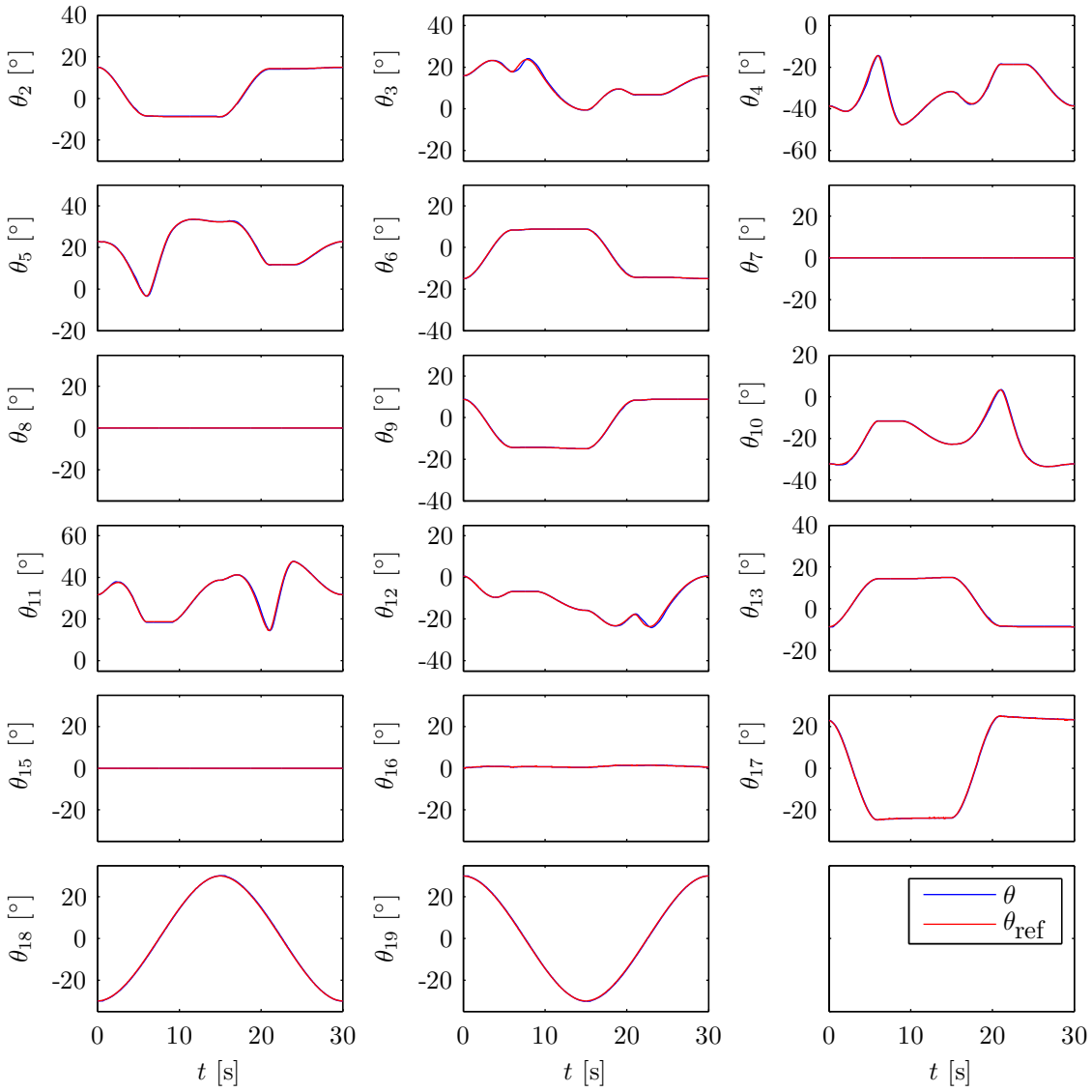


Figure 9.1: *Simulation results of the ideal case, here shown by the joint angle reference illustrated by red and the obtained joint angle illustrated by blue for each joint.*

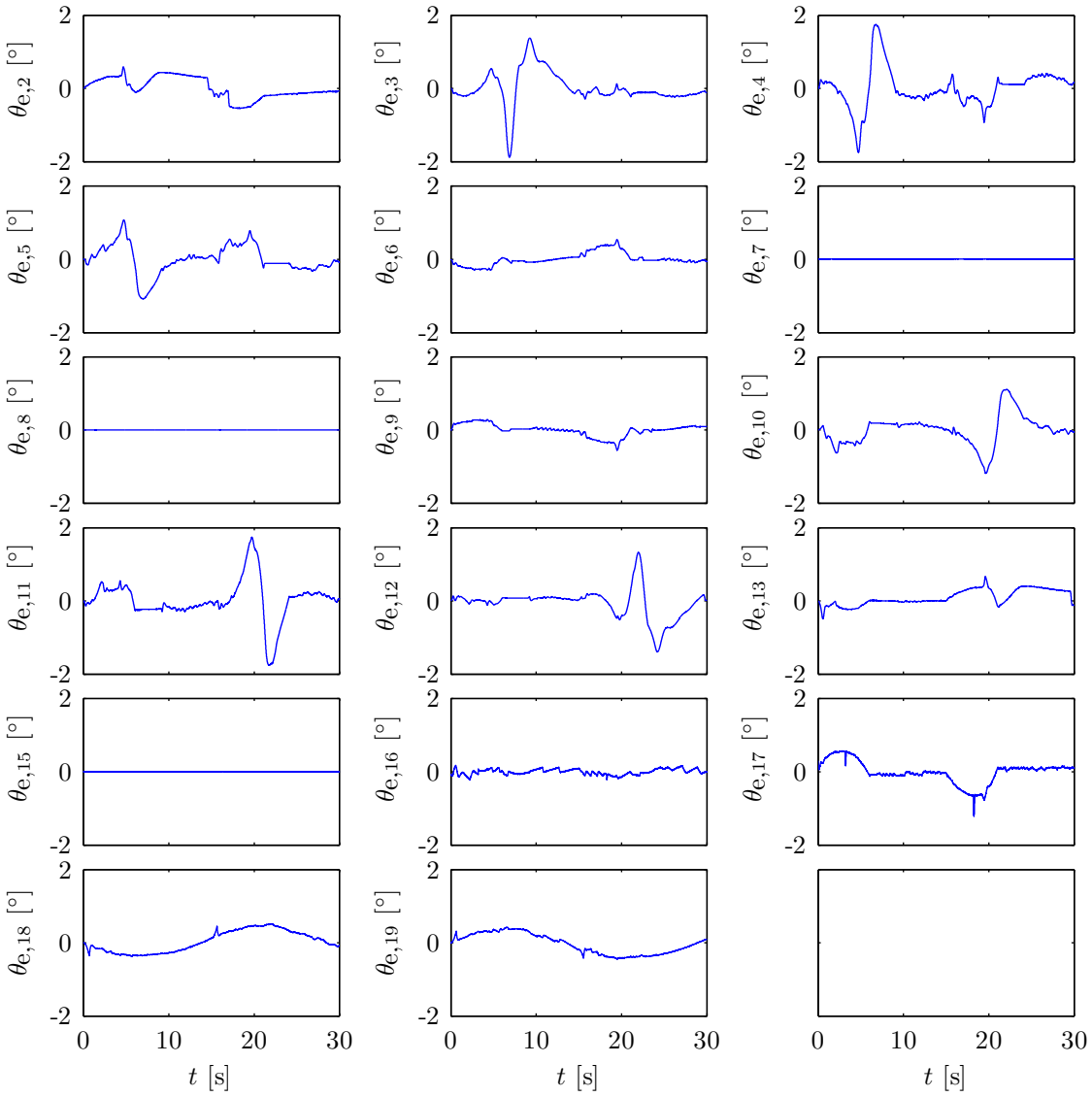


Figure 9.2: *Simulation results for the ideal case shown by the error between the joint angle reference and the obtained joint angle for each joint.*

In Figure 9.3 the result of the balance during the gait cycle for the test in the ideal case is shown. Note that due to the undesired feature with the calculated ZMP, described in Section 6.4 on page 68, the GCoM is used in the balance control in the simulation. Due to the well tracking of the joint angles the robot is also able to keep the balance without large compensation angles. Note that the jumps in the compensation angles, seen around 3 s and 18 s, are due to changes between the two single support models in the control loop. These jumps are expected as the feet are assumed to be flat on the ground at the time of the change, but due to the springs in the feet their angles will always be slightly different, which becomes evident as small jumps in the compensation angle.

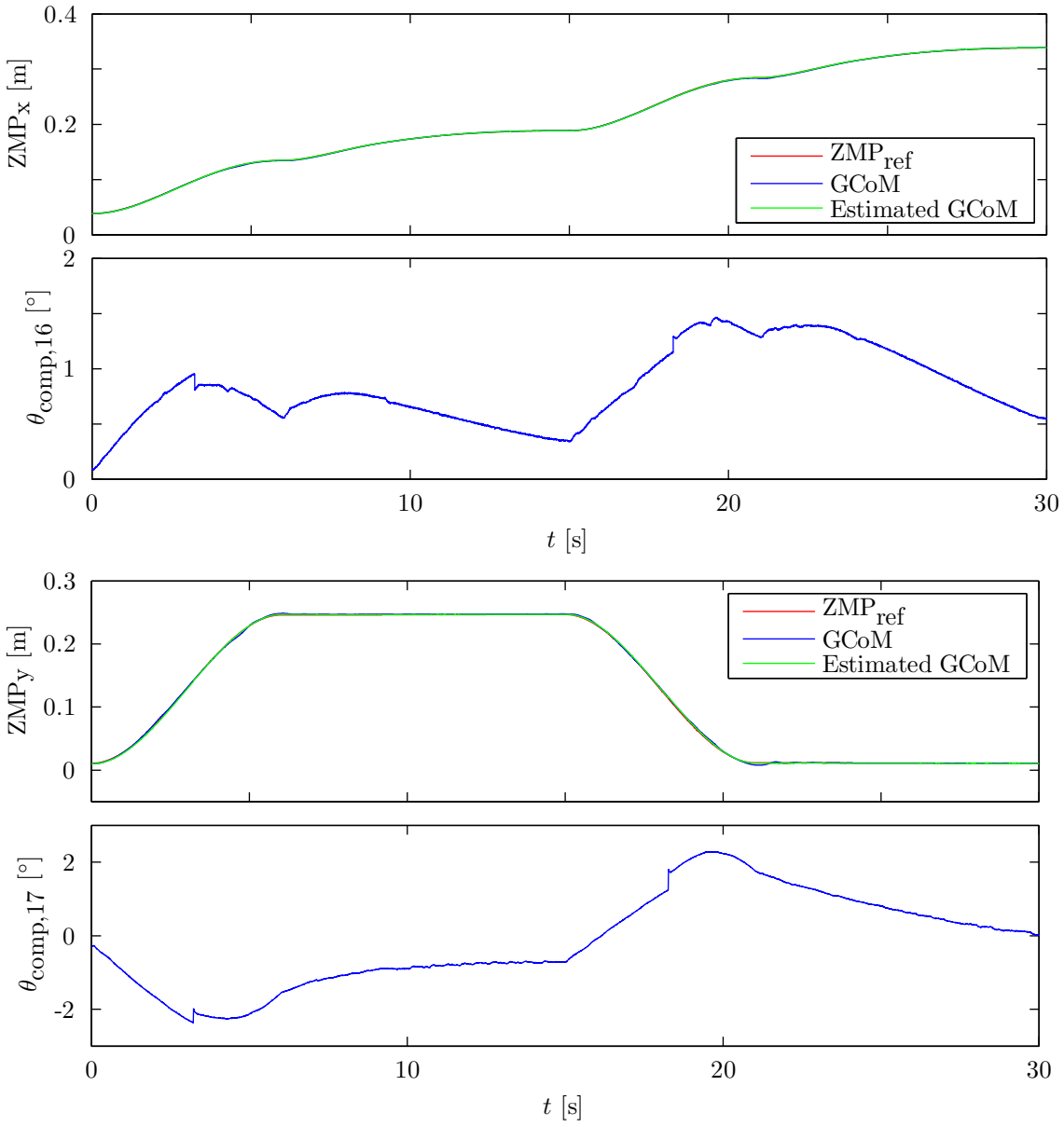


Figure 9.3: *Result of the balance in the test of the ideal case. The two top plots shows the x-coordinate of the ZMP and GCoM together with the compensation angle for it,* $\theta_{\text{comp},16}$*. The two bottom plots is the y-coordinate and the compensation angle for it,* $\theta_{\text{comp},17}$*.*

In Figure 9.4 the motor current and the angular velocities are shown for 4 joints in the right leg. The sudden steps that occur in the motor current for joint 2 and 4 around time 6 s and 15 s are caused by change in the direction of the velocities, causing the friction compensation also to change sign. The high fluctuations on the motor current at the end of the gait cycle are caused by fluctuations on the estimated angular velocity for the ankle of the standing foot, i.e. joint 2. The reason why there is no fluctuations on the motor current for joint 2 in the time between time 6 s and 15 s, is because the ankle is free from the ground and the inertia is therefore low, resulting in a small gain through $\boldsymbol{M}$ in the controller giving smaller fluctuations on the estimated states and the motor current. Even though the hip yaw joint, i.e. joint 7, is not moving, the motor current varies which is due to changing load in the joint.

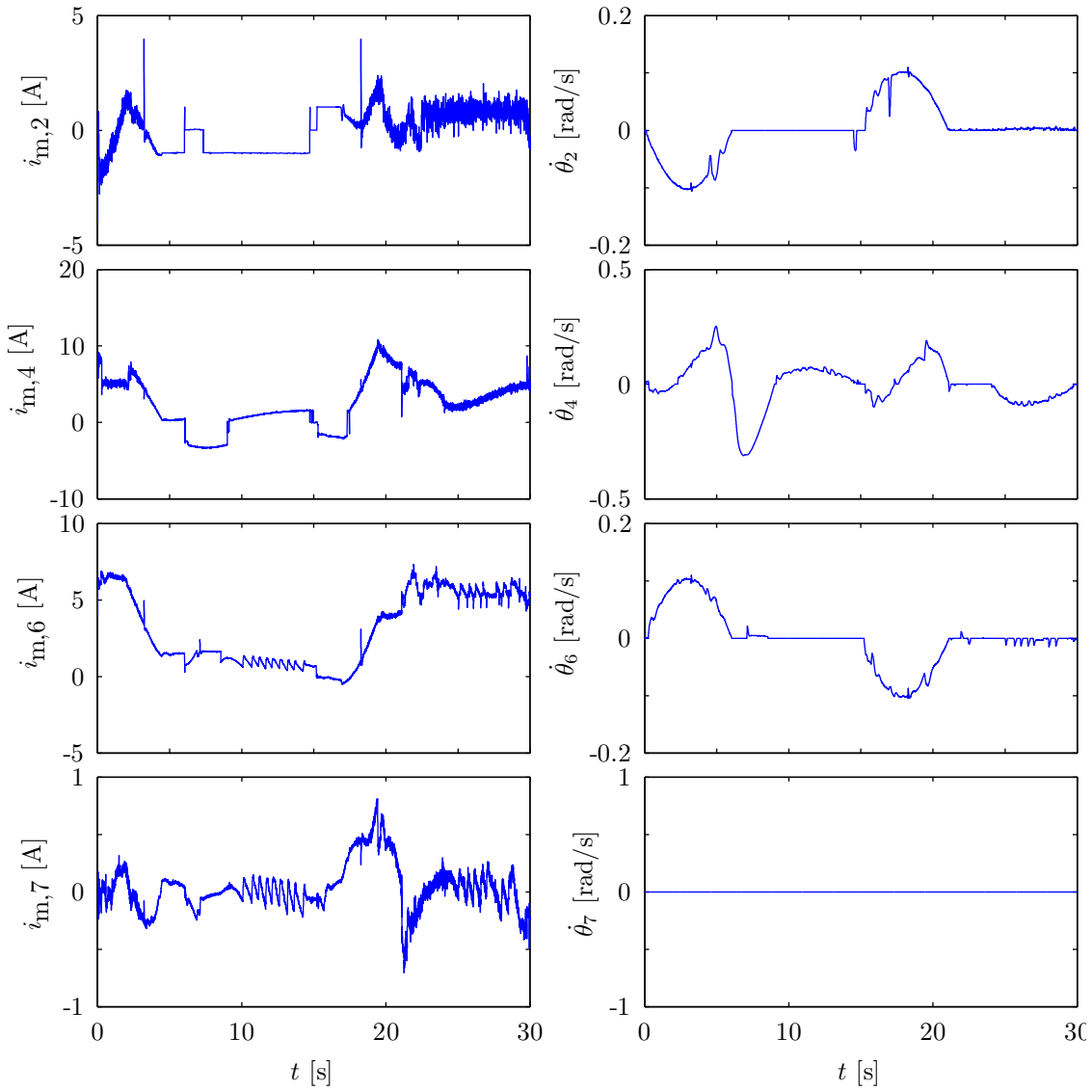


Figure 9.4: *The motor current compared to the angular velocities for 4 joints in the right leg, during the gait cycle in the ideal case.*

9.2.2 Changing the Mass Distribution

The test with the redistributed mass has the purpose of testing the ability of the control system to decouple the system. In the two tests, where first 4 kg is moved from the legs to the torso and afterwards 2 kg is moved from the torso to legs, there was an insignificant difference in the tracking ability compared to the ideal case. This means that the expected difference in the mass distribution that may occur is not affecting the control performance significantly.

The changed mass distribution also affects the balance of the robot which can be seen in Figure 9.5, where 4 kg has been moved from the legs to the torso. Due to the higher mass in the torso the y-coordinate of the GCoM moves further. Since the balance controller uses the estimated GCoM, calculated from the control model with the estimated angles, it is not able to compensate the balance. This is an error made in the simulations since on the actual **AAU-BOT1** the FTS provides the actual balance point and thereby the balance controller should be able keep the position of the GCoM.

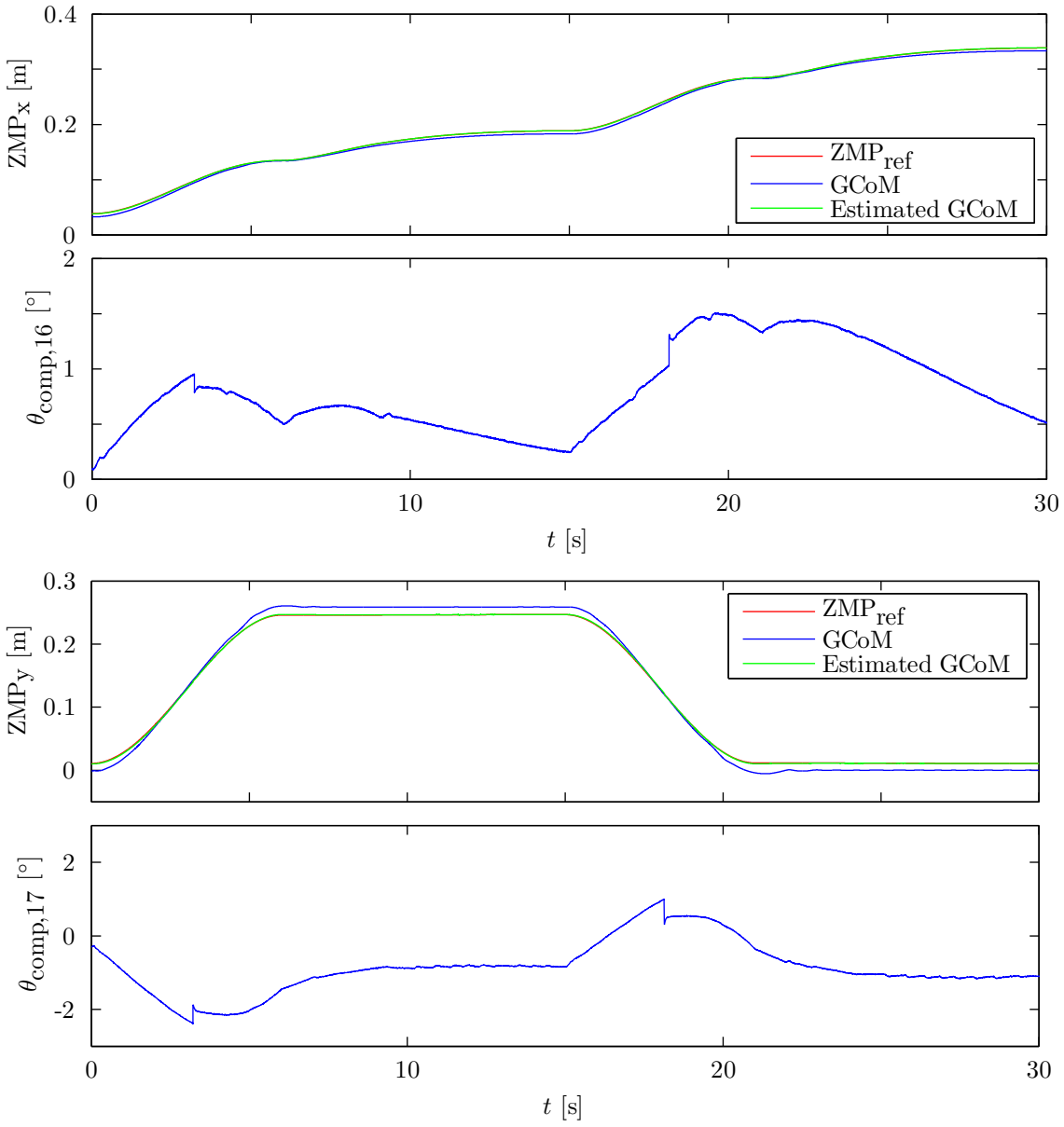


Figure 9.5: *Result of the balance in the test where 4 kg has been moved from the legs to the torso. The two top plots show the x-coordinate of the ZMP and GCoM together with the compensation angle, $\theta_{\text{comp},16}$. The two bottom plots are the y-coordinate and the compensation angle, $\theta_{\text{comp},16}$.*

In Figure 9.6 the balance for the test where 2 kg is moved from the torso to the legs is shown, as for the previous test the GCoM changes due the changed mass. Since the mass change in this test is smaller the variation on the GCoM is also smaller.

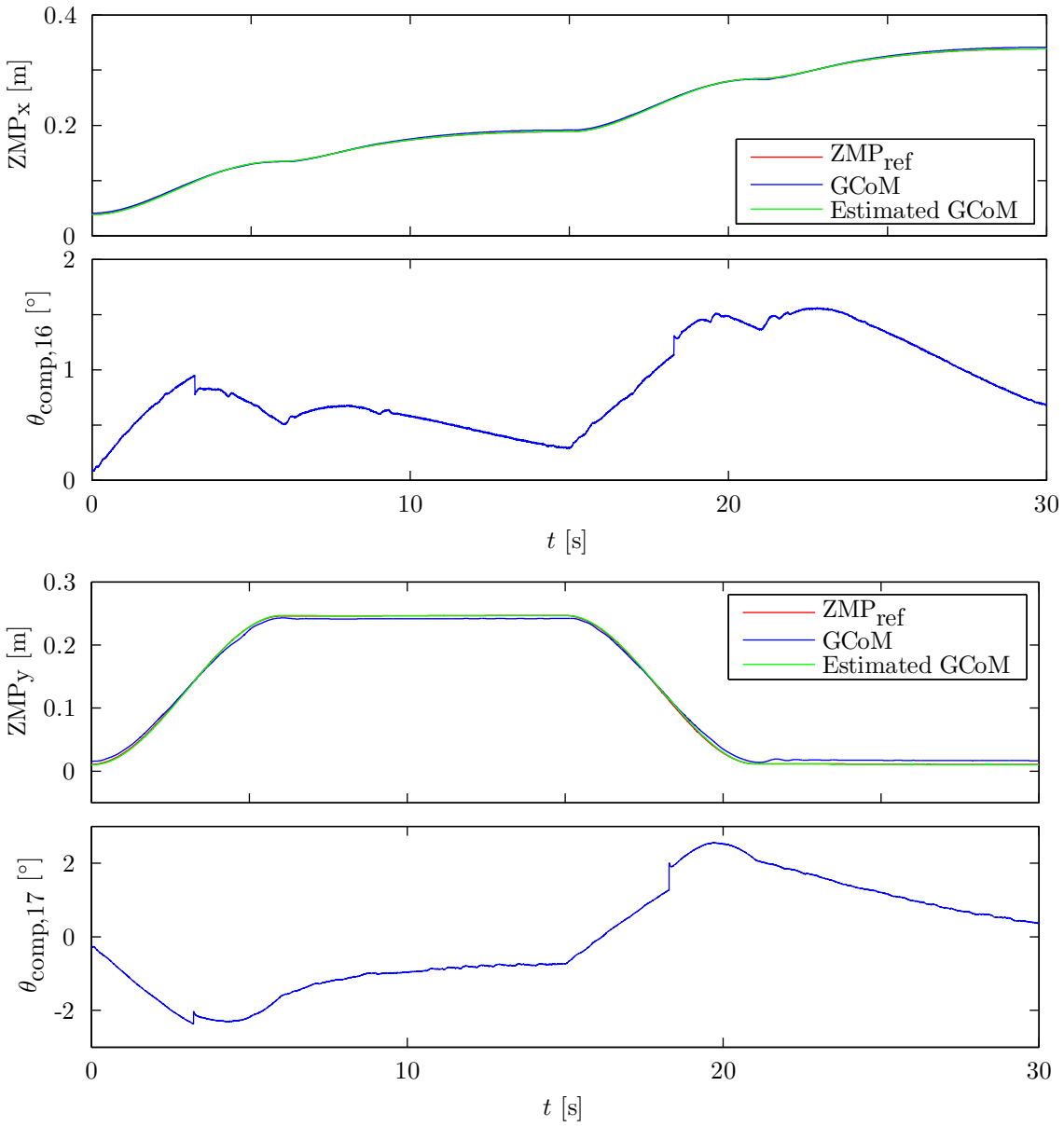


Figure 9.6: *Result of the balance in the test where 2 kg is moved from the torso to the legs. The two top plots show the x-coordinate of the ZMP and GCoM together with the compensation angle,* $\theta_{\text{comp},16}$*. The two bottom plots are they-coordinate and the compensation angle,* $\theta_{\text{comp},16}$*.*

9.2.3 Changing the Coulomb friction

To test the effect of incorrect friction compensation the coulomb friction has been changes 10 % up and down. For the test with 10 % reduction of the coulomb friction, Figure 9.7 shows the error between the joint angle reference and the obtained angle. The reduction of the coulomb friction means that the friction compensation from the controller becomes too large and is enough to make some joints move. This result in fluctuations on the error as shown since the controller in some situations is unable to keep the joint angles on a steady reference due to the high friction compensation making the joints move.

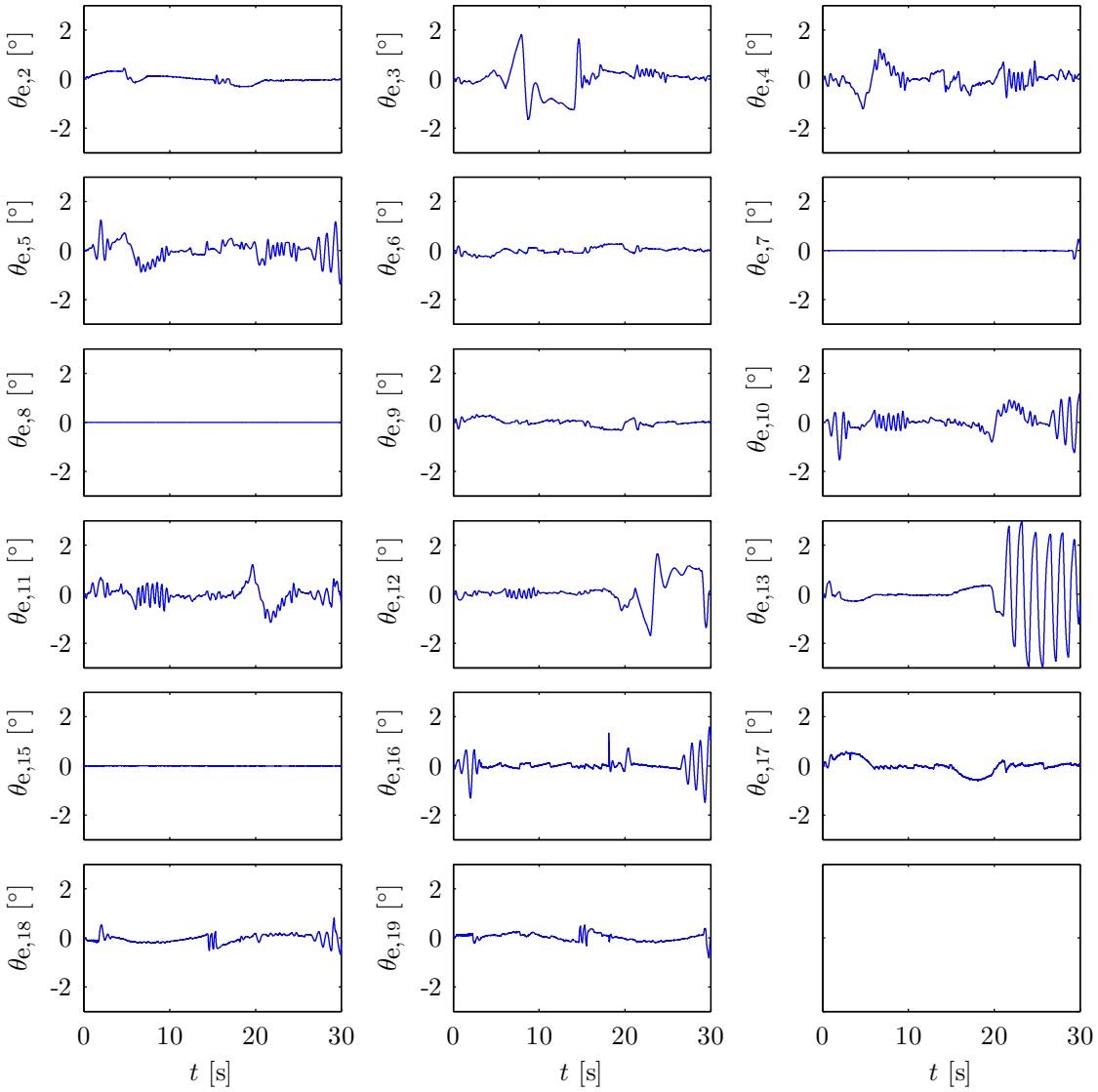


Figure 9.7: *Simulation results of the error between the joint angle reference and the obtained angle in the test where the Coulomb friction is reduced by 10 %.*

In Figure 9.8 the joint angle references and the obtained angles are showed where the coulomb friction is increased by 10 %. The largest differences compared to the ideal case is the two ankle pitch joints, i.e. joint 3 and 12, where the higher friction means that the joints starts to move later. The reason why it is primarily the ankle pitch that is affected is because of the low inertia in the swinging phase and thereby low gain through $\boldsymbol{M}$ in the controller. The results show that it is preferable to have too low friction compensation than too high.

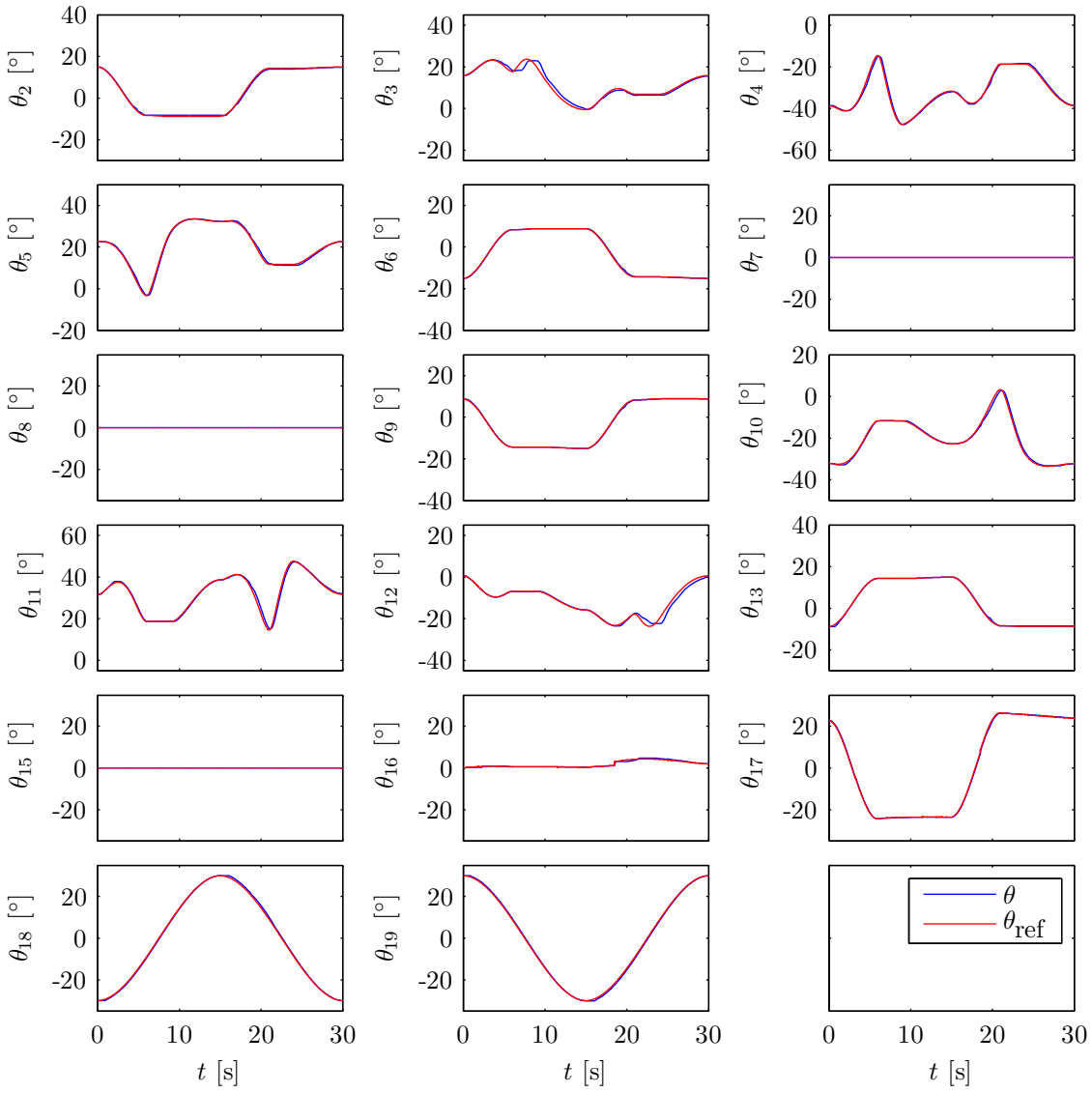


Figure 9.8: *Simulation results of the obtained joint angles compared to the references in the test where the Coulomb friction is increased by 10 %.*

The test results show by simulations, that the designed controller and the Kalman estimator are able to control the robot to perform static gait. Even with significant model differences between the simulation model and the model used in the controller and the Kalman estimator, the robot was still able to perform static gait. This indicates that the control system is able to make the actual **AAU-BOT1** perform static gait which has not been tested yet.

10 Epilogue

In this chapter an epilogue of this master's thesis is given consisting of a summary of the main parts that has been studied in the project and a conclusion for the thesis.

Contents

10.1 Thesis Summary

The summary of this master's thesis is given to provide an overview of the main achievements of the project, including the work conducted on the hardware interface, software platform, model of the robot, trajectory generation, estimator design, control system design, and the test results.

10.1.1 Finalizing the Hardware Interface

At the beginning of this project the new on-board computer had just been assembled and did not have any connections to the robot or have any software installed.

The reason why a new computer had been bought was that the old on-board computer, which was a standard laptop computer, did not have other connection possibilities than USB-ports to connect to the robot. This meant that a number of CAN-USB dongles and Serial-USB dongles had to be utilized, however, it was discovered by [Jensen et al., 2008] that this was not an optimal solution because of the delays that are present in standard USB drivers.

The new computer is a standard desktop computer which is equipped with a CAN extension card and a Serial extension card in the PCI-slots. This set-up is better suited for real time control of **AAU-BOT1** as it does not impose undesirable delays on the communication. The superiority of the new system also becomes evident when looking at the throughput of the network. With the previous system there was an error rate of 13.6 % at a sampling frequency of 250 Hz [Jensen et al., 2008, p. 221], where as with the new system it is achieved to lower the error rate to 0.0017 % at the same sampling frequency.

Furthermore, on the hardware side, the interface to the IMU has been made and the connections between the computer and the FTS amplifiers have been re-wired as consequence of the new connection possibilities of the new computer. Lastly, a new solid-state hard disk has been bought, which is sufficiently fast, also when developing software, contrary to the old solid-state hard disk [Jensen et al., 2008, p. 46].

10.1.2 Developing Software Platform

To have a suitable operating system with real time capabilities it was decided to utilize a Linux installation combined with RTAI. A real time driver was only provided for one of the PCI extension cards, namely the CAN interface card. For the other extension card only a standard Linux driver was provided which meant that a new driver had to be developed.

In order to give a basis for the controller software a universal kernel module has been developed, which has an easy to use interface where messages can be transmitted to the hardware, utilizing

the interface cards and the drivers for them. Also, the kernel module automatically saves the last received measurement from all sensors making it easy to get the most resent measurement from the control software. During the development of the kernel module, and during development of the control software, a console interface with a simple user interface has been developed and used. This operator interaction software is capable of performing all necessary supervision tasks related to the hardware and the kernel module.

Furthermore, an interface between the kernel module and `MATLAB Simulink` has been created with use of the `RTAI Simulink` Real Time Workshop framework. This makes it possible to implement controllers directly in Simulink and then compile them to compilable C-code which ends up with an executable program with hard real time capabilities.

10.1.3 Setting up a Model of the System

A complete model of the robot is set up in order to simulate the behavior of the robot and design a control system. The model consist of a kinematic part, for mapping the joint angles into Cartesian coordinates, and a dynamic part, that is based on Lagrange-d'Alembert, to describe the motion of the robot. The model set up by the previous project group, [Jensen et al., 2008], did not include a complete description of the robot in double support; therefore, in this project the simulation model has been developed to include a ground contact model based on a spring damper system to model the robot in all the phases of a gait cycle.

During some pre-tests made on the robot it was discovered that the flexibility in the gear systems connecting the motors to the joints could result in unstable behavior of the control system. The gear systems are therefore modeled as two-mass systems and included in the simulation model to give the model more realistic characteristics. Then, the gear model is used in the Kalman estimator to be able to estimate the unmeasured joint angles and velocities.

To design a static trajectory for the robot an inverse kinematic model is set up. This model converts Cartesian coordinates, describing the position of the robot parts, into joint angles.

From the simulation model two simpler models are derived, describing the robot in single support right and single support left, respectively. These models are used in the design of both the controller and the Kalman estimator.

Parameters as the mass and inertia of the robot together with kinematic geometry are extracted from the technical drawings of the robot, which have been updated to correspond to the physical robot. Friction coefficients and parameters for the gear model are estimated based on measurements made at a joint on the robot.

10.1.4 Designing a Trajectory for the Robot

A trajectory suitable for static gait has been found. This trajectory is created in a way so that all restrictions given by the physical **AAU-BOT1** are met, which have not been achieved by the previous project groups due to missing knowledge of the real **AAU-BOT1**. The trajectory is generated by considering the Ground Center of Mass (GCoM) in multiple simulated trajectory variations and choosing the trajectory with the highest stability, i.e. the distance from the GCoM to the edge of the support area is maximized.

For this project the trajectory has been generated offline, but for future projects the same method can be utilized to generate a set of trajectories which the robot can autonomously switch between online, resulting in a hybrid form of trajectory generation.

10.1.5 Estimating Unmeasured States

As not all system states necessary in the controller are measured an estimator has been created. The problem is that only the angle and angular velocity of the motor shaft is measured while the angle that must be controlled is that of the joint shaft on the other side of the gear. Due to torsion and slack in the gears it was discovered that it was necessary to estimate the states of the joint shaft; hence, an estimator is set up.

Because of the large complex system model it has not been possible to neither linearize it properly nor differentiate it to create a Kalman estimator or an extended Kalman estimator. To include the entire non-linear system model in the estimator the procedure of an unscented Kalman estimator is

created. The developed estimator performs very well, even when some of the parameters describing the system are changed in order to emulate modeling uncertainties.

10.1.6 Designing Control System

In order to make the robot track the designed trajectory, a control system is designed consisting of a posture controller and a balance controller. Two strategies for the posture controller are designed and tested. The first strategy is based on a simple structure with an independent linear controller for each joint. This strategy showed to have some weaknesses concerning the cross couplings in the system and the nonlinear load in the joints. Therefore, a second strategy is designed that uses the nonlinear model in a feedback linearization and decoupling based on the computed-torque control method combined with linear controllers in order to track the joint angle references.

The balance controller is designed in an outer loop on the posture controller and should keep the robot in balance during the static gait. The balance controller uses the pitch and roll joints in the torso to set the torso in a position that keep the robot in balance. The output from the balance controller is a compensation angle that is added to the reference for the posture controller. The balance controller controls the x- and y-coordinates of the GCoM according to a reference for the GCoM specified by the trajectory.

The control system also consists of a phase estimator that determines whether the robot is in single support right, single support left, or in double support. This information is used in the control to switch between the two models describing the robot in single support right and left, respectively. The phase is determined based on the vertical forces in each foot generated from the ground contact.

10.1.7 The Final Results

The control system is only tested by simulation since it was not possible to test it on the actual **AAU-BOT1** within the time period of the project. Since the control system is based on models of the robot, different test scenarios with variations in the model parameters are used to test more realistic situations. The model parameters that are changed are the mass distribution, the coulomb friction, and the stiffness in the gear model. The mass distribution is changed since this affects the feedback linearization and decoupling in the control system. The coulomb friction is changed since the friction compensation in the control system is important due to the high friction in the joints. Last, the stiffness in the gear model is changed to test the performance of the Kalman estimator, since the gear model is a crucial part of the estimator.

The test results show that the control system has a god tracking ability of the joint angle reference and is able to make the robot perform static balanced gait. Even with model changes the control system can control the robot without significant performance changes. The Kalman estimator is also able to estimate the unmeasured states successfully with the changed gear model. The control system is therefore assumed to be able to make the actual **AAU-BOT1** able to perform static gait.

10.2 Conclusion

The area of robotics has become more and more interesting in order to develop humane like machines that are able to perform tasks that previously have been possible only for humans to do. Also in the field of studying human gait behaviors, robotics is of great interest since robots can be used to simulate different gait patterns and thereby gain knowledge about how gait patterns affect the human body. It is also with this purpose in mind that **AAU-BOT1** has been designed. **AAU-BOT1** is designed based on human gait patterns in order to give it the ability to perform different human like walking movements. The overall purpose of the **AAU-BOT1** project is to use the robot to simulate different human like dysfunctional properties and then study how a gait pattern in this situation affects other parts of the body. To fulfill this goal **AAU-BOT1** must first be able to walk with human like properties. Therefore, the purpose of this master's thesis has been to make **AAU-BOT1** able to perform static gait and develop a platform that can be used in the future work such that more advanced gait patterns can be tested on **AAU-BOT1**.

To be able to control the robot a complete interface has been developed consisting of a hardware interface between the on-board computer and the sensors and actuators, together with a real time software environment that is designed to interface with `MATLAB Simulink` to ease the implementation of different control systems. In order to simulate the behavior of the robot and to design a control system, a complete model of the robot is set up. The model is based on the previous work made on **AAU-BOT1**, which is further developed in order to cope with all the phases in the gait and to cover some features that have not been included earlier, e.g. the flexibility in the gears. Different versions of the model is utilized, a complete simulations model and simpler versions used in the control system.

A trajectory is designed that enables the robot to perform static balanced gait. The design of the trajectory includes the physical limitations such that the actual **AAU-BOT1** should be able to follow the trajectory. To make the robot walk according to the trajectory a control system is designed consisting of a Kalman estimator, a posture controller, and a balance controller. The Kalman estimator is designed as an unscented Kalman estimator that estimates the unmeasured joint angles based on the measured motor angles. The posture controller uses feedback linearization and decoupling based on the computed-torque control method, combined with linear controllers. The balance controller keeps the robot in balance by controlling the orientation of the torso.

The control system is tested in simulations and has accomplished making the robot complete a static gait cycle where it takes two steps. Furthermore, the control system is tested with changed parameters in the simulation model to cover the model uncertainties that may occur compared to the actual **AAU-BOT1**. The changed parameters consist of the mass distribution, the stiffness in the gear model, and the friction. All tests showed that the control system is able to make the robot perform static balance gait in the simulations. The control system has not been tested on the actual **AAU-BOT1** but the simulation test results indicates that it is possible to use the designed control system on the actual **AAU-BOT1**, which will be tested prior to the exam.

Conclusively, it has been achieved in this master's thesis to develop a hardware and software platform that can be used in the future work on **AAU-BOT1**. Furthermore, a complete model has also been set up in this project. The complete model can be used in the future as a basis in the design of different control systems. The control system designed in this project has proven to be effective in making **AAU-BOT1** perform static stable gait in simulations.

V

Appendices

A Calibration of the Force Torque Sensors

In this section the calibration and test of the Force Torque Sensors is described. The calibration is performed so that they can be used to measure the forces and torques that are acting in the ankle, and thereby determine the position of the CoP. The results of the FTS calibration can also be seen on the attached DVD [`FTS_Calibration/`].

The voltages sampled from the FTS are amplified in three strain gauges amplifiers, and to transform the measured signals into forces and torques the calibration is made to determine a calibration matrix. With use of the calibration matrix the measured signals can be directly transformed into forces and torques as shown in Eq. (A.1).

$$\boldsymbol{F} = \boldsymbol{C}\,\boldsymbol{V} \tag{A.1}$$

where:

$\boldsymbol{C}$ is the FTS calibration matrix

$\boldsymbol{F}$ is the measured forces and torques in the three principal axes [N; Nm]

$\boldsymbol{V}$ is the measured strain signals from the FTS amplifiers $\left[\frac{\mu m}{m}\right]$

The calibration matrix is to be determined based on experimental work where a known load is applied to the FTS and the signals generated by the FTS are recorded. The method used to determine the calibration matrix based on the experimental data is the least squares method which is a well known and commonly used method. The least squares method was shown by [Pedersen et al., 2007, p. 168] to be more accurate than the simpler and more straight forward method where only six measurements are used. The least squares method allows using an arbitrary large number of measurements which increases the accuracy of the calibration. The calibration matrix is determined from the least squares method as shown in Eq. (A.2).

$$\boldsymbol{C} = \boldsymbol{F}\,\boldsymbol{V}^T\left(\boldsymbol{V}\,\boldsymbol{V}^T\right)^{-1} \tag{A.2}$$

The calibration matrix will have a size of 6 x 6, and from the measurement $\mathbf{F}$ and $\mathbf{V}$ will have a size of 6 x m where m is the number of measurements.

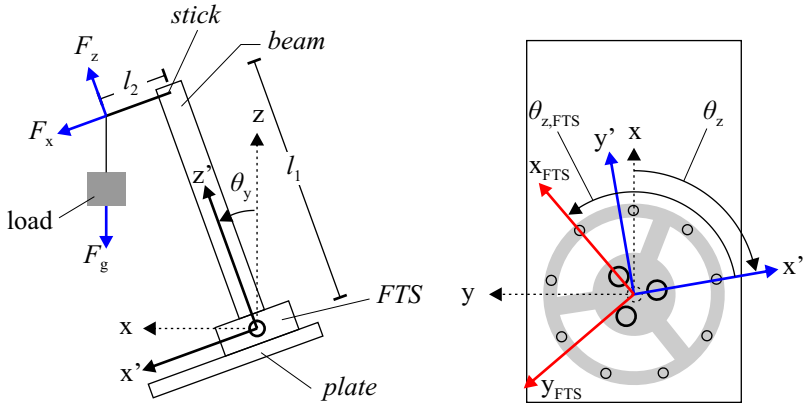
In the following the test set-up used for the calibration is described.

A.1 Calibration Test Set-Up

In order to get a good calibration it is important that when the FTS is excited in the test situation, it is mounted in the same way as it is mounted on the robot. For this purpose a test rig has been designed by the previous group [Jensen et al., 2008] and Ph.D. student Mads Sølver Svendsen. The test rig consists of a frame with a plate, and the FTS is mounted on this plate in the same way as it is mounted on the foot. Then, a beam is mounted on the FTS to represent the leg, and on top of the beam a stick is located which allows to excitate the FTS in all directions. The designed test rig is illustrated in Figure A.1. Pictures of the test set-up can be seen on the attached DVD in [`Pictures/`].

To get different measurements to the calibration, the test rig is tilted in different angles as shown in Figure A.1(a) and the FTS is rotated around the z-axis as shown in Figure A.1(b). The FTS is also rotated in relation to the beam, to get more realistic loads on the FTS. This is illustrated in Figure A.1(b) where the blue axis shows the orientation of the beam and the red axis shows the orientation of the FTS.

In each situation different loads are mounted on the stick. In Table A.1 the test situations and the four loads are listed. To test the calibration four other loads are applied. In each situation a measurement is made with no load which is used as a zero point for the FTS. Since the FTS is not



(a) Sagittal view of the calibration test set-up.

(b) Transverse view of the calibration test set-up.

Figure A.1: *Illustration of the FTS calibration test set-up. The blue axis illustrates the orientation of the beam and the red axis illustrates the orientation of the FTS.*

linear but an affine transformation the zero point of the FTS is subtracted from the measurements to achieve a linear transformation. This is needed since the least squares method assumes a linear model.

The on-board computer is used to sample the measurements from the FTS and afterwards a mean value from a period of 20 s is determined and used. To get a precise angle of the tilting the IMU is mounted on the frame of the test rig to measure the angles θ_x and θ_y, the rotation angle θ_z is fixed from the number of screws on the FTS.

Calibration Load

θ_x [°]	θ_y [°]	θ_z [°]	$\theta_{z,FTS}$ [°]	**Load [kg]**
0	9	0	0	5,15,25,35
0	20	0	0	5,15,25,35
0	34	0	0	5,15,25,35
0	9	80	120	5,15,25,35
0	20	80	120	5,15,25,35
0	34	80	120	5,15,25,35
0	9	-80	-120	5,15,25,35
0	20	-80	-120	5,15,25,35
0	34	-80	-120	5,15,25,35

Test Load

0	34	80	120	10,20,30,40
0	20	-80	-120	10,20,30,40

Table A.1: *FTS calibration test situations. The loads are used for both the left and right FTS, respectively.*

From the tilting and rotation angles and the applied load, the forces acting on the FTS can be calculated using an X-Y-Z Euler rotation as shown in Eq. (A.3). This assumes that the test rig is totally rigid and mass less. The beam is made hollow to reduce the weight; in this way the weight of the beam is low compared to applied load and is ignored. Also the beam is made of aluminum to make sure that the beam is very rigid.

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \boldsymbol{R}_z(\theta_{z,FTS}) \left[\boldsymbol{R}_x(\theta_x)\boldsymbol{R}_y(\theta_y)\boldsymbol{R}_z(\theta_z)\right]^{-1} \begin{bmatrix} 0 \\ 0 \\ F_g \end{bmatrix} \tag{A.3}$$

where:

F_x, F_y, and F_z	are the forces acting on the FTS [N]
F_g	is the load force acting vertical downwards from the stick [N]
$\boldsymbol{R}_x(\theta_x)$, $\boldsymbol{R}_y(\theta_y)$, and $\boldsymbol{R}_z(\theta_z)$	are the rotation matrices from the beam frame to the base frame
$\boldsymbol{R}_z(\theta_{z,FTS})$	is the rotation matrix from the beam frame to the FTS frame

The torques acting on the FTS are calculated from the forces and a skew matrix of the vector pointing to the end of the stick $[l_2\ 0\ l_1]^T$ as shown in Eq. (A.4).

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \boldsymbol{R}_z(\theta_{z,FTS}) \begin{bmatrix} 0 & -l_1 & 0 \\ l_1 & 0 & -l_2 \\ 0 & l_2 & 0 \end{bmatrix} [\boldsymbol{R}_x(\theta_x)\boldsymbol{R}_y(\theta_y)\boldsymbol{R}_z(\theta_z)]^{-1} \begin{bmatrix} 0 \\ 0 \\ F_g \end{bmatrix} \tag{A.4}$$

where:

M_x, M_y, and M_z are the torques acting on the FTS [N]

Note that in Eq. (A.4) the skew matrix becomes a singular matrix since the torques M_x and M_z both are calculated only from the force F_y. It can also be seen that a force having the direction of $[-l_2\ 0\ -l_1]^T$, i.e. from the stick to the bottom center of the beam, will not induce any torque, which is consistent with the fact that such a force should only induce forces on the FTS.

Having described the test set-up, the results of the calibration is described in the next section.

A.2 Calibration Results

In the following the results of the calibration of both FTSs are presented and discussed.

A.2.1 Right FTS

The resultant calibration matrix for the right FTS is shown in Eq. (A.5), and the RMS deviations between the applied test loads and the calculated loads are shown in Table A.2.

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} -0.0247 & 0.0301 & -0.0007 & -0.1314 & -0.0077 & 0.0190 \\ -0.0012 & 0.0194 & -0.0004 & 0.0090 & 0.0035 & 0.0119 \\ 0.2070 & -0.0041 & 0.2041 & 0.0220 & 0.2032 & -0.0019 \\ 0.0066 & -0.0001 & 0.0006 & -0.0011 & -0.0052 & -0.0002 \\ -0.0024 & 0.0004 & 0.0076 & 0.0031 & -0.0033 & -0.0003 \\ 0.0002 & -0.0021 & 0.0001 & -0.0010 & -0.0003 & -0.0015 \end{bmatrix} \begin{bmatrix} V_{s1} \\ V_{b1} \\ V_{s2} \\ V_{b2} \\ V_{s3} \\ V_{b3} \end{bmatrix} \tag{A.5}$$

where:

V_{b1}, V_{b2}, and V_{b3} are the measured bending from the FTS $\left[\frac{\mu m}{m}\right]$

V_{s1}, V_{s2}, and V_{s3} are the measured shear from the FTS $\left[\frac{\mu m}{m}\right]$

Force/Torque	RMS Deviation [%]
F_x	4.7
F_y	0.9
F_z	0.7
M_x	0.4
M_y	0.9
M_z	1.9

Table A.2: *Result of the calibration of the right FTS.*

The largest deviation is 4.7 % on F_x which partly is caused by the zero point of the FTS is changing, which was observed on measurements before and after having applied some loads to the FTS. The calibration made by the previous group indicated a large deviation on F_z which was

caused by the design of the test rig [Jensen et al., 2008, p. 217]. With the use of the redesigned test rig in this calibration it is clear that there is no larger deviation on F_z.

The forces and torques measured by the FTS are used to estimate the CoP which can be calculated with the use of F_z, M_x, and M_y, which all have a deviation less than 1 %.

In Figure A.2 the result of the calibration is shown as function of the applied load in the test situation.

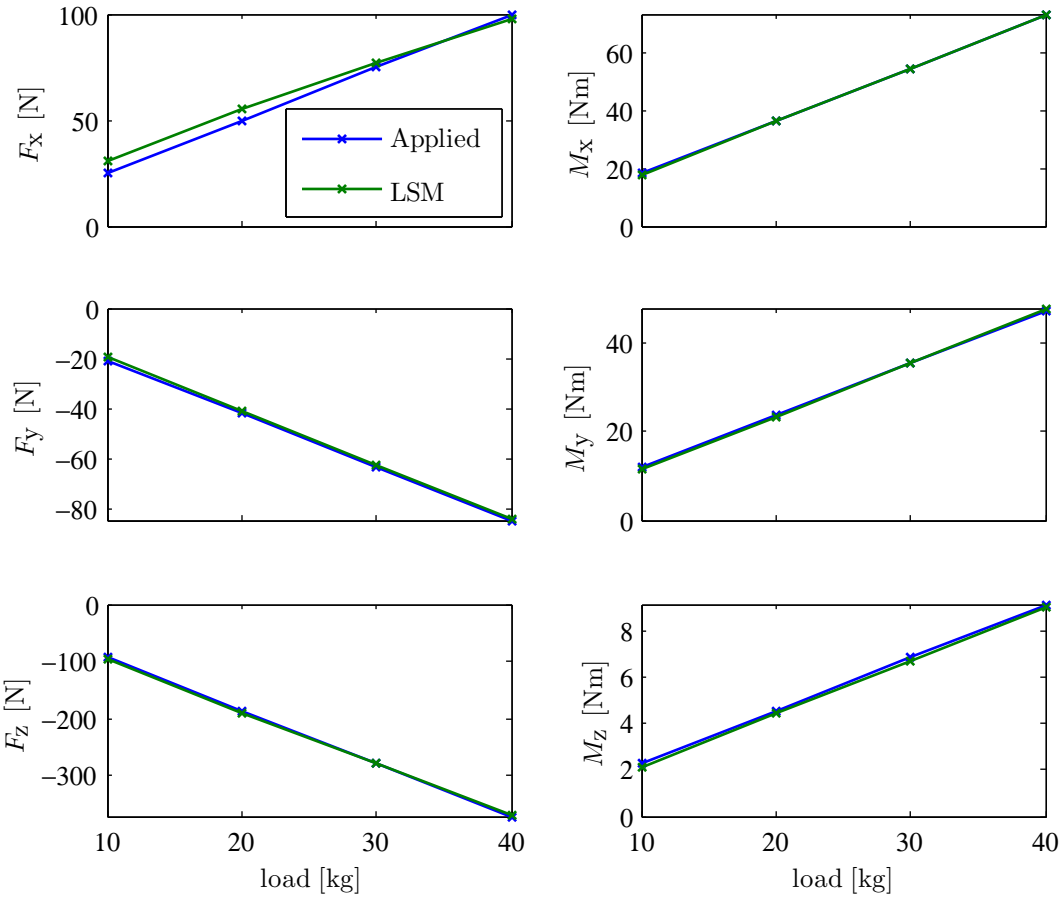


Figure A.2: *Test of the calibration for the right FTS as function of the applied load in the test, for the actual forces and torques and the forces and torques calculated from the calibration matrix.*

A.2.2 Left FTS

The resultant calibration matrix for the right FTS is shown in Eq. (A.6), and the RMS deviations between the applied test loads and the calculated loads are shown in Table A.3.

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} -0.0198 & -0.0073 & 0.0029 & -0.0160 & -0.0184 & 0.0225 \\ -0.0000 & 0.0139 & 0.0004 & 0.0241 & 0.0043 & 0.0041 \\ 0.2093 & 0.0008 & 0.2093 & -0.0113 & 0.2102 & 0.0001 \\ 0.0064 & -0.0005 & 0.0006 & -0.0009 & -0.0055 & 0.0002 \\ -0.0032 & 0.0002 & 0.0073 & 0.0012 & -0.0037 & -0.0000 \\ 0.0001 & -0.0012 & -0.0001 & -0.0025 & -0.0005 & -0.0005 \end{bmatrix} \begin{bmatrix} V_{s1} \\ V_{b1} \\ V_{s2} \\ V_{b2} \\ V_{s3} \\ V_{b3} \end{bmatrix} \quad \text{(A.6)}$$

The deviation is partly caused by the changing zero point for the test measurements as described for the right FTS. The changing zero point result in an offset deviation which is seen in Figure A.3. This offset deviation is reduced when the zero point for the used test measurements is corrected, which indicate that the calibration is satisfactory.

Force/Torque	RMS Deviation [%]
F_x	9.8
F_y	6.1
F_z	0.7
M_x	0.9
M_y	0.6
M_z	7.1

Table A.3: *Result of the calibration of the right FTS.*

As for the right FTS the calibration of the left FTS has small deviations on F_z, M_x, and M_y, which are used to estimate the CoP.

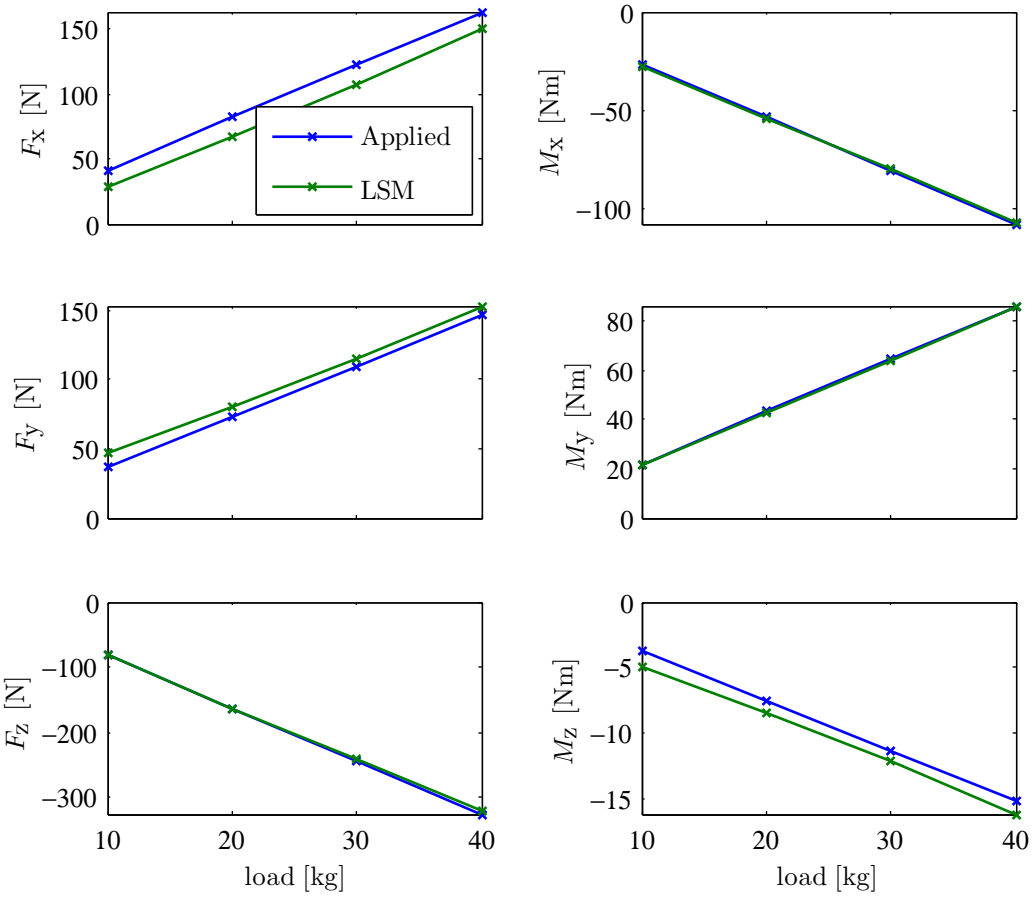


Figure A.3: *Test of the calibration for the left FTS as function of the applied load in the test, for the actual forces and torques and the forces and torques calculated from calibration matrix.*

In this section both of the FTSs have been calibrated and tested with satisfactory results. Though, it was observed that the zero point of the FTSs has a tendency to change after a load has been applied. To cope with this when the FTSs are mounted on **AAU-BOT1** an initial measurement from the FTSs are recorded and used as the zero point.

B Network Throughput Test

The purpose of the network throughput test is to evaluate the functionality and performance of the developed set-up. The elements that are tested in this test are the CAN drivers, the serial drivers, the AAU-BOT1 I/O module, and the AAU-BOT1 `MATLAB Simulink` interface. The results of the throughput test can also be seen on the attached DVD [`Network_Throughput_Test/`].

B.1 Test Method

The test is conducted by enabling the streaming mode for all of the FTSs and for the IMU to ensure that their influence on the system is the same as in the real control situation. For the purpose of this test the EPOSs are all set into position mode and a ramp is used as input signal. The input signal is designed so that for each sample the position is incremented by one encoder tick. In order to test the throughput to and from the EPOSs a special PDO package is set up. This PDO package is designed to transmit the position value demand sent to the EPOS instead of a measurement like in the regular packages. By doing this, the effect of the mechanics and the functionality of the EPOS regulators do not affect the throughput test. Furthermore, the EPOSs are set to transmit an additional PDO package in order to have the most traffic that can occur on the network.

Using this procedure the performance can be evaluated by checking the values returned by the EPOSs. If for any sample the value is not equal to the last value plus one there has been an error, either in the transmission of the command to the EPOS or in the reception of the value from the EPOS. Also, by using this procedure data from all of the EPOSs can be used at the same time, which gives 23 times as much test data.

The test is conducted for multiple sampling rates to prove that lowering the sample rate also lowers the communication errors. To ensure that the test basis is equal for all sample rates it is necessary to adjust the running time to correspond to the sample rate. For the nominal sample rate of 250 Hz the test is set to run for 70 s, and to ensure that the initialization process does not affect the results the first 10 s of the data set are not used in the evaluation. This gives a data set of $60\ \text{s} \cdot 250\ \text{Hz} = 15,000$ samples for each EPOS, which gives a total of 345,000 samples.

B.2 Test Results

The test has been conducted for the following sampling frequencies: 250 Hz, 400 Hz, and 500 Hz. The error rates are given in Table B.1. The data set for one of the EPOSs is shown in Figure B.1.

The results show that for the intended sampling frequency of 250 Hz there is only an insignificant error rate of 0.0017 %. Even when the sampling frequency is increased to 400 Hz there is an insignificant error rate of 0.0107 %. This indicates that the chosen sampling frequency of 250 Hz can be handled by the network and the drivers and is not close to the frequency where significant errors start to occur. The last column in Table B.1 shows the result of a test where the sampling frequency is increased beyond the tolerable frequency.

As the test has been conducted under the most heavy network load that can occur during the normal control scenario and still ran with only insignificant errors, it can be concluded that the CAN network and the hardware drivers can cope with a sampling frequency of 250 Hz without any problems.

CAN bus id	EPOS id	250 Hz	400 Hz	500 Hz
5	2	0.013 %	0.020 %	0.033 %
5	3	0.007 %	0.027 %	0.087 %
5	4	0.000 %	0.033 %	3.573 %
3	5	0.007 %	0.007 %	0.013 %
3	6	0.000 %	0.000 %	0.020 %
3	7	0.000 %	0.007 %	3.773 %
2	8	0.000 %	0.007 %	4.853 %
2	9	0.000 %	0.007 %	10.559 %
2	10	0.000 %	0.007 %	19.785 %
4	11	0.013 %	0.027 %	8.399 %
4	12	0.000 %	0.027 %	14.819 %
4	13	0.000 %	0.033 %	18.385 %
1	15	0.000 %	0.000 %	0.007 %
1	16	0.000 %	0.000 %	0.007 %
1	17	0.000 %	0.000 %	0.073 %
3	18	0.000 %	0.000 %	22.685 %
2	19	0.000 %	0.007 %	9.106 %
5	23	0.000 %	0.007 %	23.905 %
5	24	0.000 %	0.007 %	3.840 %
3	25	0.000 %	0.007 %	4.306 %
2	30	0.000 %	0.007 %	2.187 %
4	31	0.000 %	0.007 %	7.793 %
4	32	0.000 %	0.007 %	1.667 %
Total error		0.0017 %	0.0107 %	6.9511 %

Table B.1: *The error rate for each of the EPOSs during the throughput test at different sampling rates. The bottom row shows the error rates calculated using the entire data set.*

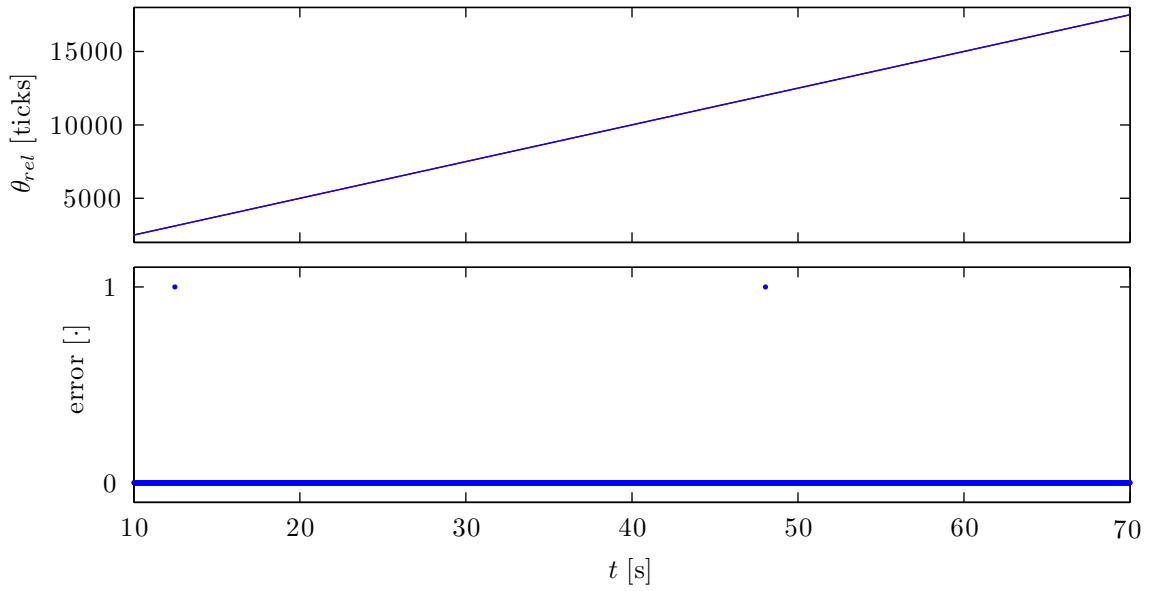


Figure B.1: *Result for the EPOS with id 2 during the performance test at 250 Hz. The top subplot shows the input signal together with the values returned by the EPOSs. The bottom subplot shows the errors, where a 0 indicates no error and a 1 indicates an error, i.e. only two errors are present in the figure.*

C Estimation of Friction Coefficients

In this appendix the friction coefficients in the joints of the robot are estimated based on experimental data. The procedure and results of the friction estimation can also be seen on the attached DVD [`Friction_Estimation/`].

C.1 Estimation Method

In order to get an accurate model of the real robot one of the important aspects is the model parameters. These must be known with a reasonable accuracy to get realistic behavior of the model corresponding to the real robot. Basically, two types of parameters exist, namely the kinematic and dynamic parameters, where the kinematic parameters are given from the mechanical structure of the robot. The dynamic parameters are the masses, moments of inertia, and the friction coefficients, where the masses and moments of inertia are determined from the technical drawings in `Solidworks`. This leaves the friction coefficients to be determined for each joint.

To estimate the friction coefficients the `MATLAB` toolbox `Senstools` is used [Knudsen, 2004]. The toolbox uses a model, an input, and a measured output from the real system. Using these, it estimates the parameters in the model such that the best fit between the model output and the measured output is achieved. The model used is simply a model of the DC-motor where the moment of inertia and the friction of the robot are transferred to the motor shaft. Since the DC-motor enables use of current control the model becomes as shown below:

$$I_{\mathrm{m}}\ddot{\theta}_{\mathrm{m}} = k_{\mathrm{t}} i_{\mathrm{m}} - \mu\dot{\theta}_{\mathrm{m}} - \tau_{\mathrm{c}} \tag{C.1}$$

where:

- I_{m} is the moment of inertia experienced by the motor [kg·m^2]
- θ_{m} is the angle of the motor shaft [rad]
- k_{t} is the motor torque constant [Nm/A]
- i_{m} is the motor current [A]
- μ is the viscous friction [Nms]
- τ_{c} is the Coulomb friction [Nm]

The friction coefficients to be estimated are the viscous friction, the Coulomb friction, and the stiction. Since the stiction only has an influence when the motor shaft begins to rotate it is not easy to estimate using `Senstools`. The stiction is therefore manual set such that the beginning of the rotation fits the measurements.

The friction coefficients should be estimated in all of the joints of the robot to get the most accurate model. But performing an estimation in all of the joints will be a very time consuming process. Since the motors and gears are all of the same type in all the joints, it is decided to make the estimation in one joint and then use these friction coefficients in all joints. The joint used for the estimation is the left hip yaw joint since this is, from a practically point of view, one of the easiest joint to use. This is due to the fact that influence from gravity does not change when the angle is changed, as long as the pelvis is horizontal.

The input used for the estimation is a square wave signal superimposed with a ramp as shown in Figure C.1. The input is designed with use of a tool in `Senstools` and then modified to obey the practical limitations. During the estimation it is the measured input that is used to make sure that the model is given the same input as the real DC-motor.

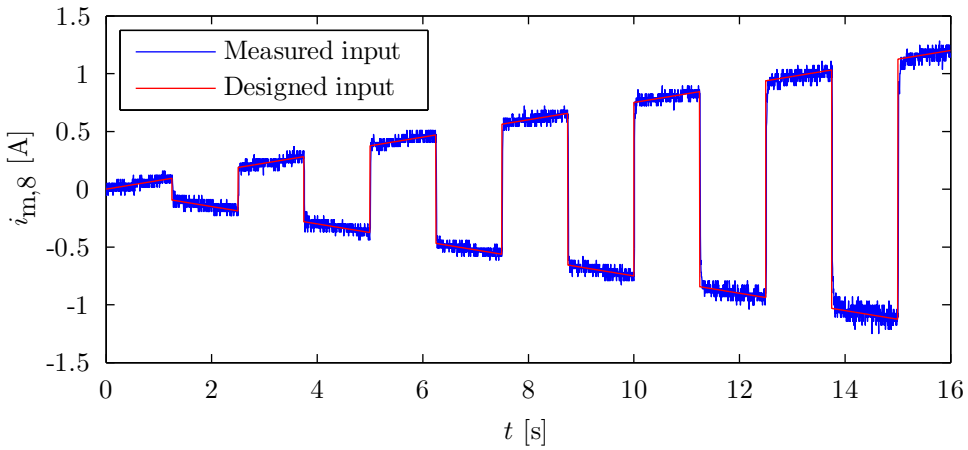


Figure C.1: *The designed input and the measured input current applied to the motor.*

C.2 Estimation Results

The result of the estimation is shown in Figure C.2. The parameters estimated by `Senstools` are shown below:

- Viscous friction
  $\mu = 5.28$ Nm
- Coulomb friction
  $\tau_c = 10.19$ Nm
- Stiction
  $\tau_s = 1$ Nm

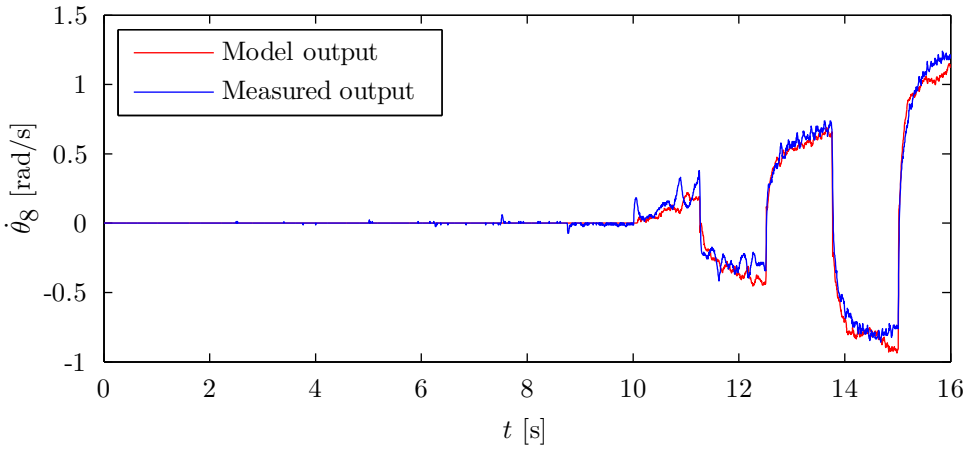


Figure C.2: *The result of the estimation for the right leg illustrated by the measured output compared to the model output with the estimated friction coefficients.*

The deviation between the model and the real measurement is determined by `Senstools` as the normed root mean squared output error and is 15 %, which is considered to be sufficiently good.

D Double Support Suggestions

In order to model the robot in double support some different approaches has been tried out in this project. In this appendix some of the alternative suggestions are described.

D.1 Double Support Modeling by Extra Arm

A large problem with using a single support model to model double support is that without any special care one of the feet will always be locked to the floor in all directions, while the other foot is free to move. Ideally, both feet would be locked to the floor, but that is difficult to accomplish. Instead, the purpose of the proposed model is to make both feet free, i.e. not locked to the floor, in order to model them in the same manner.

Modeling the first foot in the kinematic chain as being free was not possible in the previous method as it was used as the origin of all calculations. Instead, the new proposed method utilizes an imaginary point in front of the robot as origin. The new chosen origin is connected to the origin in one of the feet by an arm. This arm is set up of 6 individual joints to form a manipulator capable of reaching all positions and orientations within its reach. The manipulator has a roll and a pitch joint positioned in the same point at the origin, then an arm and a single joint in the pitch direction, followed by a second arm and a roll, a pitch, and a yaw joint in the same point. This manipulator set-up is illustrated in Figure D.1.

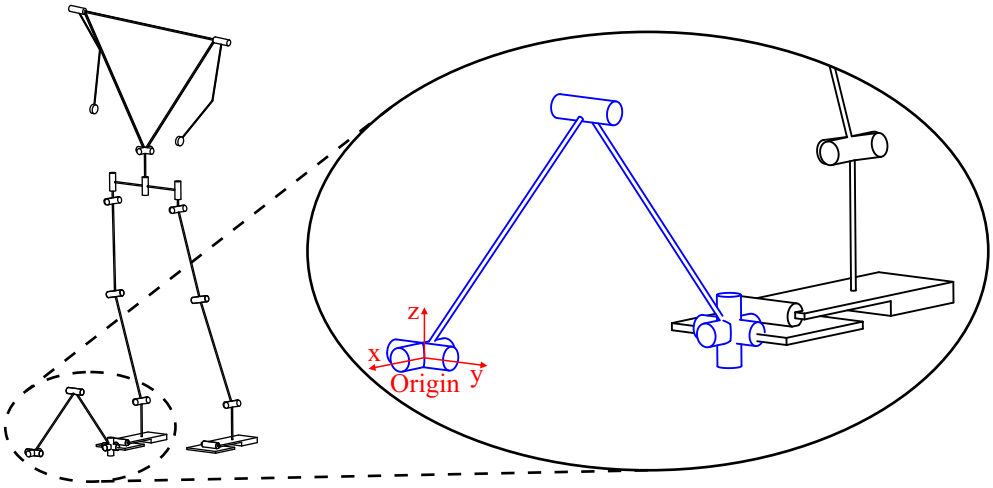


Figure D.1: *The proposed support model. The blue joints and links are the new additions, while the black part is the actual robot.*

By making all new links mass less and all new joints friction less, it is ensured that they just follow the motions of the robot without any influence on it, as long as the reaching limit of the manipulator is not exceeded. Now, when both feet are free to move, they can be equipped with a number of springs to model the ground reaction forces. The utilized springs are shown in Figure D.2. The purpose of spring 1 and 2 is to keep the foot parallel to the ground to keep the full foot in contact with the ground. Spring 3 is keeping the toe aligned with the ground in the direction not handled by spring 1 and 2.

Spring 4 models the major part of the ground reaction force acting on the toe, namely the vertical part, whenever the toe moves below the ground plane the spring imprints an opposite force, keeping the toe close to the ground plane. In the simulation the spring constant is chosen such that the toe never gets more than 1 mm below ground level. In the exact same manner spring 5 models the vertical part of the ground reaction force on the heel.

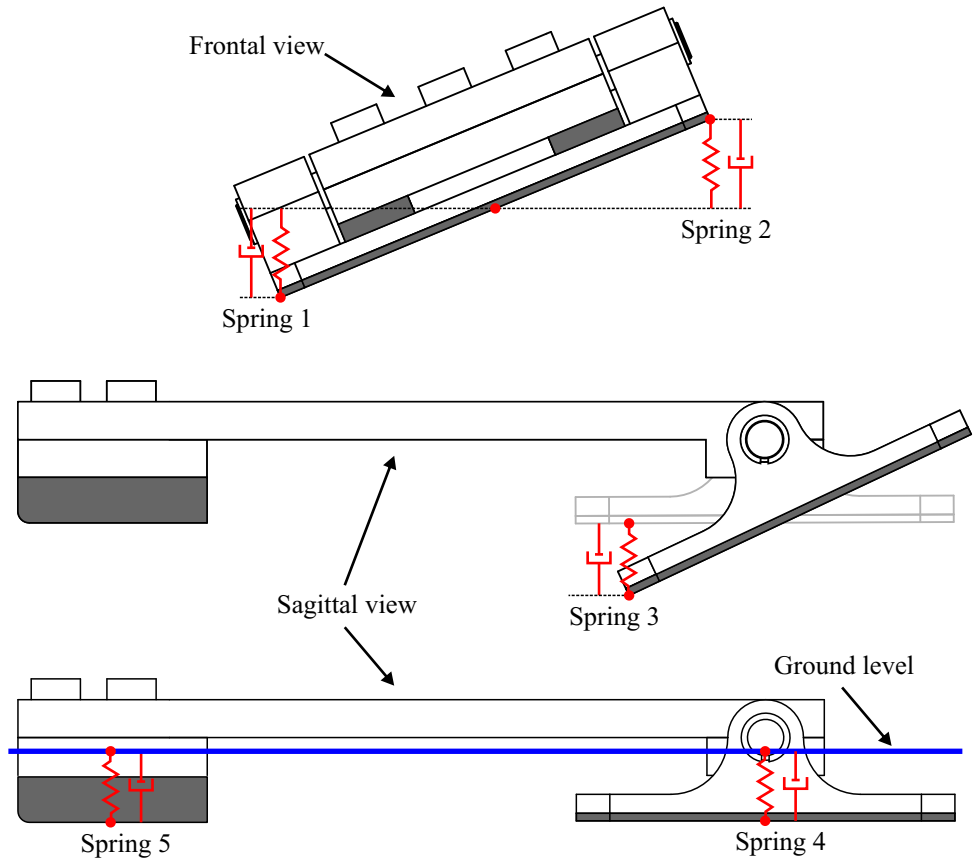


Figure D.2: *Location of springs used to model ground reaction forces. The ground level, illustrated by the blue line in the bottom part of the figure, is raised significantly compared to the actual level for illustrative purposes. (The part showing horizontal springs is missing).*

Similarly, three springs are used to model the horizontal part of the ground reaction forces. On the toe, two springs are utilized, one in the x- and one in the y-direction. The last spring is positioned in the heel and acts in the y-direction to prevent the foot from rotating around the z-axis.

The point of attachment with the ground for the horizontal springs is recorded at the time of impact when the foot hits the ground and the springs start to work.

D.2 Ground Reaction Forces

When some part of the robot touches the ground, a force from the ground acts upon the touching part. This force is the ground reaction force. To have a realistic model of **AAU-BOT1** it is necessary to include this force, as erroneous torques will be seen in the joints otherwise. In single support the ground reaction forces acts upon the toe and the heel of the supporting foot. As the origin for all calculations is the toe, only the ground reaction force affecting the heel must be included. Note that in the following only the ground reaction forces for the right foot are set up, i.e. the index r is added to the variables, but the procedure for the left foot is the same.

The ground reaction force affecting the heel is calculated based on the resultant of the normal forces of the entire robot. The resultant of the normal forces acts upon the CoP of the robot, and is easily calculated by multiplying the total mass of the robot with the gravitational acceleration. The ground reaction force on the heel is determined as a weighting based on the x-component of the distance from the CoP to the heel and the distance between the heel and the toe, as shown in

Figure D.3, and given by the following equation:

$$w_{\mathrm{r}} = \begin{cases} 0 & \text{if } p_{\mathrm{CoP,x}} > p_{\mathrm{r,toe,x}} \\ 1 & \text{if } p_{\mathrm{CoP,x}} < p_{\mathrm{r,heel,x}} \\ \frac{p_{\mathrm{r,toe,x}} - p_{\mathrm{CoP,x}}}{p_{\mathrm{r,toe,x}} - p_{\mathrm{r,heel,x}}} & \text{otherwise} \end{cases} \tag{D.1}$$

where:

$p_{\mathrm{CoP,x}}$ is the x-component of the CoP coordinate [m]
$p_{\mathrm{r,heel,x}}$ is the x-component of the coordinate of the heel joint [m]
$p_{\mathrm{r,toe,x}}$ is the x-component of the coordinate of the toe joint [m]
w_{r} is the weighting factor between heel and toe support for the right foot [·]

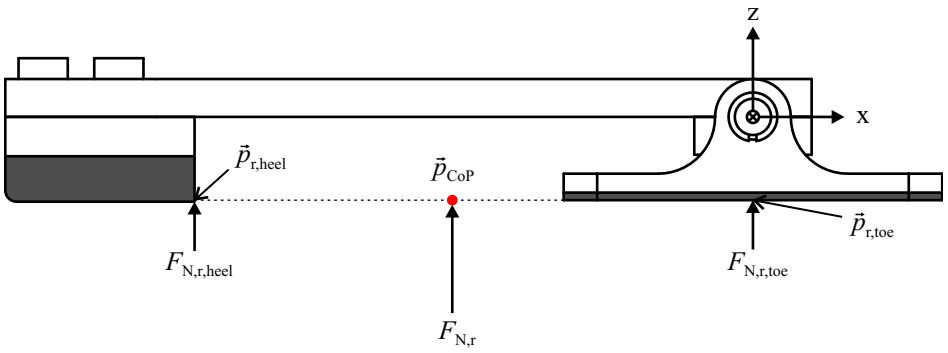


Figure D.3: *Distribution of normal forces on the two support points of the foot.*

The normal force acting on the heel is then given by the following equation:

$$F_{\mathrm{N,r,heel}} = w_{\mathrm{r}} F_{\mathrm{N,r}} \tag{D.2}$$

where:

$F_{\mathrm{N,r}}$ is the resultant of the normal forces acting upon the CoP [N]
$F_{\mathrm{N,r,heel}}$ is the normal force acting upon the heel support point [N]

When w_{r} becomes 1, i.e. the CoP has moved behind the point p_{heel}, the interaction point of the heel normal force is moved so that the x-coordinate follows the x-coordinate of the CoP.

To include these forces into the model the generalized coordinate vector defined in Eq. (5.1) on page 37 is extended to include the coordinates of the heel support point. Then Eq. (5.12) on page 43 describing the potential energy in the system is modified to include the force in the heel so it becomes as shown below:

$$E_{\mathrm{pot,r}} = \sum_{i=1}^{N} (m_i \mathrm{g} z_i) + k_{\mathrm{t}} (\theta_1 + \theta_{14}) - E_{\mathrm{grf,r}} \tag{D.3}$$

$$E_{\mathrm{grf,r}} = F_{\mathrm{N,r,heel}} z_{\mathrm{r,heel}} \tag{D.4}$$

where:

$z_{\mathrm{r,heel}}$ is the z-coordinate of the heel support point [m]
$E_{\mathrm{grf,r}}$ is the potential energy contribution from the ground reaction forces [J]

Note that $z_{\mathrm{r,heel}}$ is equal to 0 at all times when a normal force is affecting the point, but is needed in order to get the correct relations in the formulas based upon the potential energy. After these modifications, the formulas presented in Subsection 5.3.1 on page 43 can be applied directly.

D.2.1 Dynamic Model in Double Support

A simple approach is used to handle DSP, based on a weighting of the two SSP models for left and right as shown in the equation below.

$$\vec{\tau}_{\mathrm{DSP}} = (1 - w_{\mathrm{DSP}}) \vec{\tau}_{\mathrm{R}} + w_{\mathrm{DSP}} \vec{\tau}_{\mathrm{L}} \tag{D.5}$$

where:

$\vec{\tau}_{\text{DSP}}$ is the torque of each link in DSP [Nm]
$\vec{\tau}_{\text{L}}$ is the torque of each link in SSP-L [Nm]
$\vec{\tau}_{\text{R}}$ is the torque of each link in SSP-R [Nm]
w_{DSP} is the weighting factor between the feet [·]

This results in an equation for the angular acceleration, different from Eq. (5.18) on page 45, as shown below:

$$\ddot{\vec{\theta}} = \left(w_{\text{DSP}}\boldsymbol{M}_{\text{l}}(\vec{\theta}) + (1-w_{\text{DSP}})\,\boldsymbol{M}_{\text{r}}(\vec{\theta})\right)^{-1}\left(\vec{\tau} - w_{\text{DSP}}\left(\vec{V}_{\text{l}}(\vec{\theta},\dot{\vec{\theta}}) + \vec{G}_{\text{l}}(\vec{\theta})\right)\right.$$
$$\left. - (1-w_{\text{DSP}})\left(\vec{V}_{\text{r}}(\vec{\theta},\dot{\vec{\theta}}) + \vec{G}_{\text{r}}(\vec{\theta})\right)\right) \tag{D.6}$$

Furthermore, the ground reaction forces acting on the foot opposite to the supporting foot of the SSP model needs to be included. This is done by first distributing the resultant of the normal forces to each foot by a weighting with w_{DSP}. Afterwards, the force is distributed to the heel and the toe exactly as described for the supporting foot in the last subsection. This results in the ground reaction forces as follows:

$$\begin{aligned}
F_{\text{N,r,heel}} &= (1-w_{\text{DSP}})\,w_{\text{r}}F_{\text{N}}\\
F_{\text{N,r,toe}} &= (1-w_{\text{DSP}})\,(1-w_{\text{r}})\,F_{\text{N}}\\
F_{\text{N,l,heel}} &= w_{\text{DSP}}w_{\text{l}}F_{\text{N}}\\
F_{\text{N,l,toe}} &= w_{\text{DSP}}\,(1-w_{\text{l}})\,F_{\text{N}}
\end{aligned}$$

Then, the potential energy included from the reaction forces in double support becomes as shown in Eq. (D.7) for the right model and as shown in Eq. (D.8) for the left model.

$$E_{\text{grf,r}} = F_{\text{N,r,heel}}z_{\text{r,heel}} + F_{\text{N,l,heel}}z_{\text{l,heel}} + F_{\text{N,l,toe}}z_{\text{l,toe}} \tag{D.7}$$
$$E_{\text{grf,l}} = F_{\text{N,r,heel}}z_{\text{r,heel}} + F_{\text{N,r,toe}}z_{\text{r,toe}} + F_{\text{N,l,heel}}z_{\text{l,heel}} \tag{D.8}$$

D.2.2 Double Support Weight Factor

The double support weighting factor, w_{DSP}, is calculated based on the distance between each foot and the CoP, e.g. the further away the CoP is from the right foot the less pressure is on the right foot and therefore the right model is weighted less. The weighting factor is calculated as shown in Eq. (D.9). When stable gait is assumed the CoP equals the ZMP and is calculated from Eq. (2.8) on page 14 and Eq. (2.9) on page 14.

$$w_{\text{DSP}} = \begin{cases} 0 & \text{if CoP is located on the right foot}\\ 1 & \text{if CoP is located on the left foot}\\ \frac{\Delta y_{\text{right}}}{\Delta y_{\text{right}}+\Delta y_{\text{left}}} & \text{otherwise}\end{cases} \tag{D.9}$$

where:

Δy_{right} is the distance from the CoP to the right foot in the y-direction [m]
Δy_{left} is the distance from the CoP to the left foot in the y-direction [m]

The distance used in the calculation of the weight factor is the y-component of the shortest distance between the CoP and the convex hull that compose the support area of the foot. This ensures that the weight factor becomes 1 or 0 when the CoP is located inside the support area of one of the feet, i.e. all the weight of the robot is placed over one foot. Alternative a fixed point on the foot can be used to calculate the distance, but then the weight factor will give a contribution to both feet even when the CoP is located inside the support area of one of the feet.

An example of the distances is shown in Figure D.4. The convex hull is defined by a set of points where the half circle around the heel is defined by 30 points to get a reasonable ratio between accuracy and computations time.

To calculate the y-component of the shortest distance, the intersection point, p_{int}, located on the edge of the convex hull is used. The intersection point is calculated as the intersection by two lines, as illustrated in Figure D.5, where the line between p_1 and p_2 compose the first line. The

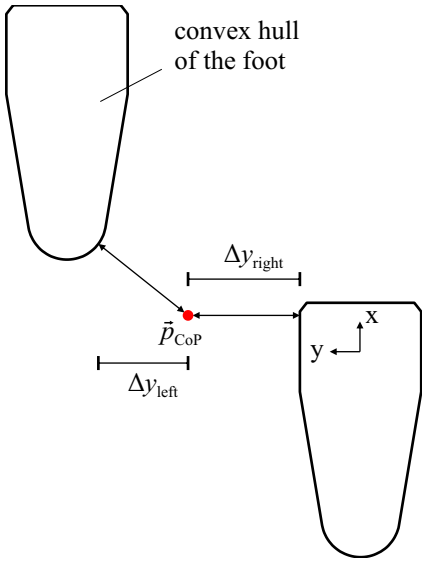


Figure D.4: *The distances between the CoP and the feet used to calculate the double support weight factor.*

second line used is generated by adding a vector to p_{CoP} being perpendicular to the first line. Since the convex hull is defined by a set of points a corresponding set of distances is calculated. When the sequence of points from the convex hull used to calculate the distance is in a counterclockwise order, it results in a negative distance when the CoP is located inside the convex hull and positive when located outside. Hence, it is the minimum of the positive distances that is needed.

In some situations it is not the nearest intersection point that is of interest. In the situation where the nearest intersection point lies outside p_1 and p_2, the nearest point of interest will be either p_1 or p_2. Such a situation is illustrated in Figure D.5.

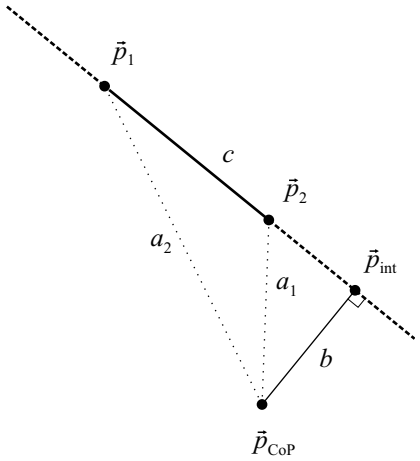


Figure D.5: *Illustration of how the distance to the convex hull is determined.*

To ensure the correct intersection point in all situations the following algorithm is used:

```
if a1²-b²>c² or a2²-b²>c²
   if a2 > a1
      pi = p1
   else
      pi = p2
   end
else
   pi = pint
end
```

The algorithm ensures that if the perpendicular distance is outside the two points, it is simply the nearest end point that is used.

D.3 Calculation of Ground Reaction Forces

Another approach to distribute the forces by weight factors, as described in the previous section, is to calculate the forces that act on the four support points, i.e. the toes and feet. During static gait the movement of the robot is rather slow and with slow movement the reaction forces can be assumed to be static, therefore one suggestion is to calculate the vertical static forces. This can be done by solving the equations Eq. (D.11), which says that the resultant torque around the CoM must be zero, and Eq. (D.12), which says that the resultant force acting on the robot must be zero.

$$\vec{r}_1 \times \vec{F}_{\mathrm{n1}} + \vec{r}_2 \times \vec{F}_{\mathrm{n2}} + \vec{r}_3 \times \vec{F}_{\mathrm{n3}} + \vec{r}_4 \times \vec{F}_{\mathrm{n4}} = 0 \tag{D.10}$$

$$\begin{bmatrix} r_{1,\mathrm{y}}F_{\mathrm{n1,z}} + r_{2,\mathrm{y}}F_{\mathrm{n2,z}} + r_{3,\mathrm{y}}F_{\mathrm{n3,z}} + r_{4,\mathrm{y}}F_{\mathrm{n4,z}} \\ -r_{1,\mathrm{x}}F_{\mathrm{n1,z}} - r_{2,\mathrm{x}}F_{\mathrm{n2,z}} - r_{3,\mathrm{x}}F_{\mathrm{n3,z}} - r_{4,\mathrm{x}}F_{\mathrm{n4,z}} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{D.11}$$

where:

$\vec{F}_{\mathrm{n}i}$ is the reaction force in the ith support point [N]

$\vec{r}_i$ is the vector pointing from the ith support point to the CoM [m]

$$F_{\mathrm{g}} + F_{\mathrm{n1,z}} + F_{\mathrm{n2,z}} + F_{\mathrm{n3,z}} + F_{\mathrm{n4,z}} = 0 \tag{D.12}$$

As it can be seen there exist four unknown parameters and only three equations, hence the problem is static indeterminate. According to theory, a system is static indeterminate when there is more supporting points than actually needed [Meriam and Kraige, 1993, p. 163]. Therefore, in order to calculate the static forces one of them must be known, by setting one to zero the remaining static forces can be calculated as shown:

$$F_{\mathrm{n1,z}} = 0 \tag{D.13}$$

$$F_{\mathrm{n2,z}} = \frac{r_{3,\mathrm{x}}r_{4,\mathrm{y}} - r_{4,\mathrm{x}}r_{3,\mathrm{y}}}{r_{3,\mathrm{x}}r_{4,\mathrm{y}} - r_{4,\mathrm{x}}r_{3,\mathrm{y}} - r_{2,\mathrm{x}}r_{4,\mathrm{y}} + r_{4,\mathrm{x}}r_{2,\mathrm{y}} + r_{2,\mathrm{x}}r_{3,\mathrm{y}} - r_{3,\mathrm{x}}r_{2,\mathrm{y}}}F_{\mathrm{g}} \tag{D.14}$$

$$F_{\mathrm{n3,z}} = \frac{r_{4,\mathrm{x}}r_{2,\mathrm{y}} - r_{2,\mathrm{x}}r_{4,\mathrm{y}}}{r_{3,\mathrm{x}}r_{4,\mathrm{y}} - r_{4,\mathrm{x}}r_{3,\mathrm{y}} - r_{2,\mathrm{x}}r_{4,\mathrm{y}} + r_{4,\mathrm{x}}r_{2,\mathrm{y}} + r_{2,\mathrm{x}}r_{3,\mathrm{y}} - r_{3,\mathrm{x}}r_{2,\mathrm{y}}}F_{\mathrm{g}} \tag{D.15}$$

$$F_{\mathrm{n4,z}} = \frac{r_{2,\mathrm{x}}r_{3,\mathrm{y}} - r_{3,\mathrm{x}}r_{2,\mathrm{y}}}{r_{3,\mathrm{x}}r_{4,\mathrm{y}} - r_{4,\mathrm{x}}r_{3,\mathrm{y}} - r_{2,\mathrm{x}}r_{4,\mathrm{y}} + r_{4,\mathrm{x}}r_{2,\mathrm{y}} + r_{2,\mathrm{x}}r_{3,\mathrm{y}} - r_{3,\mathrm{x}}r_{2,\mathrm{y}}}F_{\mathrm{g}} \tag{D.16}$$

E Simple Control Design

In this appendix the design and test of a rather simple control strategy will be presented. The purpose of designing a simple control strategy is to study whether such a controller is able to make the **AAU-BOT1** perform static gait.

The basic purpose of the controller is to make the robot follow a certain trajectory, meaning that each joint must follow the corresponding angle trajectory. This invites to design individual controllers for each joint that are able to follow an angle reference. The use of individual controllers can be problematic due to the cross couplings that naturally exist in a biped robot. But by designing and tuning the controllers appropriately it should be possible to suppress the effect of the cross couplings to an acceptable level. When the controllers are designed individually for each joint it can be treated as a SISO system where the joint torque is the input and the joint angle is the output. When designing a controller for a SISO system a natural approach would be to design a PID controller, but it turns out that it is difficult to tune a PID controller to control the robot, in practice. This is due to the fact that the robot can be seen as an inverted pendulum when only a single joint is considered, and tuning a PID controller for a pendulum is a rather difficult task.

Therefore, it is decided to design a simple state feedback controller that utilizes both the joint angle and the angular velocity as shown in Figure E.1.

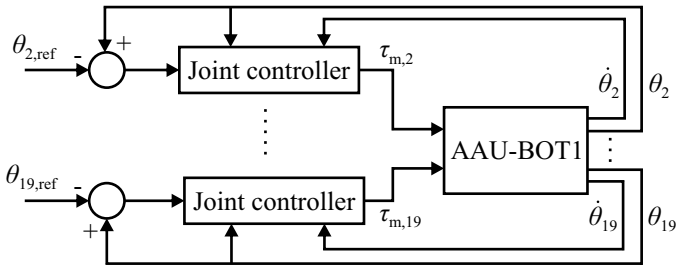


Figure E.1: *The structure of the joint controllers with independent controller for each joint. .*

For designing a state feedback controller an appropriate model is needed. A description is given in the following subsection of the models that are used to design an optimal controller, i.e. an LQR controller.

Models for LQR Design

To design the optimal LQR for each joint, a model is needed in each joint. The principle used for designing the joint controllers is to design one controller at a time and then assume that the remaining joints are locked. When the remaining joints are locked the robot can be considered as a rigid body acting as an inverted pendulum on the single free joint. This is illustrated in Figure E.2, where the right ankle roll joint is considered and the rest of the robot is reduced to a point mass. Note that it is only the part of the robot that, in the kinematic chain, is placed above the considered joint, meaning that the right foot is not included. The assumption of the locked joints is only valid if the controllers are able to hold the angles in the joints independent of the movement of the other joints. Since the purpose of the controllers is to hold an angle reference the assumption is valid.

The robot with one free joint can then be modeled as an inverted pendulum, which is shown in Eq. (E.1). Note that the inverted pendulum has been linearized at the operating point $\theta = 0$ rad and thereby assuming that $\sin(\theta) \approx \theta$. Also, the Coulomb friction has been omitted in the linearization.

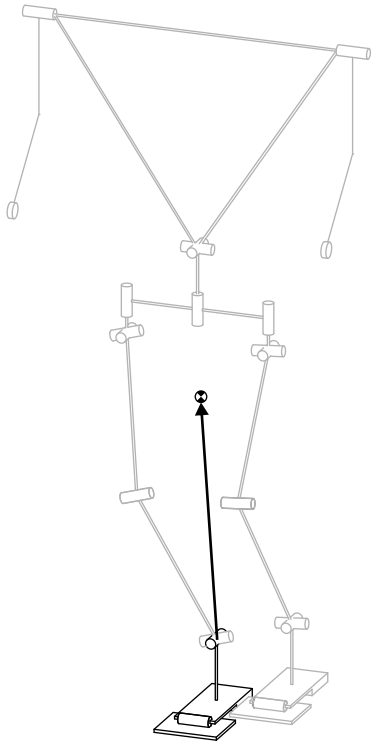


Figure E.2: *The principle of considering the robot as an inverted pendulum. The part of the robot that is marked with gray is included in the pendulum mass and inertia.*

$$I\ddot{\theta} = Gk_{\mathrm{t}}i_{\mathrm{m}} + mgl\theta - \mu\dot{\theta} \tag{E.1}$$

where:

- I is the moment of inertia [kg·m^2]
- θ is the joint angle [rad]
- G is the total gear ratio in the joint [·]
- k_{t} is the motor torque constant [Nm/A]
- i_{m} is the motor current [A]
- μ is the viscous friction [Nms]
- m is the point mass of the pendulum [kg]
- g is the gravitational acceleration constant [m/s^2]
- l is the length of the pendulum arm [m]

The length of the pendulum arm is calculated as the distance from the axis of rotation in the joint to the center of mass of the remaining part of the robot. The moment of inertia of the pendulum is calculated from the inertia tensors shown in Table 5.4 on page 44 with use of the parallel axis theorem. Since the mass and the moment of inertia in the inverse pendulum model is depended on the joint angles, i.e. the position of the robot, the robot is held in a position that correspond to the single support phase.

Note that the control signal for the robot is normally stated as the joint torque, as shown in Figure E.1, but in the controller design it is represented as the motor current, as shown in the pendulum model in Eq. (E.1). This change can be performed easily as the torque is linearly related to the motor current. The change is made because the DC-motor is driven in the current control mode.

The inverted pendulum is set up in the state space form as shown below:

$$\dot{\vec{x}} = \mathbf{A}\vec{x} + \mathbf{B}i_{\mathrm{m}} \tag{E.2}$$

$$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 1\\ \frac{mgl}{I} & -\frac{\mu}{I}\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0\\ \frac{Gk_{\mathrm{t}}}{I}\end{bmatrix} i_{\mathrm{m}}$$

$$y = \mathbf{C}\vec{x} \tag{E.3}$$

$$= \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix}$$

To design the controllers for the joints in the swinging leg, the model used becomes a pendulum instead of an inverted pendulum. This is caused by the fact that the center of mass is located underneath the considered joint, as illustrated in Figure E.3.

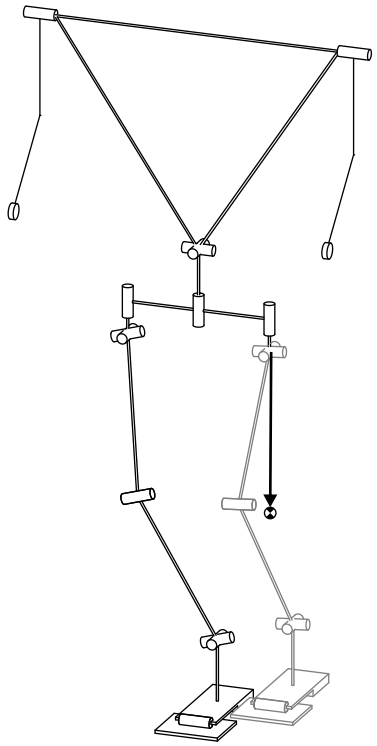


Figure E.3: *The principle of considering the swing leg of the robot as a pendulum. The part of the robot that is marked with gray is included in the pendulum mass and inertia.*

The model for the pendulum is shown in Eq. (E.4), where the difference compared to the inverted pendulum is the sign on the gravity term.

$$I\ddot{\theta} = Gk_{\mathrm{t}}i_{\mathrm{m}} - mgl\theta - \mu\dot{\theta} \tag{E.4}$$

The three yaw joint, J_7, J_8, and J_{15}, illustrated in Figure 5.3 on page 38, are not affect by the gravity as long as pelvis is held horizontal. Therefore, the gravity term in the pendulum model is ignored in the model used to design controllers for the yaw joints.

The joint controllers are designed as LQRs which are illustrated as a block diagram in Figure E.4. Besides the standard LQR a feed forward gain is added from the reference signal to reduce the response time.

The controllers are designed including integral gains and following the procedure described in Subsection 6.3.1 on page 65.

When the LQR controller is tuned the feed forward gain is added with a gain as high as possible without generating an overshoot.

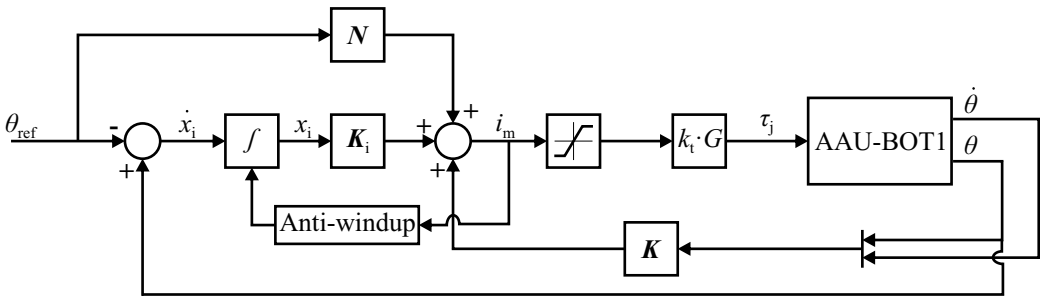


Figure E.4: *Block diagram of the LQR-controller.*

Bibliography

[Andersen and Pedersen, 2007] Andersen, P. and Pedersen, T. S. (2007). Modeldannelse. http://www.control.aau.dk/%7Epa/kurser/PR6model/modelnote.pdf 02/03-2009.

[Anybots, 2007] Anybots (2007). *Dexter*. http://anybots.com/abouttherobots.html.

[Bachar, 2004] Bachar, Y. (2004). *Developing Controllers for Biped Humanoid Locomotion*. Master's thesis, University of Edinburgh. http://www.inf.ed.ac.uk/publications/thesis/online/IM040191.pdf 01/19-2009.

[Buschmann et al., 2006] Buschmann, T., Lohmeier, S., Ulbrich, H., and Pfeiffer, F. (2006). Dynamics Simulation for a Biped Robot: Modeling and Experimental Verification. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 2673–2678, Institute of Applied Mechanics, Technische Universität München, Munich, Germany. http://ieeexplore.ieee.org/xpls/abs_all.jsp?tp=&arnumber=1642105.

[Caux and Zapata, 1999] Caux, S. and Zapata, R. (1999). Modeling and control of biped robot dynamics. In *Robotica*, volume 17, pages 413–426, Laboratorie d'Informatique, de Robotique et de Microélectronique, LIRMM-UM CNRS C5560-Université de Montpellier. http://portal.acm.org/citation.cfm?id=980471.980479.

[Choi et al., 2004] Choi, Y., You, B.-J., and Oh, S.-R. (2004). On the stability of indirect ZMP controller for biped robot systems. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004)*, volume 2, pages 1966–1971. http://ieeexplore.ieee.org/xpls/abs_all.jsp?tp=&arnumber=1389686.

[Christensen et al., 2007a] Christensen, J., Nielsen, J. L., Svendsen, M. S., and Ørts, P. F. (2007a). *Development, Modeling and Control of a Humanoid Robot*. Master's thesis, Aalborg University. Project group: 07gr1033 http://projekter.aau.dk/projekter/research/a_humanoid_robot(9841892)/.

[Christensen et al., 2007b] Christensen, R., Fogh, N., Hansen, R. H., Hansen, H., Iversen, A. M., Jensen, M. S., and Lantow, L. S. (2007b). *Modelling and Control of a Biped Robot*. Technical report. Aalborg University. Project group: 07gr830 http://projekter.aau.dk/projekter/research/modelling_and_control_of_a_biped_robot(9776435)/.

[Collins et al., 2001] Collins, S. H., Wisse, M., and Ruina, A. (2001). A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research*, 20(7):607–615. http://ruina.tam.cornell.edu/hplab/downloads/walking_papers/CollinsWisseRuina.pdf.

[Craig, 2005] Craig, J. J. (2005). *Introduction to Robotics Mechanics and Control*. Pearson Prentice Hall, 3. edition. ISBN: 0-13-123629-6.

[Esbensen et al., 2007] Esbensen, T., Jensen, B. T., Niss, M. O., and Sloth, C. (2007). *Velocity Control of Portal Crane*. Technical report. Aalborg University. Project group: 07gr632 http://projekter.aau.dk/projekter/fbspretrieve/9740868/Report.pdf.

[Esbensen et al., 2008] Esbensen, T., Jensen, B. T., Niss, M. O., and Sloth, C. (2008). *Joint Power and Speed Control of Wind Turbines*. Technical report. Aalborg University.
Project group: 08gr830
http://projekter.aau.dk/projekter/fbspretrieve/14458968/Report.pdf.

[Goswami, 1999] Goswami, A. (1999). *Postural Stability of Biped Robots and the Foot-Rotation Indicator (FRI) Point*. In *The International Journal of Robotics Research*, volume 18, pages 523–533, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Pennsylvania 19104-6389, USA.
DOI: http://dx.doi.org/10.1177/02783649922066376.

[Harmonic Drive AG, 2005] Harmonic Drive AG (2005). Datasheet: *CPU-S*.
http://www.harmonicdrive-ag.de/contenido/cms/upload/pdf/GK2007/english/2_Units/3_CPU-S/CPU-S_Technical_Data_en.pdf
04/30-2009.

[HONDA, 2000] HONDA (2000). *ASIMO*.
http://world.honda.com/ASIMO/.

[Huang et al., 2001] Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Arai, H., Koyachi, N., and Tanie, K. (2001). Planning walking patterns for a biped robot. *IEEE Transactions on Robotics and Automation*, 17(3):280–289.
http://ieeexplore.ieee.org/xpls/abs_all.jsp?tp=&arnumber=938385.

[Inman et al., 1981] Inman, V. T., Ralston, H. J., and Todd, F. (1981). *Human Walking*. Wiliams & Wilkins.
ISBN: 0-683-04348-x.

[Jensen et al., 2008] Jensen, P. K., Garbus, M., and Knudsen, J. V. (2008). *Instrumentation, Modeling and Control of AAU-BOT1*. Master's thesis, Aalborg University.
Project group: 08gr1032b
http://projekter.aau.dk/projekter/research/instrumentation_modeling_and_control_of_aaubot1(14466612)/.

[Julier and Uhlmann, 1997] Julier, S. J. and Uhlmann, J. K. (1997). A new extension of the kalman filter to nonlinear systems. In *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL, USA*, pages 182–193.

[Kajita et al., 2001] Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., and Hirukawa, H. (2001). *The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation*. In *International Conference on Intelligent Robots and Systems*, pages 239–246, Maui, Hawaii, USA.
http://staff.aist.go.jp/k.kaneko/publications/2001_publications/5.pdf.

[Knudsen, 2004] Knudsen, M. (2004). Experimental modelling of dynamic systems.
http://www.control.auc.dk/~mk/ExpMod/Publicdoc/LectureNote02pdf.pdf
17/02-2009.

[Lorenz Messtechnik GmbH, 2008] Lorenz Messtechnik GmbH (2008). Datasheet: *SI-RS485*.
http://www.lorenz-messtechnik.de/pdfdatbl/elektron/deutsch/080612b_SI_RS485.pdf
09/25-2008.

[Löffler et al., 2004] Löffler, K., Gienger, M., Pfeiffer, F., and Ulbrich, H. (2004). Sensors and control concept of a biped robot. *IEEE Transactions on Industrial Electronics*, 51(5):972–980.
http://ieeexplore.ieee.org/xpls/abs_all.jsp?tp=&arnumber=1339468.

[maxon motor, 2006] maxon motor (2006). *EPOS Communication Guide*.
http://www.maxonjapan.co.jp/manual/epos/EPOS-CommunicationGuide-E.pdf
11/03-2008.

[maxon motor, 2008a] maxon motor (2008a). Datasheet: *Encoder MR, Type L, 256 - 1024 CPT, 3 Channels, with Line Driver*.
https://shop.maxonmotor.com/maxon/assets_external/Katalog_neu/Downloads/Katalog_PDF/maxon-tacho/Encoder%20MR/ENC-MR-256-1024_225783_08_259_e.pdf
09/25-2008.

[maxon motor, 2008b] maxon motor (2008b). Datasheet: *maxon DC motor and maxon EC motor, Key information*.
http://shop.maxonmotor.com/maxon/assets_external/Katalog_neu/Downloads/allgemeine_informationen/Das_wichtigste_ueber_maxon_motoren/wichgtiges-dc-ec-motoren_08_EN_036-43.pdf
05/13-2009.

[Meriam and Kraige, 1993] Meriam, J. and Kraige, L. (1993). *Engineering Mechanics - Statics*. John Wiley & Sons, Inc., 3. edition.
ISBN: 0-471-59272-2.

[Pedersen et al., 2007] Pedersen, M. M., Nielsen, A. A., and Christensen, L. F. (2007). *Design of Biped Robot AAU-BOT1*. Master's thesis, Aalborg University.
Master Thesis - Project group: DMS10, Group 43A, Pon103
http://projekter.aau.dk/projekter/research/design_of_biped_robot_aaubot1(9778651)/.

[Siciliano and Khatib, 2008] Siciliano, B. and Khatib, O. (2008). *Springer Handbook of Robotics*. Springer-Verlag Berlin Heidelberg, 1. edition.
ISBN: 978-3-540-23957-4.

[Takahashi and Kawamura, 2002] Takahashi, T. and Kawamura, A. (2002). *Posture Control using Foot Toe and Sole Biped walking Robot "'Ken"'*. In *Advanced Motion Control, 2002. 7th International Workshop on*, pages 437–442, Department of Electrical & Computer Enginering, Yokohama National University, Japan.

[Takanishi Laboratory, Waseda University, 2006] Takanishi Laboratory, Waseda University (2006). *WABIAN-2R*.
http://www.takanishi.mech.waseda.ac.jp/top/research/wabian/index.htm.

[Technische Universität München, 1998] Technische Universität München (1998). *Johnnie*.
http://www.amm.mw.tum.de/index.php?id=182&L=1.

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. The MIT Press, 1. edition.
ISBN: 978-0262201629.

[Vaughan et al., 1992] Vaughan, C. L., Davis, B. L., and O'Conner, J. C. (1992). *Dynamics of Human Gait*. Kiboho Publishers, 2. edition.
ISBN: 0-620-23558-6.

[Vukobratović et al., 2007] Vukobratović, M., Borovac, B., and Potkonjak, V. (2007). *Towards a unified understanding of basic notions and terms in humanoid robotics*. In *Robotica*, pages 87–101, Cambridge University Press.
http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=649728
10/9-2008.

[Westervelt et al., 2007] Westervelt, E. R., Grizzle, J. W., Chevallereau, C., Choi, J. H., and Morris, B. (2007). *Feedback Control of Dynamical Bipedal Robot Locomotion*. Taylor & Francis/CRC, 1. edition.
ISBN: 1-42005-372-8.

[Wollherr, 2005] Wollherr, D. (2005). *Design and Control Aspects of Humanoid Walking Robots*. Dissertation, Technische Universität München, München.
http://www.lsr.ei.tum.de/fileadmin/publications/Wollherr2005_Diss.pdf.

[xsens, 2006] xsens (2006). *MTi and MTx Low-Level Communication.*

Contents of Attached DVD

The attached DVD contains additional material that has been used or made in the project. The content of the DVD is listed below:

- Bibliography with internet sources as PDFs.
- Report in digital form.
- Design and simulation of trajectory.
- Simulink model of the system with the designed control system.
- Results of the system test.
- Results of the network throughput test.
- Results of the FTS calibrations.
- Results of the friction coefficients estimation.
- The developed source code for **AAU-BOT1**.