Control of AAU-BOT1

08gr1032b

Printed June 12, 2008

AALBORG UNIVERSITET

Department of Control Engineering
Fredrik Bajers Vej 7C
Telefon 96 35 87 00
Fax 98 15 17 39
http://www.control.aau.dk

Title:
Instrumentation, Modeling and Control of AAU-BOT1

Project period:
September 2007 - June 2008

Project group:
08gr1032b

Members:
Per Kingo Jensen
Mathias Garbus
Jan Vestergaard Knudsen

Supervisors:
Jakob Stoustrup
Jan Helbo
Mads Sølver Svendsen

Copies: 7

Pages in master thesis: 161

Appendices: 13

Printed: June 12, 2008

Synopsis:

The aim of this master's thesis is to equip the humanoid robot **AAU-BOT1** with sensors, model and control it such that it can obtain static gait. **AAU-BOT1** has human proportions and features 17 actuated degrees of freedom.
To enable **AAU-BOT1** to obtain static gait, an instrumentation strategy has been proposed and implemented. Furthermore a software platform is developed to complete the instrumentation.
Models of the DC-motors, kinematics, inverse kinematics and the dynamics of **AAU-BOT1** have been made. By utilizing the inverse kinematic model, static gait trajectories are developed.
The remaining models are utilized to create two different control strategies. The first control strategy is based on a Linear Quadratic Gaussian controller(LQG), which controls the posture of the robot based on the dynamic model. The second control strategy is based on classical PID controllers, and utilizes the build in features of the digital DC motor amplifiers. Both control strategies contains a balance controller, that is used to maintain stability during walk.
It is furthermore decided to develop a virtual representation of the **AAU-BOT1** for test purposes. The LQG controller strategy with the proposed trajectories was tested on the virtual **AAU-BOT1** but the controller could not stabilize the robot sufficiently. The second control strategy was successful and the virtual robot is able to walk with the proposed trajectories and maintain stability at the same time. The second control strategy was only partially implemented on the physical **AAU-BOT1** and showed promising results of obtaining static walk.

Preface

This thesis has been composed during two semesters from the first of September 2007 to the 12th of June 2008 at the Department of Electronic Systems at the Section for Automation and Control, under the Master program Intelligent Autonomous Systems.

In order to obtain inspiration for this thesis, the group was on at study trip to Massachusetts Institute of Technology (MIT) in Boston, Northeastern University in Boston and visited Boston Dynamics, who also develop autonomous robots.

For all the simulation and verification and developing of **AAU-BOT1** MATLAB^{TM} has been widely used. The MATLAB^{TM}version used is 7.3 R2006b with a Simulink version 6.5 R2006b. During the this thesis a program called Webots is used to simulate **AAU-BOT1**. The program version is Webots Pro 5.9.0.

For citations an adapted Harvard method is used, these citations are made in square brackets and contain the author of the literature, DOI-reference and the year of publication. An example of this can be seen here [Craig, 2005]. Furthermore web-pages and conversations with professors will also be referred to.

In the start of each chapter, a short description of the chapter is given. The description is formated with italic. E.g. *'In this preface...'*. At the end of each chapter a summary is given.

The thesis also has a nomenclature and an index. A list of acronyms can be found in Appendix L on page 229. On the last page, a CD-ROM is enclosed, which contains literature, model files, figures and drawings used for the thesis. A complete description of the contents on the CD-ROM can be found in Appendix M on page 231.

Per Kingo Jensen

Jan Vestergaard Knudsen

Mathias Ramskov Garbus

GROUP 08GR1032B

Contents

Nomenclature **9**

1 Introduction **13**
1.1 Background information . . . 13
1.2 Existing Biped Robots . . . 13
1.3 Walking Robots at Aalborg University . . . 15
1.4 Objectives . . . 17
1.5 Thesis Outline . . . 18

2 Humanoid Robotics Definitions **23**
2.1 Coordinate system . . . 23
2.2 Definitions used in this report . . . 23

3 Instrumentation and Network Design **31**
3.1 **AAU-BOT1** Description . . . 31
3.2 Network Design . . . 33
3.3 Actuators . . . 35
3.4 EPOS Amplifiers . . . 37
3.5 Absolute Joint Angle Measurements . . . 37
3.6 Force Torque Sensor . . . 38
3.7 On-board Computer . . . 43
3.8 Inertia Measurement Unit . . . 45
3.9 Summary of Instrumentation and Network Design . . . 46

4 Software Architecture **49**
4.1 General Software Description . . . 49
4.2 Simulink S-function Interface . . . 50
4.3 Shared Memory Server . . . 54
4.4 Sensor Servers . . . 54
4.5 Actuator Server . . . 57
4.6 EPOS/CAN Driver . . . 57
4.7 FTS driver . . . 60
4.8 Visulisation . . . 65
4.9 Summary of Software . . . 68

5 Modeling **69**
5.1 Introduction to Modeling . . . 69
5.2 Elements in the Model . . . 69
5.3 DC Motor Model . . . 71
5.4 Kinematic Model . . . 77
5.5 Dynamic Model . . . 83
5.6 Support Phase Estimator . . . 89
5.7 Inverse Kinematics . . . 91
5.8 Summary of Modeling . . . 96

6 Trajectory generation **97**
6.1 Trajectory Generation Requirements . . . 97
6.2 Different Trajectory Generation Approaches . . . 99
6.3 Establishing Trajectories for **AAU-BOT1** . . . 102
6.4 Simulation and Results of Trajectory Generation . . . 112
6.5 Summary of Trajectory Generation . . . 116

7 Control **119**
7.1 Controller Structure . . . 119
7.2 Control Strategy A . . . 121
7.3 Control Strategy B . . . 128
7.4 Observers . . . 139
7.5 Supervisor . . . 140
7.6 Summary of Control . . . 141

8 System Test **143**
8.1 Introduction to Complete Test . . . 143
8.2 Virtual Robot in Webots . . . 143
8.3 Actual **AAU-BOT1** . . . 145
8.4 Summary of System Test . . . 147

9 Epilogue **149**
9.1 Discussion . . . 149
9.2 Conclusion . . . 154
9.3 Future Work . . . 155

Bibliography **157**

A Verification of Models **163**
A.1 DC Motor Model . . . 163
A.2 Verification of Kinematic Model . . . 166
A.3 Verification of Inverse Kinematic Model . . . 169

B Verification and Implementation of Controllers **175**
B.1 Verification of Control Strategy A . . . 175
B.2 Verification of Control Strategy B . . . 180

C Mechanical Data **183**

D Foot Model **188**
D.1 Foot Design Overview . . . 188
D.2 Forces and Torques on the Foot . . . 189
D.3 Constraints . . . 191

E Motivating Example **193**
E.1 Kinematic Model . . . 193
E.2 Dynamics in SSP . . . 196
E.3 Dynamics of Strider in DSP . . . 200

F Dynamic Gait Trajectories **203**
F.1 Foot Trajectory for Dynamic Gait . . . 203
F.2 Torso Trajectory for Dynamic Gait . . . 205

G Alternative FTS DAQ **209**
G.1 Analog FTS DAQ . . . 209
G.2 Alternative Digital FTS DAQ . . . 213

H Calibration and Test of the FTS's and Amplifiers **215**
H.1 Calibration Test Method . . . 215
H.2 Results of Calibration . . . 217
H.3 Future Work . . . 220

I Throughput Test **221**
I.1 Method . . . 221
I.2 Result . . . 221
I.3 Discussion . . . 221
I.4 Summary . . . 222

J CAN Frame Overview **223**

K Node Overview **227**

L List of Acronyms **229**

M Contents of the enclosed CD **231**

Index **233**

Nomenclature

$\ddot{\theta}_M$	Angular acceleration of the DC motor shaft $\left[\frac{\text{rad}}{\text{s}^2}\right]$, page 71
$\Delta\mathbf{L}$	Penetration depth of the spring in the heel [m], page 190
$\Delta\mathbf{L}_{\max}$	Maximum penetration depth of the spring in the heel[m], page 190
$\mathcal{L}$	Lagrangian matrix, page 197
$\boldsymbol{J}_F(\vec{\theta})$	The Jacobian matrix of the system, with regards to $\vec{\theta}$, page 194
$\boldsymbol{J}_n$	The inertia tensor of link n, page 30
μ	Viscous friction coefficient $\left[\frac{\text{Nm s}}{\text{rad}}\right]$, page 72
$\omega_{x,n}$	The angular velocity around the x axis of link n $\left[\frac{\text{rad}}{\text{s}}\right]$, page 30
$\omega_{y,n}$	The angular velocity around the y axis of link n $\left[\frac{\text{rad}}{\text{s}}\right]$, page 30
$\omega_{z,n}$	The angular velocity around the z axis of link n $\left[\frac{\text{rad}}{\text{s}}\right]$, page 30
τ_c	Coloumb friction torque [N m], page 72
τ_F	Friction torque [Nm], page 71
τ_L	Load torque [Nm], page 71
τ_M	Motor torque [Nm], page 71
τ_s	Stiction torque [N m], page 72
θ_M	Angle of the shaft [rad], page 71
θ_n	Angle of link n [rad], page 71
$\vec{P}_{\text{CoM}}$	Position vector of the CoM, page 29
$\vec{P}_n$	Position vector of the center of mass of link n, page 29
$\vec{\mathcal{F}}$	External force vector, page 197
$\vec{\omega}_n$	The angular velocity vector of link n, page 30
$\vec{a}_n$	Distance vector from joint $n-1$ to joint n, page 194

$\vec{b}_n$ Distance vector from joint $n-1$ to CoM of link n, page 194

$\vec{g}$ Gravitational acceleration constant vector, page 30

$\vec{M}$ The total moment vector, page 30

$\vec{P}_{\text{GCoM}}$ The position vector of the GCoM, page 29

$\vec{P}_{\text{ZMP}}$ Position of ZMP, page 89

$\vec{P}_L$ Position of the left foot, page 89

$\vec{P}_R$ Position of the right foot, page 89

$\vec{q}$ State vector, page 197

${}^m_n\boldsymbol{R}$ The rotation matrix from frame m to frame n., page 194

c_n The cosine to angle n [ ], page 194

c_r The rotational dampening coefficients $[\frac{\text{Nm s}}{\text{rad}}]$, page 190

c_t The translational dampening coefficients $[\frac{\text{N s}}{\text{m}}]$, page 190

$d_{i,stabMarg}$ Stability margin, positive if the ZMP is within the support area, page 104

E_{kin} Kinetic energy [J], page 197

E_{pot} Potential energy [J], page 197

G Gear ratio [ ], page 71

h_{amax} Maximum ankle height during T_{step}, page 109

h_{tmax} Maximum height of the torso., page 110

h_{tmin} Minimum height of the torso. , page 110

i_M Motor current [A], page 71

J Total inertia of both the motor and the load [kg cm^2], page 71

$J_{x,n}$ The inertia of link n around the x axis[kg cm^2], page 30

$J_{y,n}$ The inertia of link n around the y axis[kg cm^2], page 30

$J_{z,n}$ The inertia of link n around the z axis[kg cm^2], page 30

K_{emf} Motor back emf voltage constant $\left[\frac{\text{Vs}}{\text{rad}}\right]$, which has the same value as K_T, page 71

k_r The rotational spring coefficients $[\frac{\text{Nm}}{\text{rad}}]$, page 190

K_T Motor torque constant $\left[\frac{\text{Nm}}{\text{A}}\right]$, which has the same value as K_{emf}, page 71

k_t The translational spring coefficients $[\frac{\text{N}}{\text{m}}]$, page 190

L_M Terminal inductance [H], page 71

l_{an} Height of the ankle from the ground., page 109

l_{at} Horizontal distance between the ankle and toe., page 109

m_n Mass of link n [kg], page 29

M_x Moment around the x-axis [Nm], page 29

M_y Moment around the y-axis [Nm], page 29

M_z Moment around the z-axis [Nm], page 29

m_{Tot} The **AAU-BOT1**'s total weight [kg], page 190

N_{Joints} Number of joints[], page 86

N_{Links} Number of links[], page 30

R_M Terminal resistance [Ω], page 71

s_n The sine to angle n [ ], page 194

$stabIndex$ Array of stability indexes for all simulations., page 105

$stabIndex_{max}$ Maximal number of stable ZMP's is all simulations., page 105

T_{max} Time it takes to raise the ankle to max height h_{amax}., page 107

T_{sim} Simulation time., page 106

T_{SSP} Time **AAU-BOT1** is in SSP during T_{step}., page 107

T_{step} Time it takes to move right foot in front of the left., page 107

u Input voltage [V], page 71

x_{ZMP} The x-coordinate of the ZMP vector ($\vec{P}_{\text{ZMP}}$) [m], page 30

$x_a(t)$ Horizontal movement of the ankle during T_{step}., page 109

x_n x-coordinate of the center of mass of link n[m], page 30

$x_t(t)$ Movement of the torso in the x-direction during T_{step}., page 110

x_{tmax} Length in the x-axis from the torso to the right foot., page 110

y_{ZMP} The y-coordinate of the ZMP vector ($\vec{P}_{\text{ZMP}}$) [m], page 30

y_n y-coordinate of the center of mass of link n[m], page 30

$y_t(t)$ Movement of the torso in the y-direction during T_{step}., page 111

y_{mid} Distance from center of the pelvis to one hip., page 111

y_{tmin} Length in the y-axis from the torso to the right foot., page 111

$z_a(t)$ Vertical movement of the ankle during T_{step}., page 109

z_n z-coordinate of the center of mass of link n[m], page 30

$z_t(t)$ Vertical movement of the torso during T_{step}., page 110

Chapter 1

Introduction

In this chapter, the background for the project is given. This includes a summary of some existing biped robots in the world Hereafter the robots at Aalborg University will be elaborated on. Finally the objectives and a problem formulation of the master's thesis is stated and an overview of the project is given.

1.1 Background information

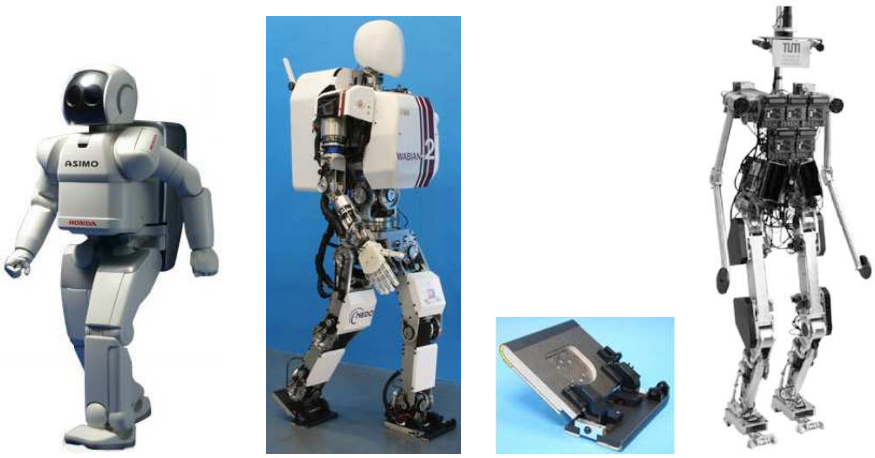
Biped robots have been widely studied by science departments at Universities and the industry for many years. The purpose is to continuously improve the intelligence of the robots and making them faster, more stable and stronger. This is done so the robots can perform heavy or dangerous work previously imposed on humans. Other sections of the industry such as the medical industry also use robots. This section do a great deal of work on artificial limbs for war veterans with missing limbs. The limping robots are examined in order to fully understand which muscles are affected and how, to improve the manufacturing of artificial limbs. These robots could also be used for rehabilitation and thereby help people with gait disorder with the daily training.

1.2 Existing Biped Robots

In this section three advanced biped robots are presented. The three robots are AISMO, WABIAN-2R and Johnnie, see Figure 1.1 on the following page. Since Honda is keeping AISMO confidential and covering it in plastic shields, then the specifications is unavailable. Therefore only WABIAN-2R and Johnnie is described in the following. Both robots are using servo motors for actuation of manipulators, and have force-torque sensors in the ankles. The features together with the performance and certain soft- and hardware aspects will be discussed. The scope of this section is to study the specifications of existing robots, to achieve inspiration for a platform design for a walking robot. The following are based on [Ogura et al., 2006] and [Löffler et al., 2004a].

WABIAN-2R

Waseda University, started researching biped robots in 1966, and presented their first humanoid robot in 1973. Since then, a number of biped robots have been developed, latest the WABIAN-2R in 2006, see Figure 1.1(b) on the next page. This biped robot



(a) ASIMO (b) WABIAN-2R (c) Foot mechanism (WABIAN-2R) (d) Johnnie

Figure 1.1: Existing walking robots.

has 41 Degrees of Freedom (DoF) and with this relatively large number of DoF's the WABIAN-2R support a relatively smooth human-like walk. In Figure 1.2 shows a sketch of the system structure of WABIAN-2R.

WABIAN-2R is controlled by a computer mounted in its torso. The on-board computer consists of a PCI CPU board, this is connected to I/O-boards through the PCI bus. This I/O-board has 16ch D/A, 16ch Counters and 16ch PIO, and six axis force/torque sensor receiver board. The operating system used with the on-board computer is QNX, which is a real time system. Every actuator is equipped with an incremental encoder attached to the motor shaft. To detect the initial posture a photo sensor is attached to each joint shaft. Each ankle is equipped with a six axis force-torque sensor which is used for measuring floor reaction force. Furthermore WABIAN-2R is equipped with a toe feature as seen in Figure 1.1(c), this feature can be helpful when human-like walk is the objective.

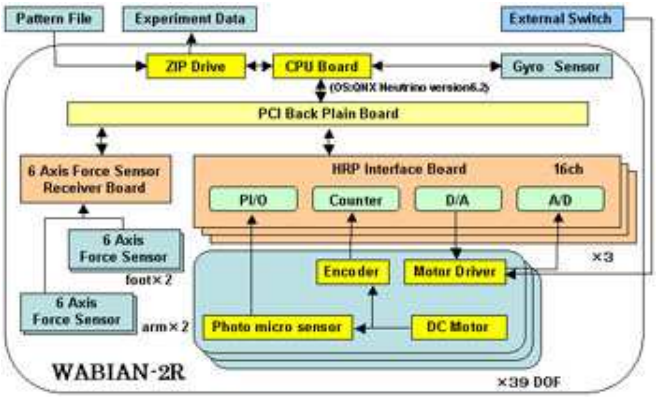


Figure 1.2: System structure of WABIAN-2R [Ogura et al., 2006].

Johnnie

Technical University of Munich initiated research in bipedal walking with the robot Johnnie, as seen in Figure 1.1(d) on the facing page. Johnnie was built for the German Research Foundation in the Priority Program Autonomous Walking. This started in 1998, with the intention to create a human-like stable gait for a humanoid robot. Johnnie can pass obstacles in its path, this is achieved by a visual guiding system, based on a stereo camera system. Johnnie is an european built biped robot, whereas WABIAN-2R is made in Japan. Johnnie features 17 degrees of freedom, which are mainly located in the lower body region. The control computer used for Johnnie is a 2.8 GHz computer which runs under RTAI-Linux. The computed data are sent to 17 decentralized micro controller which drives the power amplifiers and reads sensor data. All this is achieved with an overall sampling time of 4 ms (250 Hz). Figure 1.3 is a sketch of the control structure on Johnnie.

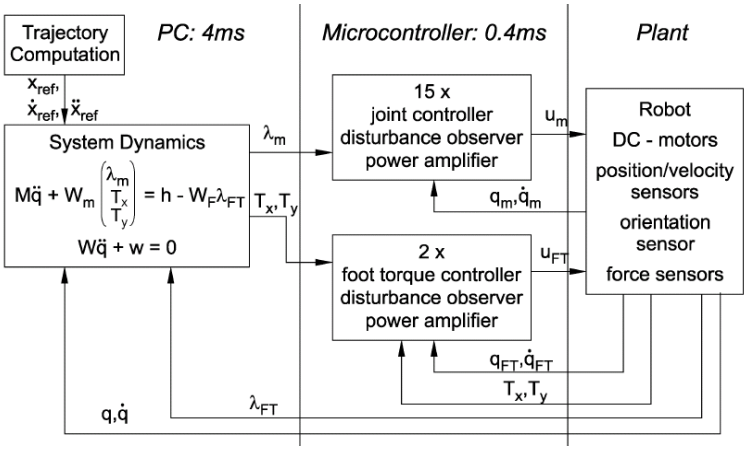


Figure 1.3: The control structure of Johnnie [Löffler et al., 2004a].

Johnnie and WABIAN-2R are two sophisticated robots that both have similarities like the six axis force torque sensor. Where WABIAN-2R differs from Johnnie is the toe feature and on-board computer and seems more likely to success in obtaining the most human-like gait.

1.3 Walking Robots at Aalborg University

The biped robot research at Aalborg University was initiated three years ago. This was done to clarify the opportunity to design a biped robot with human proportions. This began with the research of two smaller robots and should form basis for the development of a human sized robot. In the following a summation of the different robots at AAU is elaborated.

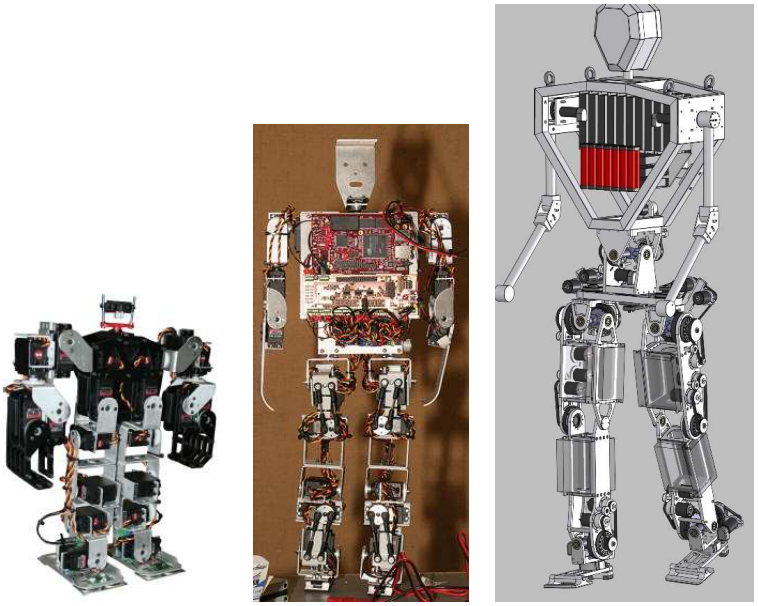
The First Small Biped Robot (The Sergeant)

The biped robotics research at AAU started with the purchase of the small biped robot [Christensen et al., 2006], currently called the Sergeant. The Sergeant is a commercially made robot, which is capable of stable static gait, by means of fuzzy logic.

The design of the Sergeant is unfortunately too slow to enable human-like gait. In Figure 1.4(a) picture of the Sergeant.

The Second Biped Robot (Roberto)

Roberto was designed and constructed during a master's thesis started in September 2006 and ended June 2007. The work is described in the master's thesis [Christensen et al., 2007] which was conducted at the section of Automation and Control at Aalborg University. The design philosophy was to create a humanoid robot, scaled down to a height of 58 cm, and with 21 actuated degrees of freedom. The group were unfortunately unable to make Roberto walk with a human-like walk due to system limitations. These limitations are conceived to be interface to the servo-motors and a slow on-board computer. See Figure 1.4(b) for a picture of Roberto.



(a) The Sergeant (30cm, 21 DoF)

(b) Roberto (58 cm, 21 DoF)

(c) **AAU-BOT1** (180cm, 19 DoF where 2 of the DoF are spring actuated)

Figure 1.4: The biped robots at Aalborg University [Helbo, 2008]. Not to scale.

The Third Biped Robot (AAU-BOT1)

The newest biped robot at Aalborg University is **AAU-BOT1** which is in human size. This robot is proposed to help closing the gap between the fields of Health technology and Robotics. It was funded by the Dannin foundation in 2006, where a **AAU-BOT1** research project was initiated. The research project is scheduled to finish in 2010. During this time a mechanical design is developed and the **AAU-BOT1** has to manufactured. This is done by a group at the Department of Mechanical Engineering at Aalborg University in the time period September 2006 to June 2007. This process is described in [Pedersen et al., 2007]. In the time period from September 2007 to June 2010. The remaining time is used for instrumentation of the robot, development of a proper control

solution. Finally the **AAU-BOT1** has to obtain human-like gait. See Figure 1.4(c) on the facing page to see a SOLIDWORKS drawing of **AAU-BOT1**. **AAU-BOT1** is designed to be in human-like proportions (180 cm tall).

The 30. of November **AAU-BOT1** is handed over to the Section of Automation and Control and the instrumentation and initial implementation of motion trajectories, are parts of the challenges that will be elaborated on in this thesis. It is important the strategies developed in this master's thesis can be used to achieving the main goal of obtaining human-like gait.

Furthermore **AAU-BOT1** at Aalborg University forms the basis for further research and development of other and even more sophisticated and improved walking robots. Since the development of **AAU-BOT1** is done in individual stages in different institutes of Aalborg University. This thesis will depend on the quality of the work done at the Institute of Mechanical Engineering. This also applies the other way around when the Institute of Mechanical Engineering will be making the next version of **AAU-BOT1** (AAU-BOT2).

1.4 Objectives

Former reports dealing with humanoid robots have been conducted at the Section of Automation and Control at Aalborg University. In one case only has it been possible to achieve static gait but not human-like walk. As far as the project group is aware of, no robot have ever obtained complete human-like gait. The main objective of the 2010 research project is to obtain human-like gait. This master's thesis will help this main, by taking the first step of instrumentation and modeling and control of the **AAU-BOT1**. Thus, the objectives of this project are:

- Completion of hardware platform.
- Modeling of **AAU-BOT1**.
- Design a controller that enables static gait on **AAU-BOT1**.
- Design a controller that enables dynamic gait on **AAU-BOT1**.

Problem Formulation

With the background information given in the previous sections, the problem investigated is:

*Is it possible to obtain static gait with **AAU-BOT1**?*

This will be achieved in stages. The first stage is to develop the model in Simulink such that controllers can be designed, implemented and simulated with static gait. The second stage is instrumenting the **AAU-BOT1** with the necessary hardware, such that the controllers can be tested on the real system. The third stage is to implement static-like gait in simulation and on the real system.

Delimitation

The delimitations of the master's thesis are described in this section. The delimitations are described below:

- The final product has to include batteries before a completely autonomous system can be obtained. It is decided to use a power supply instead of batteries. This is done since the project already is very hardware heavy compared to the scope of the 9th and 10th semester.
- As it is chosen to investigate if static walk are obtainable, the toe off and heel strike motions are not implemented on **AAU-BOT1**. The toe off feature are however included in the development such that it can be utilized in the future.
- In order to design a total instrumentation strategy, an Inertial Measurement Unit (IMU) is implemented in the design phase. However the IMU is included in this master's thesis, but it is not utilized, as the work load has been extensive and the IMU is therefore a subject for future groups working with **AAU-BOT1**.

1.5 Thesis Outline

This thesis is a documentation of the instrumentation, the modeling and the control of the humanoid robot **AAU-BOT1**. This includes the development of the necessary hardware and software needed to enable the system to perform human-like gait. In Figure 1.5 on the next page an illustration shows the different parts of the project. The work flow are not indicated on the figures as an iterative development approach is used.

In Figure 1.5 the different element of the project is shown, these are explained in the following:

- **Analysis of gait and its phases:**
  This part clarifies the different phases the system can obtain during walk. Furthermore it describes walking cycles for human walk.
- **Analysis of conceptual robotics theory:**
  This part gives the conceptual knowledge of the terms and notations used in the thesis.
- **Kinematic models:**
  These models describes the position, orientation, velocity and acceleration of the links and joints of **AAU-BOT1**.
- **Dynamical models:**
  This part describes the dynamics of **AAU-BOT1** in the different phases.
- **Phase Estimator:**
  This estimator supervises the system at all times and determines which phase **AAU-BOT1** is in during walking. This is needed since the dynamical model and the kinematic model change depending on which foot **AAU-BOT1** is standing on.
- **Inverse kinematic model:**
  This model translates the position of an end manipulator to angles of all the joints. This enables the controller to control the posture of **AAU-BOT1**.

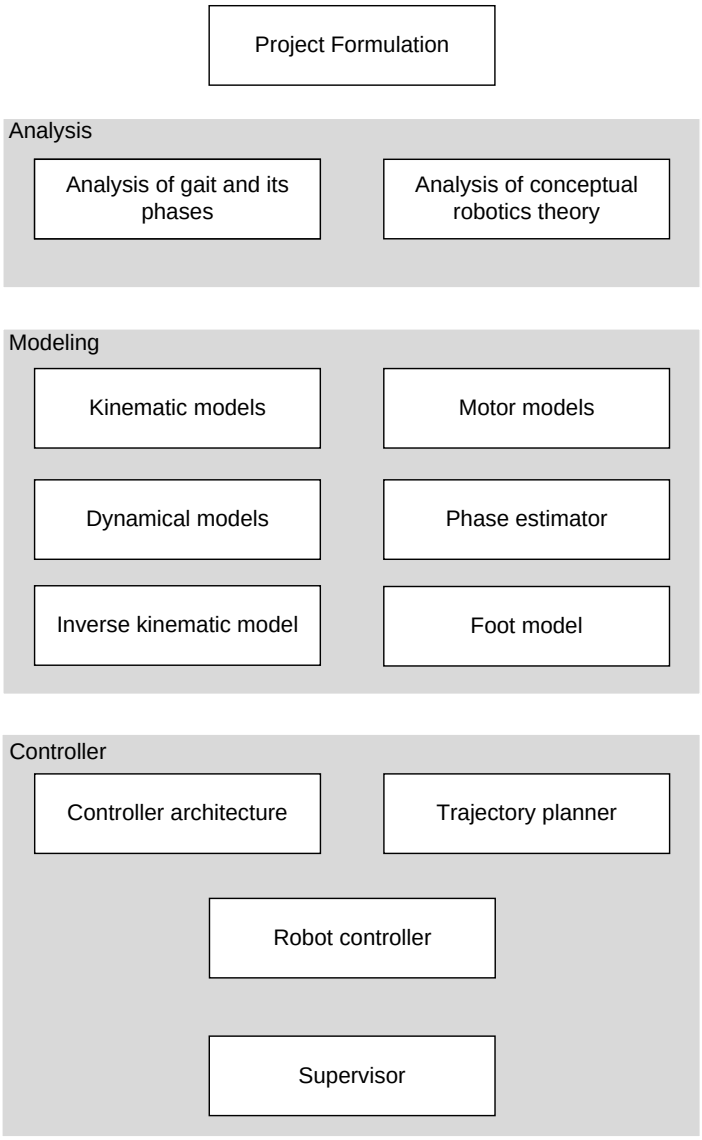


Figure 1.5: Illustration showing the connection between the different parts of the project

- **Motor model:**
  Every actuated joint is powered by one or two motors. Motor dynamics and gear relations is elaborated on in this part.
- **Foot model:** This part describes the foot model, including the unactuated toe feature and heel contact feature.
- **Trajectory planner:**
  This planner describes how all the body parts moves in order to take a step and

stay balanced at the same time.

- **Controller architecture:**
  This section will analyze different control approaches and will give a proposal to a control strategy.

- **Robot controller:**
  The controller are proposed such that the robot can be stabilized and make it capable of tracking the developed trajectories.

Some of the main parts of the thesis is mentioned, but there is still parts of it that have not been elaborated on and this is the platform design. **AAU-BOT1** is from the beginning designed and manufactured, but the control structure and instrumentation are still missing. The only electrical parts implemented are the DC motors for movement of limbs and a number of strain gauges to detect the influence the robot has on each foot. In Figure 1.6 the needed elements for a platform design is shown. The arrows show how the different topics are related.

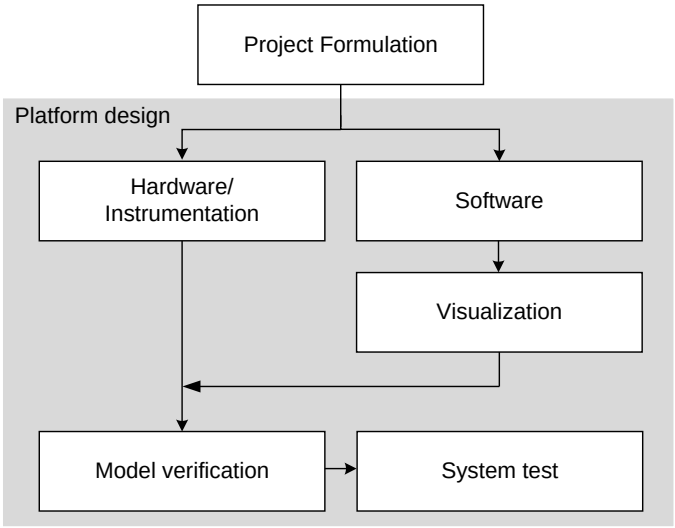


Figure 1.6: Illustration showing the elements needed for the platform design.

These topics are listed below:

- **Instrumentation:**
  This describes the implementation of all the proposed electrical equipment. This is needed to enable control of **AAU-BOT1**.

- **Software:**
  The structure of the software system and how the software has been implemented.

- **Visualization:**
  In order to see the attitude of **AAU-BOT1** while testing controllers and models a robot simulation tool called Webots is used.

- **Model verification:**
  Each model has to be verified in order to see whether it corresponds to the actual system. A verification is done at the end of each model part.

- **System Test:**
  The completed system is tested to examine how well the system performs.

The outline of the master thesis has now been elaborated on. Now the definitions and terms used in the report is explained.

Chapter 2

Definitions and notation in humanoid robotics

This chapter deals with the coordinate system and the different notations and definitions used in this report. I.e. step, walk, gait, human gait, support phases, support area, center of mass, ground projection of center of mass and zero moment point are defined here.

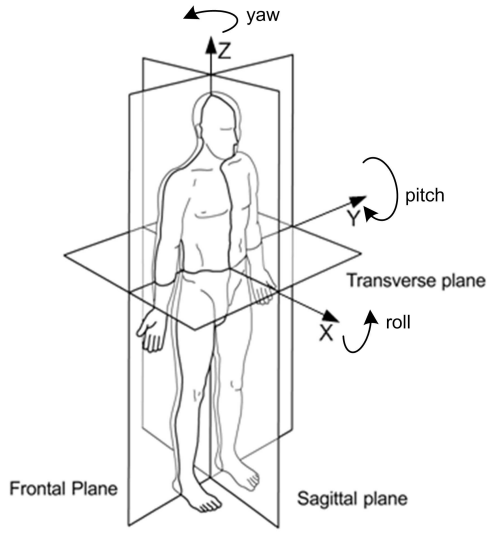
2.1 Description of utilized coordinate system

In this report, a right-handed XYZ cartesian coordinate system is used, (see Figure 2.1). Origo is chosen to be at the center of the toe of the front foot that touches the ground, see Figure 2.2. The origo position is chosen such that the coordinate system is as stationary as possible, and enable the same algorithm to control both sides of the walking cycle, as the balance controller is always to move the balance point towards the front foot.

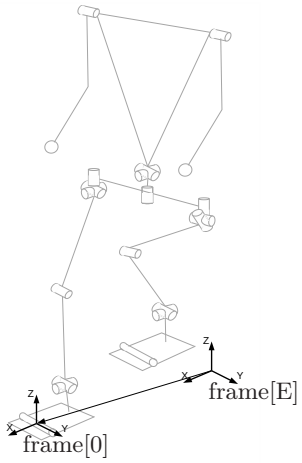
2.2 Definitions used in this report

The notation and definitions in this project are based on the article [Vukobratović et al., 2007], the primary ones are listed here, note that *italic* text is direct citations from the article:

- **Walk** is understood as the *'movement by putting forward each foot in turn, not having both feet off the ground at once'*. Therefore walk is defined as displacement of both legs, but under no circumstances may the feet get separated from the ground at the same time.
- **Gait** is the way one may walk. Every single individual has it own certain characteristics when walking and this certain characteristics is called gait.
- **Step** happens when the rear foot is moved in front of the foremost foot and thereby becoming the front foot. If this motion is repeated a locomotion of the robot will occur. A step consist of at least two phases: *'a single-support phase, when only one foot is in contact with the ground (during this time period the supporting leg from the front position with respect to the trunk comes to the rear position, while the swing leg from the rear position comes to the front position), a and double-support phase in which both feet are simultaneously on the ground.'*



(a) Overview of used planes.[Vaughan et al., 1992]



(b) Position of the Earth frame and the 0 frame. The 0 frame is placed in the center of the toe on the foremost foot that touches the ground.

Figure 2.1: Overview of used planes and coordinate system.

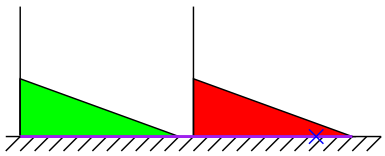
- **Human gait** Traditionally the human gait cycle has been divided into eight events or periods, five during stance phase and three during swing phase. The names of these events are self-descriptive and are based on the movement of the foot, as seen in Figure 2.3 on page 26.

  In the traditional nomenclature, the stance phase events are as follows, as described in [Vaughan et al., 1992].

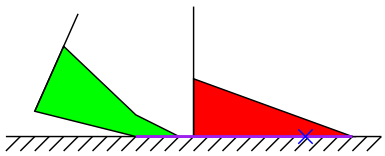
  1. Heel strike initiates the gait cycle and represents the point at which the body's centre of gravity is at its lowest position.
  2. Foot-flat is the time when the plantar surface of the foot touches the ground.
  3. Midstance occurs when the swinging (contralateral) foot passes the stance foot and the body's center of gravity is at its highest position.
  4. Heel-off occurs as the heel loses contact with the ground and push-off is initiated via the triceps surae muscles, which plantar flex the ankle.
  5. Toe-off terminates the stance phase as the foot leaves the ground.

  The swing phase events are as follows:

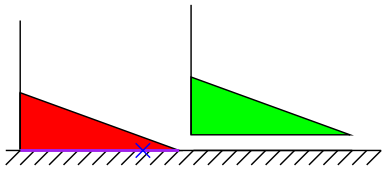
  6. Acceleration begins as soon as the foot leaves the ground and the subject activates the hip flexor muscles to accelerate the leg forward.
  7. Midswing occurs when the foot passes directly beneath the body, coincidental with midstance for the other foot.
  8. Deceleration describes the action of the muscles as they slow the leg and stabilize the foot in preparation for the next heel strike.



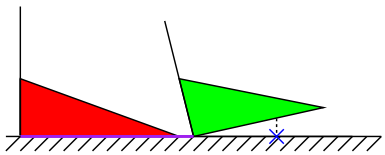
(a) Double Support Phase (DSP-R).



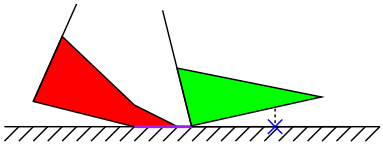
(b) Double Support Phase Right Toe-off (DSP-R-T).



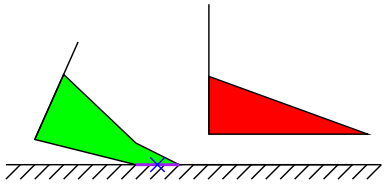
(c) Single Support Phase (SSP-L).



(d) Double Support Phase Left Heel-strike (DSP-L-H).



(e) Double Support Phase Left Toe and Heel-strike (DSP-L-TH).



(f) Single Support Phase Right Toe (SSP-R-T).

Figure 2.2: Overview of placement of origo in different support phases. Origo is marked with a blue x, support area is marked in purple, the left foot is marked in red and the right foot is green.

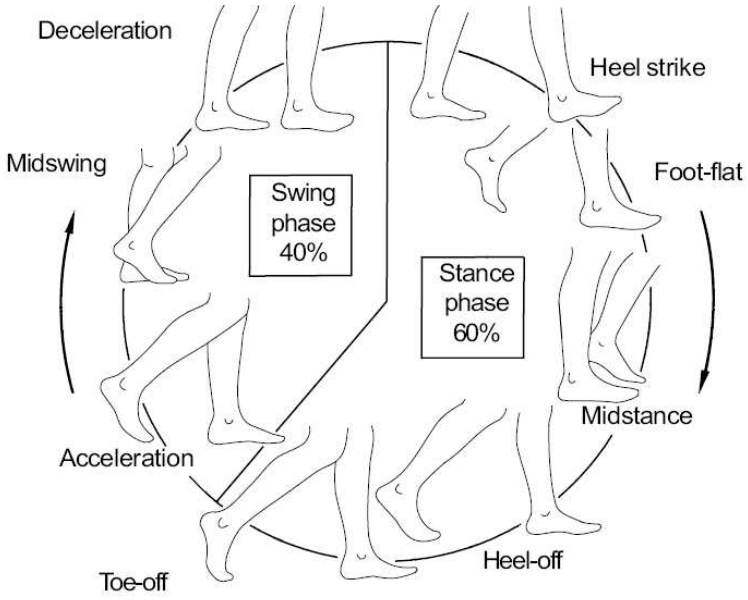
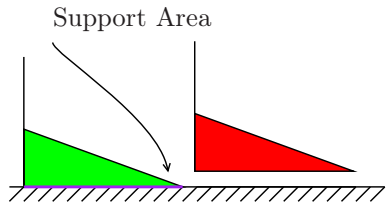
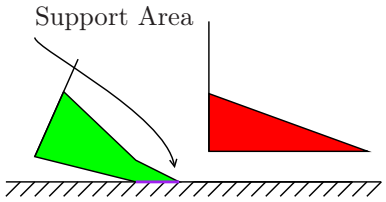


Figure 2.3: Human gait cycle [Vaughan et al., 1992].

- **Support phases** is a way to subdivide a step. The rear foot or the foot on the ground will be indicated with a index, i.e. the phase where **AAU-BOT1** stands on both full feet surfaces with the left foot being the rear foot is named DSP-L. As all the phases can be either a left or a right phase, the L og R is replaced with a X in the description. The support phases can be seen in Figure 2.4 to 2.7.
  - **Single Support Phase (SSP)** occurs when **AAU-BOT1** only has one foot on the ground. SSP has two different manifestations:
    - **Basic Single Support Phase (SSP-X)**, where **AAU-BOT1** is standing on one foot's full surface, see Figure 2.4(a) and 2.6(a). During walking, humans are 80% of the time in SSP-X.
    - **Single Support Phase Left/Right Toe (SSP-X-T)**, where **AAU-BOT1** is standing on a toe, see Figure 2.4(b) and 2.6(b).
  - **Double Support Phase (DSP)**, where **AAU-BOT1** has both feet on the ground. There are a number of special cases of DSP:
    - **Basic Double Support Phase Left/Right (DSP-X)**, where both feet are flat on the ground, see Figure 2.5(a) and 2.7(a).
    - **Double Support Phase Left/Right - Heel-strike (DSP-X-H)**, where the rear foot is flat on the ground, and the front foot's heel strikes the walking surface, see Figure 2.5(b) and 2.7(b).
    - **Double Support Phase Left/Right Toe and Heel (DSP-X-TH)**, where the rear foot is standing on the toe, and the front foot's heel strikes the walking surface, see Figure 2.5(c) and 2.7(c).
    - **Double Support Phase Left/Right Toe (DSP-X-T)**, where the rear foot is standing on the toe and the front foot is flat on the ground, see Figure 2.5(d) and 2.7(d).

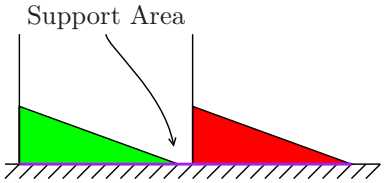


(a) Single Support Phase Right (SSP-R).

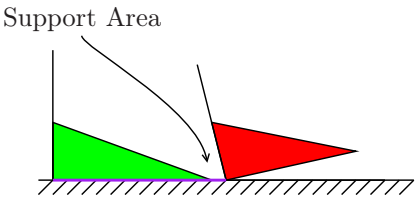


(b) Single Support Phase Right Toe (SSP-R-T).

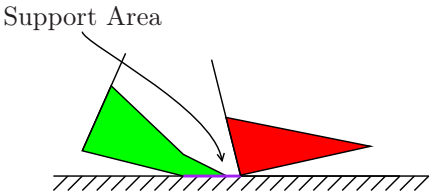
Figure 2.4: Saggital view of the different single support phases.



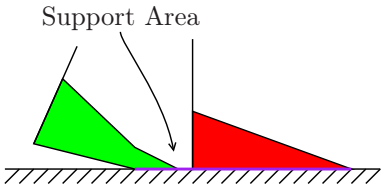
(a) Double Support Phase (DSP-R).



(b) Double Support Phase Right Heel-strike (DSP-R-H).



(c) Double Support Phase Right Toe and Heel-strike (DSP-R-TH).



(d) Double Support Phase Right Toe (DSP-R-T).

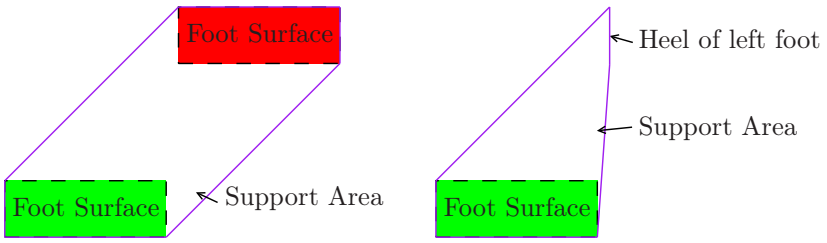
Figure 2.5: Saggital view of the different double support phases.

- **Support Area (SA)** is the convex hull of all contact points between **AAU-BOT1** and the ground. If both feet are off the ground no support area exists. In regular gait a support area always exist: *'In the single-support phase the support area coincides with the area of the foot in contact with the ground, whereas in the double-support phase, the support area is a convex area determined by the area of the feet and the ground and common tangents, so that the encompassed area is maximized'* This means that if the robot has squared feet the support area in the double-support phase make up a polygon. The support area during the support phases are illustrated in Figure 2.6 and 2.7, marked in purple.

(a) Single Support Phase Right (SSP-R).

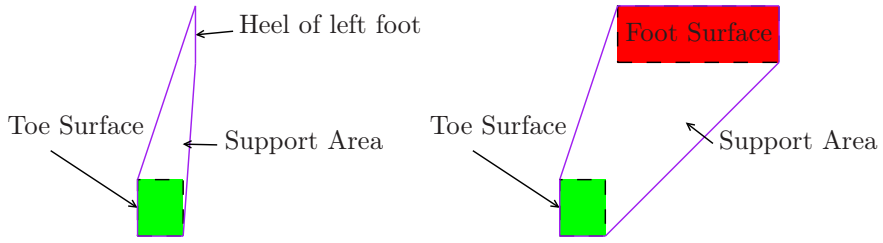
(b) Single Support Phase Right Toe (SSP-R-T).

Figure 2.6: Overview of support area in the different single support phases.



(a) Double Support Phase Right (DSP-R).

(b) Double Support Phase Right Heel-strike (DSP-R-H).



(c) Double Support Phase Right Toe and Heel-strike (DSP-R-TH).

(d) Double Support Phase Right Toe (DSP-R-T).

Figure 2.7: Overview of support area in the different double support phases. **AAU-BOT1** is facing right.

- **Center of Mass (CoM)** is the global position of the center of mass of the body,

relative to the global origo. CoM is calculated by

$$\vec{P}_{\text{CoM}} = \frac{\sum_{n=1}^{N_{\text{Links}}} \vec{P}_n \cdot m_n}{\sum_{n=1}^{N_{\text{Links}}} m_n} \tag{2.1}$$

where:
$\vec{P}_{\text{CoM}}$ is the position vector of the CoM
$\vec{P}_n$ is the position vector of the center of mass of link n
m_n is the mass of link n [kg]
N_{Links} is the number of links [ ]

- **Ground projection of Center of Mass (GCoM)** is the position of the CoM, projected to the ground level.

- **Zero Moment Point (ZMP)** is defined in [Vukobratović et al., 1969], and it is the dynamic equivalent of the GCoM. ZMP is the point on the ground, where the moments around any axis passing through this point and being tangential to the ground, is zero. Mathematically this is expressed as:

$$\sum M_x = 0 \tag{2.2}$$

$$\sum M_y = 0 \tag{2.3}$$

where:
M_x is the moment around the x-axis [Nm]
M_y is the moment around the y-axis [Nm]

Equation (2.2) and (2.3) can be expanded to the following:

$$\sum_{n=1}^{N} (m_n(\vec{P}_n - \vec{P}_{\text{ZMP}}) \times (\ddot{\vec{P}}_n + \vec{g}) + \boldsymbol{J}_n\dot{\vec{\omega}}_n + \vec{\omega}_n \times \boldsymbol{J}_n\vec{\omega}_n) = \vec{M} \tag{2.4}$$

$$\vec{M} \times \vec{g} = 0 \tag{2.5}$$

where:
$\vec{M}$ is the total moment vector
$\vec{g}$ is the gravitational acceleration constant vector
$\vec{\omega}_n$ is the angular velocity vector of link n
$\boldsymbol{J}_n$ is the inertia tensor of link n
$\vec{P}_{\text{ZMP}}$ is the position vector of the ZMP
From [Vukobratović et al., 1969], it is given that Equation (2.4) and (2.5) can be combined into one equation:

$$\sum_{n=1}^{N} (m_n(\vec{P}_n - \vec{P}_{\text{ZMP}}) \times \ddot{\vec{P}}_n + \boldsymbol{J}_n\dot{\vec{\omega}} + \vec{\omega}_n \times \boldsymbol{J}_n\vec{\omega}_n - m_n(\vec{P}_n - \vec{P}_{\text{ZMP}}) \times \vec{g}) = \begin{bmatrix} 0 & 0 & * \end{bmatrix}^{\text{T}} \tag{2.6}$$

The asterisk ($*$) denotes a vector element without significance for the equation. By assuming the ground is transverse, Equation (2.6) can be expressed as [Huang et al.,

2001]:

$$x_{\text{ZMP}} = \frac{\sum_{n=1}^{N_{\text{Links}}} (m_n(\ddot{z}_n + g_z)x_n - m_n\ddot{x}_n z_n - J_{y,n}\dot{\omega}_{y,n})}{\sum_{n=1}^{N_{\text{Links}}} m_n(\ddot{z}_n + g_z)} \tag{2.7}$$

$$y_{\text{ZMP}} = \frac{\sum_{n=1}^{N_{\text{Links}}} (m_n(\ddot{z}_n + g_z)y_n - m_n\ddot{y}_n z_n - J_{x,n}\dot{\omega}_{x,n})}{\sum_{n=1}^{N_{\text{Links}}} m_n(\ddot{z}_n + g_z)} \tag{2.8}$$

where:

x_{ZMP} is the x coordinate of the ZMP vector ($\vec{P}_{\text{ZMP}}$)
y_{ZMP} is the y coordinate of the ZMP vector ($\vec{P}_{\text{ZMP}}$)

- **The Fictitious Zero Moment Point (FZMP)** appears when the ZMP leaves the SA, making **AAU-BOT1** initiate a tilt, reducing the SA to a line or dot, as the SA then only consists of the edge of the foot. In some
- **Center of Pressure (CoP)** is the point on the ground where the resulting normal force F_N is working. This is per definition within the support polygon. CoP can be calculated using feedback from force-torque sensors in the feet. When the ZMP is within the support area, the CoP and ZMP coincides.
- **Balanced gait** is distinguished into different types:
  - **Statically balanced gait:** By using small accelerations ($\ddot{\vec{P}} = \dot{\omega} = 0$), GCoM and ZMP are equivalent, reducing Equation (2.7) and (2.8) to Equation (2.1).
  - **Dynamically balanced gait:** By increasing the accelerations, the GCOM and ZMP become seperate values, and the balance becomes harder to calculate and achieve.

Now that the coordinate system, the definitions and notation of humanoid robotics has been defined, the next subject will be the instrumentation of **AAU-BOT1**.

Chapter 3

Instrumentation and Network Design

*This chapter contains an analysis of the instrumentation and the network design for **AAU-BOT1**. First a description of **AAU-BOT1** in its current state is given this is done as the instrumentation and the network design depends on the existing mechanical setup. After this possible network designs are analyzed, i.e. determine whether the strategy is developed as a central or decentralize solution, or to determine whether the signals are transferred via a digital or analog bus. Next is the instrumentation strategy analyzed, i.e. to determine the instruments that e.g. can measure strains, accelerations. Last in the chapter is a summary describing the main points achieved during this chapter.*

3.1 AAU-BOT1 Description

This section describes **AAU-BOT1** as it was when it was handed over to the Section of Automation and Control the 30. of November 2007. The mechanical platform has been developed by a former mechanics group [Pedersen et al., 2007] in the time period September 2006 to June 2007. During the spring of 2007 the manufacturing of the **AAU-BOT1** was initiated at the Department of Mechanical Systems at Aalborg University and was handed over after completion. The **AAU-BOT1** was received in two parts an upper torso and a lower body. After assembling these **AAU-BOT1** was as shown in Figure 3.1.

The robot consists of an aluminum skeleton with: two basic arms, an upper torso, two legs, feet and two toes. In the following a list of received elements is shown:

- **AAU-BOT1**:
  The received robot features 19 degrees of freedom, where two of those are the toes. The two toes are each passive actuated by one spring. Most of the actuated DoF are located in the lower body. This robot also has a three DoF waist joint that can move the upper torso.

- **DC motors:**
  The 17 joints are actuated by 23 DC motors, where 6 of the joints are actuated by two DC motors. The joints with two DC motors are the ones that are heavily loaded. These are the ankle pitch, knee pitch and hip roll.

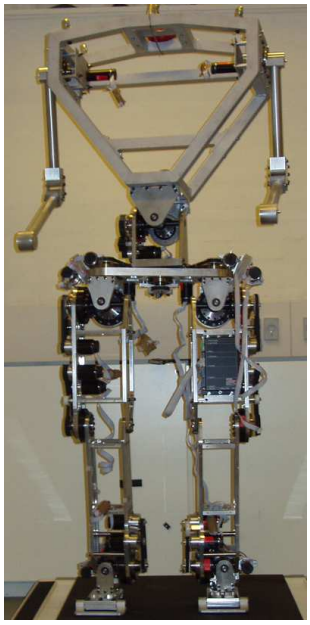


Figure 3.1: Photo of **AAU-BOT1**.

- **Analog amplifiers:**
  As there are 23 DC motors, 23 analog amplifiers powered the DC motors. These are place on the **AAU-BOT1**

- **Gears and belts:**
  The DC motors are connected to the joints via belts and Harmonic drive gears.

- **Force Torque Sensor:**
  Two force torque sensors have been developed. Each sensor consists of 6 strain gauge bridges, every bridge measures strain. This enables the FTS to measure the force and torques in three axes.

Without going into details it can generally be stated that following are needed to utilize the received hardware and features of the **AAU-BOT1**.

- **Computer:**
  One or more computers are needed as **AAU-BOT1** is proposed as an autonomous system and the computer equipment has to be located on the robot. It is taking care of all the data acquisition and has to control **AAU-BOT1**.

- **Data acquisition for analog amplifiers:**
  A data acquisition system is needed to give the DC motor amplifiers a reference input.

- **Data acquisition for FTS:**
  The signals from the FTS's have to be amplified in order to utilize the small signals measured on the strain gauge bridges. The measured signals have to be sampled such that forces and moments can be calculated.

- **Inertial Measurement Unit:**
  The IMU is used to measure x, y, z rotations this is used to give the global orientation of the robot.

The individual parts are elaborated on later in this chapter.

Before continuing with the network design the update frequency has to be determined. This can be done by observing the mechanical design and analyze the dynamics of the system. The mechanical design of **AAU-BOT1** is inspired by the design of Johnnie [Pedersen et al., 2007]. The Johnnie project [Löffler et al., 2004a] is used for further inspiration. The update frequency used for the control loop on Johnnie is 250 Hz. With this update frequency it has achieved to control it in such extent that it is possible to walk. The update frequency has to be much faster than the dynamics of the robot in order to prevent it from falling. The update frequency used on Johnnie is utilizable for **AAU-BOT1** and 250 Hz is used for the further development. If a higher update frequency appears to be obtainable, this will be revealed in later tests of **AAU-BOT1**.

With this basic information a proper implementation and network design can be derived.

3.2 Network Design

This section consists of different solutions to the network design for **AAU-BOT1**. Three solutions have been considered, a central network design, a hybrid network design and distributed network design. The choice of a network design also depends on the choice of motor amplifier, and FTS amplifier and other equipment.

3.2.1 The Central Network Design

Using a central network design means all input/out, digital or analog, will be connected to one central unit that also consist of a computer to process all data and control **AAU-BOT1**. One well known and widely used acquisition system is one from dSpace [Stoustrup, 2007]. The company manufactures many different versions, but common for them all is that they consist of several I/O cards which can be connected to one or more processor boards. Using this technology require that analogue controlled motor amplifiers are used. Furthermore dSpace is compatible with Simulink. One disliked feature concerned with the dSpace system is that all signals to the transducers are analogue and thereby sensitive to noise. Furthermore the weight is high and the system is to large to place on **AAU-BOT1**.

3.2.2 The Hybrid Network Design

The hybrid network proposed consists of 3 embedded decentralized computers connected to one central computer via Ethernet or USB. The embedded computer can be the TS7800 embedded computer from the company Embedded Arm. This computer consist of a 500 MHz ARM9 processor and supports a Linux operating system. Each of these computers handles a decentralize control loop and collect measured data and forward this to the central computer. This frees up resources on the central computer when the decentralize computer handle some of the burden. Unfortunately this solution cannot be used directly to control and receive measurement data from the motor amplifier, because it only has one 10 bit A/D converter. Since one A/D converter is not enough to handle all the

transducers, extra I/O cards are needed. In this solution the decentralized computers are place closer to the transducers, but the signals to the transducers are still transmitted as analog signals which make the system sensitive to noise.

3.2.3 The Distributed Network Design

A distributed network design is proposed such that all transducers as motor amplifiers and FTS amplifiers on **AAU-BOT1** are sampled decentralized. To enable this all the transducers can be grouped and connected to a common digital bus. One solution to this problem can be to use the EPOS motor amplifiers from Maxon Motors that features Controller Area Network (CAN). CAN is a well known network used for many automation and control applications. The CAN have a baud rate of up to 1 Mbit/s and is noise resistant. All CAN units can be connected to a central on-board computer. Thereby it is possible to control and get measured data from the motor amplifiers via CAN. Furthermore it is possible to measure relative joint angles via encoders mounted on the DC motors.

3.2.4 Evaluation of Network Design

Three network designs have been proposed. The selection of a proper network design has great influence of the hardware setup of the system, so this is carefully considered. The first described solution was the dSpace acquisition system. This solution is discarded since **AAU-BOT1** is proposed as an autonomous robot and the dSpace system cannot be placed on **AAU-BOT1** as it is to heavy. The two last solutions are similar as they both are proposed as decentralized systems. The hybrid network design with the three embedded computers cannot be used out of the box as additional I/O cards have to be purchased and interface electronics has to be developed, which is why this solution is not used.

The last proposed solution was the distributed network design. This utilizes the EPOS amplifier with CAN interface. These seem to be the obvious choice as these can be connected directly to the DC motors and be interfaced via CAN. The solution with the EPOS amplifiers with CAN interface from Maxon is chosen.

This selection of CAN amplifiers gives some consequences, as the CAN network only supports a baud rate of 1 Mbit/s. This gives a limitation to the bandwidth and thereby to the number of units that can be connected to the CAN network. In order to determine the number of units that can be connected, information about the CAN protocol and communication with the EPOS amplifiers is needed, which is retrieved from the firmware reference manual [Maxon Motors, 2007a]. Here the number of packages needed to operate the amplifiers is obtained. The most feasible way of sampling the data from the amplifiers is given below:

1. Transmit a Sync. to the amplifiers, where after the amplifiers will transmit their sampled data back via Transmit Process Data Objects (TxPDO).

2. Transmit control references to the EPOS amplifiers such that the DC motors can be controlled.

3. Activate the control references on the EPOS's

According to [Dalsgaard, 2007] the CAN communication consists of 20% bitstuffing, this is included in the later calculations.

With the overall sampling time of 4 ms (250 Hz), the number of CAN networks has to be chosen. By distributing the EPOS amplifiers on five CAN networks yields the sample time of the EPOS amplifiers, as seen in Equation (3.1).

Description	No. of packages	Time [ms]
Sync	1	= 0.1
TxPDO	5	= 1.0
RxPDO	1	= 0.2
Activate	1	= 0.2
Total		= 1.5

(3.1)

where:

Sync	is a synchronization messages.
TxPDO	is a transmit PDO from the EPOS, with feedback information.
RxPDO	is a receive PDO from the on-board computer, with control references.
Activate	is a messages that activate the new reference.

1.5 ms is the time it takes to retrieve the sampled data, transmit control references and activate the control references on the EPOS amplifiers. This leaves the rest of the system with 2.5 ms to handle e.g. a central control loop.

The developed Force Torque Sensors (FTS) have to be sampled and data has to be transmitted to the on-board computer. It means that the measurement system must be faster than 4 ms.

A detailed description of the hardware design and the choice of hardware are described in the following subsections.

It is chosen to convert all the busses to one common bus for easier implementation. Here the Universal Serial Bus (USB) is used as it is very versatile. The USB-2 standard features 480 Mbit/s throughput, this is considered sufficient for this project.

Figure 3.2 on the following page is a sketch of the entire the entire network with the different hardware elements.

3.3 Actuators

The system contains 23 DC motors distributed on the 17 actuated joints. The reason why there is more motors than joints is because 6 of the heavily loaded joints are powered by two DC motors sharing the burden. On the DC motor an encoder is mounted and thereby the relative position of the DC motor is given. Note that the absolute position of the DC motor is not supported and can only be obtained through an initialization process with a calibration potentiometer or switch. In Figure 3.2 on the next page the DC motors and encoders can be seen. The number of motors and encoders are listed below:

- 23 DC motors - 12 working synchronously in pairs of 2
- 23 two channel encoders not absolute, each encoder gives 2048 pulses at each turn.

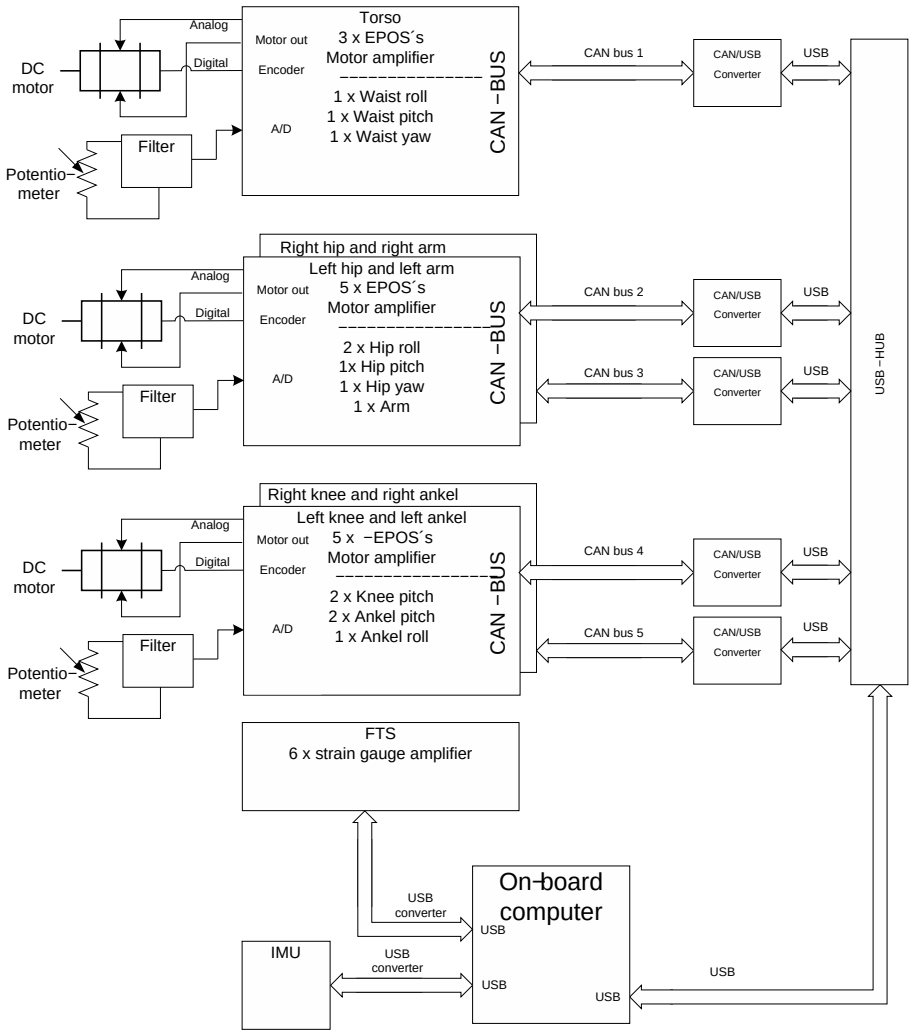


Figure 3.2: Instrumentation and network strategy

3.4 Amplifiers for DC motors

To control the 23 DC motors, an amplifier for each has been implemented. In the report from the mechanics group [Pedersen et al., 2007] it can be read that analogue controlled motor amplifiers was purchased. After the analysis in Section 3.2 on page 33, a distributed network solution was chosen and as a result of that, new amplifiers were necessary. The new amplifiers are evaluated better, they feature amongst others a built in a PID controller, A/D converter, digital I/O. Furthermore the entire motor setup will be very insensitive to noise since all measured data can be read and the motor can be controlled digitally via a CAN bus. The CAN bus is connected to the laptop via a CAN to USB converter from the company Peak Systems. The amplifiers uses the CANopen software protocol, which enables broadcasting of data to two or more amplifies and thereby enable synchronous control of joints with two motors. The amplifiers are of the type EPOS 70/10 from Maxon Motor. This means that the servo amplifiers can be powered by a voltage from 11 V to 70 V at 10 A which makes them very versatile and can be used with a wide range of power supplies.

All the relevant features, inputs and outputs are listed below:

- **Features**
  - Power supply voltage 11-70V
  - Speed control using encoder signals
  - Position control using encoder signals
  - Current control
- **Interface**
  - CANopen
  - RS232
- **Input**
  - Encoder signals from DC motor
  - 8 digital inputs
  - 2 analog inputs - 10 bit
- **Outputs**
  - DC motor connection
  - 4 Digital outputs
  - 5V voltage supply max 100mA

3.5 Absolute Joint Angle Measurements

Potentiometers is used to measure the absolute angles of the different joints. Unfortunately using potentiometers often results in very noisy measurements, due to electromagnetic noise from the surroundings. To improve the joint angle measurements, the digital encoder at the motor will be used to give an accurate relative joint angle. The motor amplifier has a 10 bit A/D converter which makes it possible to measure the absolute joint

angles for calibration via potentiometers. Via Equation (3.2) the minimum resolution is given to 0.01° since the maximum required angle according to [Pedersen et al., 2007, 49] is 95° + 5° in safety margin in case of a miscalculations. The minimum resolution at the encoder is 0.0016° given by Equation (3.3), since the minimum gear ratio is 111 [Pedersen et al., 2007, 69]. This means that it is possible to get an absolute joint angle during the initialization via the potentiometer and get a high resolution, but not absolute during operation of **AAU-BOT1** via the digital encoder.

$$\text{RES}_{\text{pot}} = \frac{\text{max joint angle} + \text{safety margin}}{\text{A/D resolution}} = \frac{95+5}{2^{10}} = 0.01^\circ \tag{3.2}$$

$$\text{RES}_{\text{enc}} = \frac{360^\circ}{\text{pulses each turn} \cdot \text{min gear ratio}} = \frac{360}{2048 \cdot 111} = 0.0016^\circ \tag{3.3}$$

The final result of the joint angle precision and resolution will be evaluated during test and implementation of the instrumentation and network design of **AAU-BOT1**.

3.6 Force Torque Sensor

Two six axis Force Torque Sensor (FTS) have been developed to measure the forces and torques that acts on the feet of **AAU-BOT1**. Therefore one FTS is situated in each ankle. With a FTS it is possible to determine the CoP from the sensor outputs. This section describes the design of the FTS, how to measure the strain gauge signals and how to interface the on-board computer. Furthermore it describes the calibration of the FTS and evaluates the performed calibration process. A comprehensive description of the calibration can be found in Chapter H on page 215. In [Pedersen et al., 2007] the development FTS is described. This involves design of the FTS, milling of the metal and mounting the stain gauges.

3.6.1 Mechanical Design

The FTS consist of three I-beams that are attached to a support core in one end and a support ring in the other end, see Figure 3.3 on the facing page for a sketch of the FTS. The FTS is mounted between the foot and the ankle. The maximum loads/overloads that the FTS is designed for are listed in Table 3.1 [Pedersen et al., 2007, 155]. During test of **AAU-BOT1** it must be validated that these loads/overloads are not violated.

Table 3.1: Force/torque maximum load [Pedersen et al., 2007, 155].

Force/Moments	**Max values**
F_x	±1000 N
F_y	±1000 N
F_z	±2000 N
M_x (roll)	±200 Nm
M_y (pitch)	±230 Nm
M_z (yaw)	±30 Nm

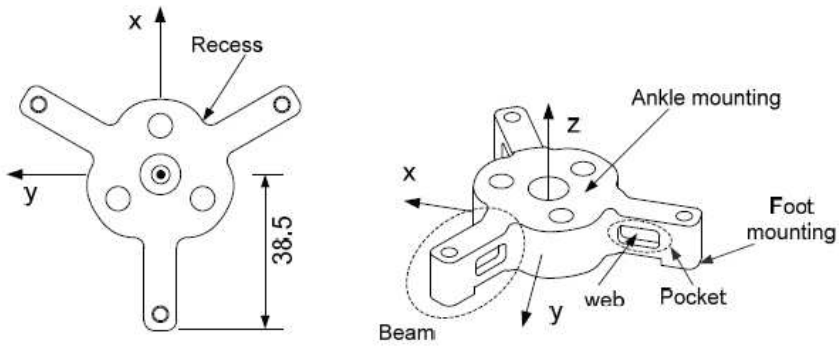


Figure 3.3: Left: FTS core, including its local coordinate system. Right: The beam part of the core is marked by a hatched circle

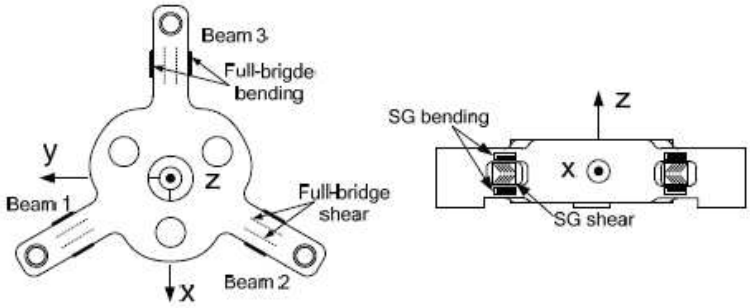


Figure 3.4: Location of the SGs on the core of the FTS.

3.6.2 Strain Gauges in the FTS

Each of the 3 I beams has 8 strain gauges mounted, which are able to measure the shears and the bendings. The 8 strain gauges are mounted in two full bridges. By connecting the strain gauges in a full bridge, undesirable things such as temperature sensitivities can be avoided and a differential signal is produced, which have a good common mode rejection. Figure 3.4 illustrates where the shear and the bending strain gauges are placed on the FTS.

All strain gauges used are from the manufacturer, Tokyo Sokki Kenkyuio Co. Ltd., their specifications are listed in Table 3.2

Table 3.2: Specification of the strain gauges from, Tokyo Sokki Kenkyuio Co. Ltd..

	Type of SG	**Resistance [Ω] (mes. val)**	**Folio [mm]**	**Gauge factor**
Shear	QFCT-2	350 (355)	7.6 / 5.3	2.09 ± 1%
Bending	FLA-3-11	120 (122)	8.8 / 1.7	2.10 ± 1%

3.6.3 Strain Gauge Measurements

Since the mechanical part of the FTS has been developed by [Pedersen et al., 2007] the main focus is to enable signal acquisition of the strain gauges. To be able to measure the

signals, the following criteria has been set up:

- The signal must be measured linear according to [Voyles et al., 1997].
- Since the applied load is measured via a full bridge, the measurement must be measured as a differential signal.
- The measurement circuit must suppress the noise from surroundings.
- The circuit must be able to measure changes in μV area.
- To reduce the number of different supply voltages, it is chosen that the strain gauge circuit should be supplied with a single supply.

Examined or developed Data aqusitions (DAQ) systems:

FTS DAQ Print Circuit Board (PCB) A FTS amplifier board was developed from scratch during the first two months of the project period. The amplifier circuit was designed with dual instrumentation amplifiers, which have high common mode rejection and is very linear. The PCB was fully developed and electronic components were also mounted on the PCB, but after a design meeting with [Bisgaard, 2007a], it was decided to buy an amplifier with digital interface and the solution was therefore not used. In Appendix G on page 209 the developed PCB is documented.

Mantacourt solution The Mantacourt solution was the obvious choice to measure the strain gauges with, since it has CANopen interface and the dimension of each amplifier is very small. Furthermore the product specification specifies that it is able to sample with 250 samples/sec. Furthermore Mantacourt also gives source code to interface the digital amplifier. A more detailed description of this solution can be found in Appendix G on page 209.

Lorenz Messtechnik GmbH This product is a 2 channel strain gauge amplifier from Lorenz Messtechnik GmbH. This device have a sample frequency up to 3000 sample/sec and has a 16 bit A/D amplifier built in. Furthermore it is possible to measure 120 Ω and 350 Ω strain gauge bridges. The product has RS485 interface. The product includes limited source code for a driver to interface it.

Choice of FTS DAQ system After analyzing the different solutions on the market, the product from Lorenz Messtechnik GmbH was chosen since the solution with Mantracourt unit could not guarantee the specified sample rate specified in their datasheet. This was first discovered after several conversation with the company and the product can therefore not fulfill the requirement for **AAU-BOT1**. A sketch of the final solution can be seen on Figure 3.5 on the facing page. Some of the product features for the digital amplifier from Lorenz MESSTECHNIK are listed here:

- 2 channel - strain gauge measurement of 0.35...3 mV/V.
- Control activation by software.
- Fast measuring rate, up to 3000 samples/s each channel.
- 10...30 V DC excitation.

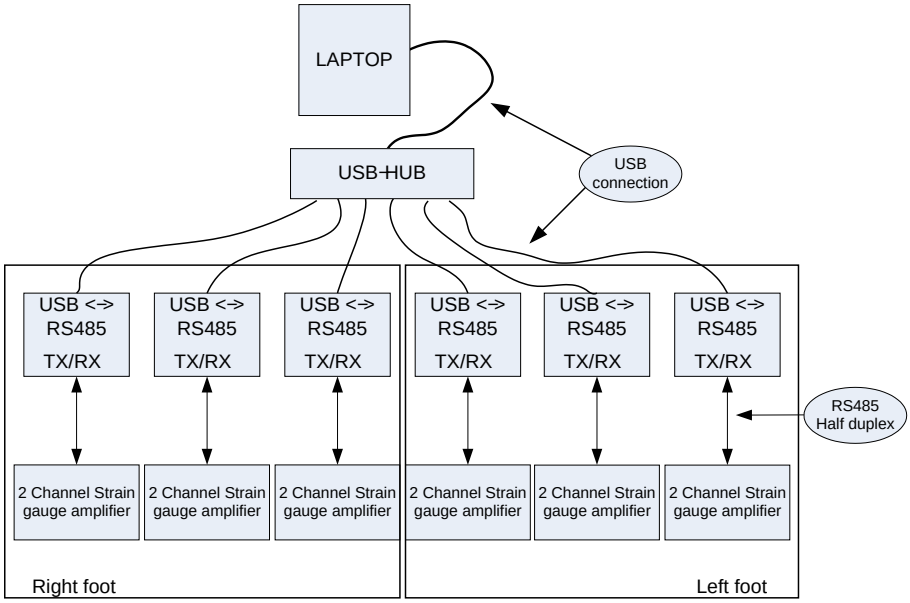


Figure 3.5: Sketch of the selected DAQ system for the FTS

- RS485 (half duplex)
- 16 bit A/D conversion.
- Streaming mode or polling mode.
- 230.4 k or 115.2 k baud.
- Screw connectors.
- C source code for connecting the digital amplifier
- 4-wire strain gauge bridge.
- Maximum strain gauge load: 120 Ω to 5000 Ω.

3.6.4 Calibration of a Six-axis FTS

The basic idea of the FTS is to measure signals from the strain gauges and estimate the forces and torques that causes them. The problem can be written as: The sensor converts the applied load vector $\vec{m}$, into a measurement vector $\vec{V}$, where the measurement vector is the strain gauges. The relationship between the measurement vector and the load vector can be expressed via a calibration matrix $\boldsymbol{C}$. Equation (3.4) describes the relationship between the measurement vector $\vec{V}$ and the load vector $\vec{m}$. The relationship is true if and only if the strain gauge measurements are linear. However this is not the case, since the amplifier and strain gauges have different gains and offsets. To overcome this, another method is used, which is the least square method. This method is described by [Voyles et al., 1997] and [Flay and Vuletich, 1995], furthermore [Pedersen et al., 2007] also used this method with success.

$$\boldsymbol{C}\vec{V} = \vec{m} \tag{3.4}$$
$$\boldsymbol{C} = C_{i,j}, \qquad i = 1,..,6 \quad j = 1,..,6 \tag{3.5}$$
$$\boldsymbol{V} = [v_{b1} \;\; v_{s1} \;\; v_{b2} \;\; v_{s2} \;\; v_{b3} \;\; v_{s3}]^T \tag{3.6}$$
$$\boldsymbol{m} = [F_x \;\; F_y \;\; F_z \;\; M_x \;\; M_y \;\; M_z]^T \tag{3.7}$$

where:

F_x	is the measured force in x-direction.
F_y	is the measured force in y-direction.
F_z	is the measured force in z-direction.
M_x	is the measured torque in x-direction.
M_y	is the measured torque in y-direction.
M_z	is the measured torque in z-direction.
V_{b1}	is the measured bending strain in bridge 1.
V_{s1}	is the measured shear strain in bridge 1.
V_{b2}	is the measured bending strain in bridge 2.
V_{s2}	is the measured shear strain in bridge 2.
V_{b3}	is the measured bending strain in bridge 3.
V_{s3}	is the measured shear strain in bridge 3.

The least square calibration matrix is found via Equation (3.8).

$$\boldsymbol{C} = \boldsymbol{F}\boldsymbol{V}^T(\boldsymbol{V}\boldsymbol{V}^T)^{-1} \tag{3.8}$$

Using the least square method require multiple measurements of different loads, this is described in Appendix H on page 215 where also the results are discussed.

3.6.5 Result of the Calibration of the FTS Amplifiers

The calibration result can be found in Table 3.3 on the facing page. The calibration is performed by using 4 different test loads and hereafter calculating the RMS error of the results. The force F_z has a large error. This error is caused by a poorly constructed calibration test rig. Calibration with the inadequate test rig results in an offset error on F_z. If an offset error of 50 N is added to F_z the RMS error is reduced to 2.103%. This gives a maximum RMS error of 2.403 %, however a new calibration test rig has to be designed, before a final result can be accomplished. Graphs and details of the calibration results are described in Appendix H.2 on page 217. In Equation (3.9) the final calibration matrix can be viewed.

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} -0.007 & -0.0263 & -0.0086 & 0.0176 & 0.0035 & -0.0193 \\ 0.2093 & -0.1552 & 0.2375 & -0.1925 & -0.0229 & -0.0107 \\ 0.017 & 0.0073 & 0.0197 & -0.0227 & -0.0036 & 0.0116 \\ -0.0029 & -0.0096 & -0.0082 & -0.0037 & 0.0222 & -0.0098 \\ 0.0002 & 0.0009 & 0.0007 & 0.0005 & -0.0021 & 0.0009 \end{bmatrix} \begin{bmatrix} V_{b1} \\ V_{s1} \\ V_{b2} \\ V_{s2} \\ V_{b3} \\ V_{s3} \end{bmatrix} \tag{3.9}$$

Table 3.3: Calibration results of the right FTS

Force/Torque	RMS error
F_x	± 2.373 %N
F_y	± 2.096 %N
F_z	± 16.693 %N
M_x (roll)	± 2.349 % Nm
M_y (pitch)	± 2.228 % Nm
M_z (yaw)	± 2.403 % Nm

Table 3.3 only contains results from calibration of the right FTS, which is because one of the amplifiers in the left FTS, started to malfunction after a short circuit in one of the EPOS's. Furthermore tests with the power supply for the FTS amplifier revealed that these are very sensitive to noise and it is not advisable to connect the FTS amplifiers and EPOS login to the same power supply. Even though the FTS amplifiers are sensitive to noise, they have proven to be very stable. It has been possible to develop drivers for them, which enables strain gauge measurements and thereby making it possible to calculate the forces and moments. In Section 4.7 on page 60 the software interface for the FTS is described. The final result of the implementation of the FTS amplifiers can be seen at Figure 3.6 on the following page. It is a custom made aluminum box to the right in the picture that contains 3 FTS amplifiers for one leg.

3.7 On-board Computer

The on-board computer retrieves and transmit data from and to the transducers on the CAN networks, the RS485 and RS323. These busses are all interfaced via USB converters. The sample rate is earlier chosen to 250 Hz (4 ms). I order to respect this sample time the on-board computer has to be able to execute the central control loop, sample data and set the control input to actuators within the given time. In Section 3.2.4 on page 34 the CAN network is calculated to use 1.5 ms for each time step. The other peripheral units connected to RS485 and RS232 are assumes to be sampled via other threads at the same time as the 5 CAN threads sampling the 5 CAN networks. It must be taken into consideration that it takes time to switch between the different networks and 0.5 ms is assumed to be used for this operation. This leaves 2.0 ms for the rest of the system including the control loop. The computer has to meet the following requirements:

- Sampling time faster than 2.0 ms
- Serial and/or USB interface
- Network 100Mbit/s
- Low power consumption
- Low weight

A Lenovo x61S has been chosen, since it is one of the lightest (1.3 kg) laptops and it has a Core 2 duo mobile processor with low power consumption. To reduce the laptops weight further and to make sure that the hard disk is not damaged during movement of **AAU-BOT1**, the original hard disk has been replaced with a solid state disk. It also has a 100 Mbit/s network and USB interface. The sample time is not verified since

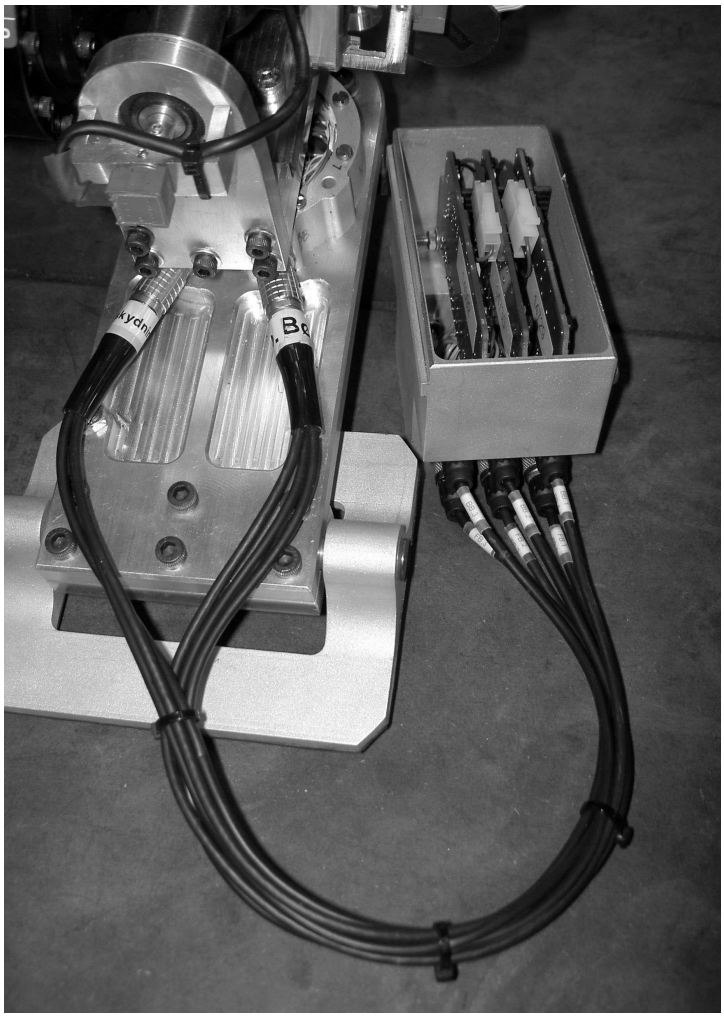


Figure 3.6: Picture of the final amplifier solution for one of the FTS.

this depend on the operating system of the Laptop and how many interrupts the USB port can handle. The on-board computer is tested as a part of the throughput test in Appendix I.

3.8 Inertia Measurement Unit

One reason why a human is able to balance is because it has the ability to feel the gravity, rotation and motion. This motion sense is located in the inner ear. If **AAU-BOT1** should be capable of human like walk then it needs a sensor that gives the position in 3D. With a Inertia Measurement Unit (IMU) it is possible to find the 3D-orientation. The following criteria have to be met:

- Since the weight of the robot is important, then the IMU's weight has to be low.
- It should be possible to read out the angles of the 3D-orientation.
- Minimum update rate: 250 Hz
- Digital output interface, i.e. RS-232, I2C.
- The IMU must be temperature compensated, to avoid drift and nonlinearities.

After reviewing a number of units, the MTi IMU from Xsens has been chosen. However it was not possible to find an IMU with a high enough update rate, low weight and high precision, so the used IMU only supports sample rate of 200 Hz. In order to solve this problem a Kalman filter can be used to estimate the missing samples [Grewal and Andrews, 2001].

The IMU features the following:

- Accurate full 360 degrees 3D orientation output (Attitude and Heading).
- Highly dynamic response combined with long-term stability (no drift).
- 3D acceleration, 3D rate of turn and 3D earth-magnetic field data.
- High update rate 200 samples second.
- The weight is 50 g.
- The measured data is Kalman filtered inside the IMU.
- Temperature, 3D misalignment and sensor cross-sensitivity compensated.
- Angular resolution: 0.05 °.
- Sensor range accelerations: +/- 5 g FS.
- Sensor range gyros: +/- 300 °/sec.
- USB interface with driver software included.

The IMU is not a active part of this project but is included in the design phase such that it can be implemented for later use.

3.9 Summary of Instrumentation and Network Design

During this chapter all instrumentation and network has been designed. After a careful analysis a solution was found. In order to save time, all instruments for this design were commercially bought items.

[Pedersen et al., 2007] had bought analog interfaced DC amplifiers, these turned out to be insufficient, since it was not possible to find acquisition hardware, that was light enough and had a high enough update frequency. Another solution is used, and all DC-motor amplifiers were changed to a version that have CAN bus interface (referred to as EPOS). These have a great advantages of having a built in controller, which support control of the position, current, and velocity. Furthermore it is possible to obtain information about the used current, absolute position (via potentiometers that are attached to the EPOS's), relative position and velocity, which all can be used for control of **AAU-BOT1**.

The FTS needed amplifiers to amplify the small strain gauge signals. It ended up with 3 different solution: Custom made amplifiers, a solution from Mantacourt and a solution from Lorenz Messtechnik GmbH. The amplifiers from Lorenz Messtechnik GmbH had excellent specification, and a custom made aluminum box was made for each shin, such that it is possible to amplify all signals from the 6 strain gauge bridges. The FTS's and the amplifiers has been calibrated, such that a RMS error of 2.4% were achieved in the right FTS. However this result could only be accomplished by removing an offset error of 50 N. The offset error was cause by a bad calibration test rig, which needs to be redesign in order to reduce the offset error. However this has not been done, due to the time limitation. It was unfortunately also discovered that the amplifiers has an unprotected supply voltages, since a short circuit in an EPOS amplifier resulted in a defect FTS amplifier in the left shin. With a defect FTS amplifier, it has not been possible to obtain a calibration of the left FTS and thereby it is not possible to determine the CoP on **AAU-BOT1**.

Furthermore an IMU was bought. It is a IMU with a solid state gyro and is low weight. It has been tested via enclosed software, but is has not been implemented on **AAU-BOT1**, since focus has been elsewhere.

The chosen on-board computer is one of the smallest laptops and it has high performance, i.e. a Pentium 2.0 GHz core 2 Duo processor and 2 GB Ram, and has a weight of 1.3 kg. The hard drive has been change with a solid state disk, such that it do not break down when **AAU-BOT1** is moving. The solid state disk, are slow compared to a regular laptop harddrive. However this does not affect the running program as this is executed in memory.

3 different types of network have been used on **AAU-BOT1**: One RS232 bus for the IMU, six RS485 busses for the FTS amplifier and five CAN buses for the EPOS's. The CAN buses and the RS485 buses are known for a high throughput, high noise suppression and well known techniques, these are all features that have been focused on in the instumentation.

The Safety of the equipment has not been a part of this thesis, however there have been installed safety mechanisms such that it can be operated safely both in relation to itself and the operator.

With the developed implementation strategy and network design, **AAU-BOT1** still

needs proper software to enable communication with the transducers. This is elaborated on in the following chapter.

Chapter 4

Software Architecture

This chapter contains the derivation of the Software Architecture, This includes development of driver software for the Force Torque amplifiers and the EPOS amplifiers, and real time execution of Simulink. To enable communication between the driver software and Simulink, a shared memory server and a number of sensor servers are developed and described. In order to test the developed controllers and models, a simulation tool called Webots has been used. The tool and the communication with Simulink is also described in the last part of this chapter. The software is documented using Doxygen[1], this documentation can be found on the enclosed CD.

4.1 General Software Description

One of the challenge when designing a software architecture for a system like **AAU-BOT1** is that the Software Architecture is very depended on the hardware. It must also be able to run in real time while it communicates by external peripherals. In order to obtain a useful solution, the Software Architecture is based on advice from [Bisgaard, 2007a]. A similar software architecture has been successfully implemented in the Ph.D. project [Bisgaard, 2007b]. However the instrumentation and network design are far more complex on **AAU-BOT1**, than on the used helicopter in [Bisgaard, 2007b], i.e. the software should be able to read data from 30 sensors and write data to 23 actuators as described in Chapter 3 on page 31. The Software Architecture is divided in two main groups, one part that is on-board **AAU-BOT1** and another part that is implemented on an external computer. The external computer is used to run a simulation program called Webots. The simulation program is implemented with a physics engine which makes it possible to create a virtual **AAU-BOT1**, which enables test of the developed controllers and models, this simulation program will be elaborated on later in this chapter.

The **AAU-BOT1** software is implemented on a Linux platform. The on-board computer uses the operating system Xubuntu [Xubuntu, 2008] which has been modified to run in soft real time. This gives the advantage of having a flexible system that runs out of the box.The soft real time part is an environment that enables real time execution of S-function/Simulink as described in Section 4.2 on the next page. During the software chapter, soft real time will be referred to as real time as it is done in [Dozio and Mantegazza, 2003].

[1] A documentation system for C++, C, Java and IDL. It generates Latex, HTML, RTF, Postscript and UNIX main page outputs from a set of documented source files [van Heesch, 2008].

All parts of the software communicates via a shared memory server, which contains data that have to be transmitted to all the actuators from the S-function or data that are received from the sensors. The shared memory server will be described in 4.3 on page 54. In order to interface the Shared Memory Server, two sensor servers and one actuator server is developed. These separates the sensors and actuators drivers from the shared memory, which make it easy to add or remove extra sensors or actuators without interfering with other sensors or actuators, as they are multi threaded.

The Actuator Server is connected to the EPOS driver. This makes it possible to forward data to the EPOS's via 5 CAN busses as described in Section 3.2.4 on page 34. The two sensor servers are connected to sensors at CAN buses and RS485 buses. In order to obtain data from the sensor, custom made drivers are developed. These drivers are described in Section 4.6 on page 57 and 4.7 on page 60 respectively. Figure 4.1 shows the designed software architecture. The driver servers that use CAN busses is marked with blue color and the sensors that uses RS485 busses is marked with purple. The IMU is not used in this project, but it is still implemented in software to ease future implementation of it.

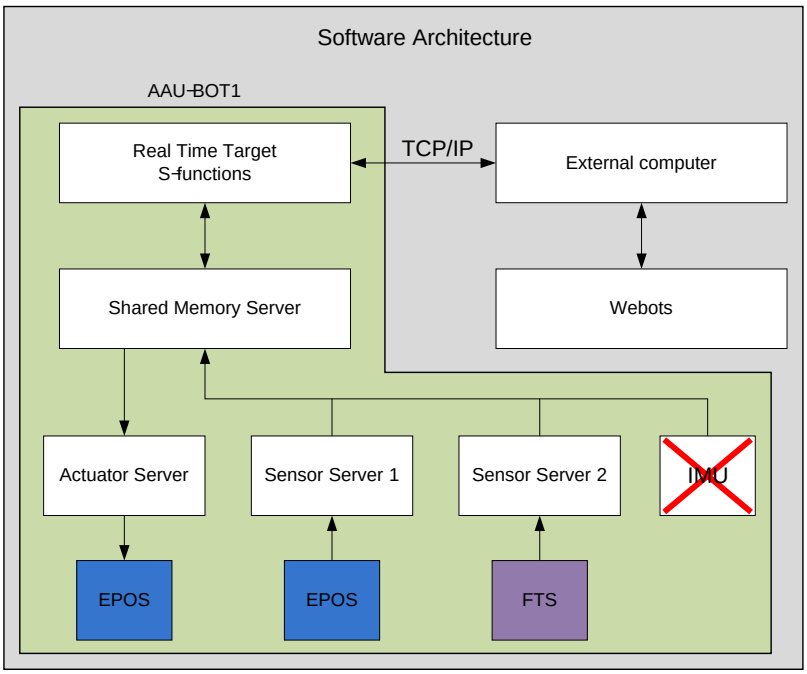


Figure 4.1: Sketch of the Software Architecture.

4.2 Simulink S-function Interface

The purpose of the Simulink S-function interface is to transmit the control signals to the actuator server and get the measurement signals from the sensor server. This is done by using two Simulink S-functions to interface with the sensor server, and adding a mask so that the settings can be set in the Simulink GUI. The inputs for **AAU-BOT1** can be three different types (current, angle and velocity), and each of these must be sendable

through the interface. Additionally, the EPOS amplifiers can measure different types of data (current, relative angle, absolute angle and velocity), which must be deactivateable to decrease the load on the CAN bus and the on-board computer.

4.2.1 General Description of the Simulink Interface

The general idea of the Simulink interface is to make **AAU-BOT1** act as a sink/source combination.

The EPOS amplifiers has three types of inputs: current commands (u_i), velocity commands (u_ω) and position commands (u_θ). To reduce the risk of false use, there are three inputs to the actuator sink, and the active input is marked with black text (see Figure 4.2). The input type and configuration parameters for the EPOS amplifiers can be chosen in the mask parameter dialog box.

The sensor read interface has seven different types of outputs that each can be turned on and off. This enables the system to in- or decrease the sample rate. The desired outputs can be chosen in the mask parameter dialog box and the active output ports are marked in black text (see Figure 4.3 on the next page).

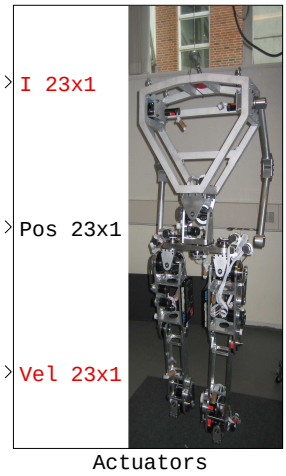


Figure 4.2: Simulink Sink block. As only one type of input can be transmitted at a time, the other two inputs are nonfunctioning.

4.2.2 General Software Description of the Simulink Interface

The Simulink S-functions consists of the parts seen in Figure 4.4, which are utilized in the following way:

- `mdlInitializeSizes()`:
  Initializes the number and width of the inputs and outputs of the model, and the number of parameters sent to the S-function.

- `mdlInitializeSampleTimes()`:
  Sets the sampletime of the S-function.

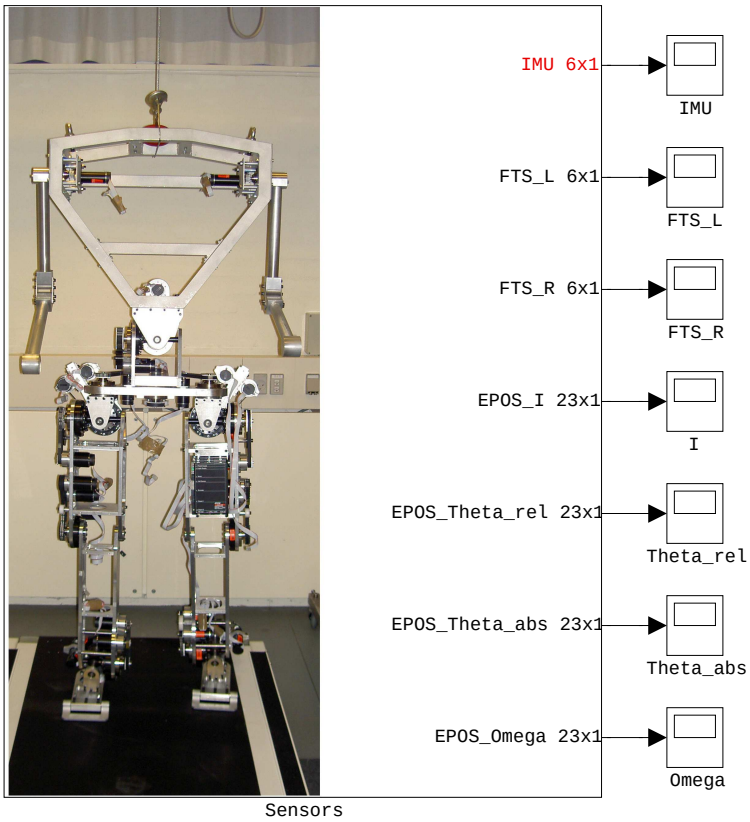


Figure 4.3: Simulink Source block. The outputs from the can be turned off individually, to optimize sampling frequency.

- `mdlStart()`:
  Connects to the shared memory, and sets the read flags in the shared memory that determines whether $\vec{i}$, $\vec{\theta_{abs}}$, $\vec{\theta_{rel}}$ and $\vec{\omega}$ to the value given in the mask.

- `mdlOutputs()`:

  - The inputs from Simulink is sent to the acuator server via the shared memory.
  - The measured values of the shared memory are output to Simulink.

- `mdlTerminate()`:
  The connection to the shared memory is severed. A close CAN bus command is sent to the actuator server.

As the model run in real time, the controller and the S-function interfaces are compiled with Linux Soft Real-Time Target v2.3, written by Dan Bhanderi. This Real-Time Target uses POSIX real-time clocks to generate periodic signals, to wake the model process every step time. In addition, the process is changed to highest priority in the Linux kernel scheduler, thus increasing the degree of real-time. The main advantage of using a

real-time target is that it enables the usage of high level programming in Simulink and has a high degree of real time.

The disadvantage is that it requires more processing power than required with a hard coded solution.

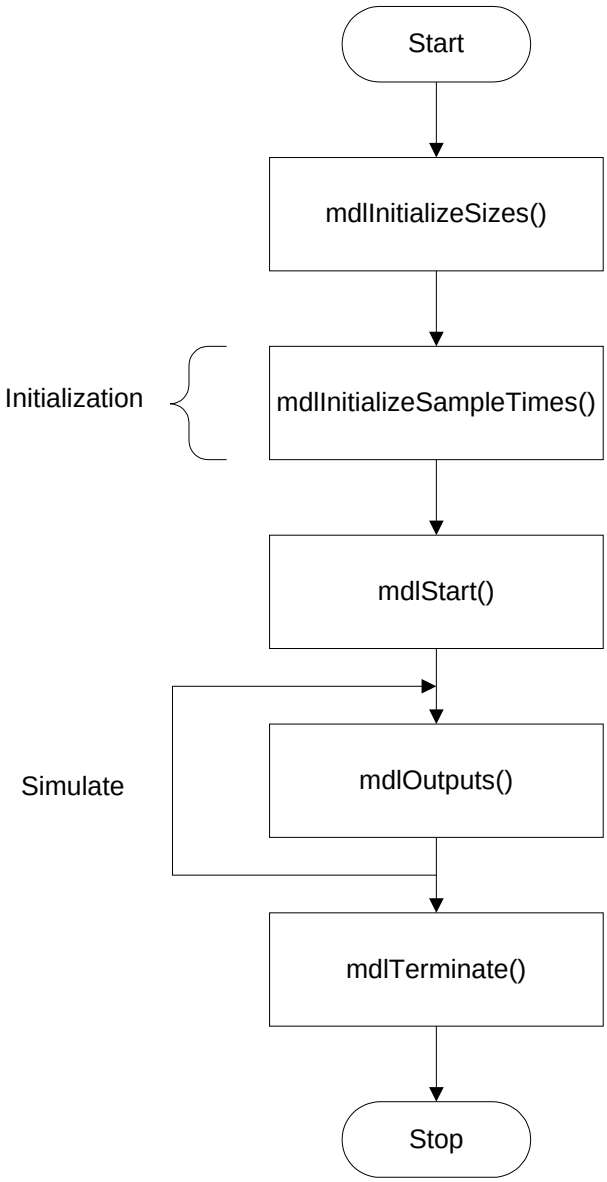


Figure 4.4: Flowchart of a S-function.

4.2.3 Test of the Simulink Interface

The Simulink read interface is tested by putting data in the shared memory, and examining whether the scopes display the data. The write interface is tested by adding inputs

to the sink. Both tests are completed succesfully.

4.3 Shared Memory Server

The shared memory server initializes and clears the shared memory. The Sensor Servers are able to get a pointer to two types of memory depending on the system runs in hard real time or in soft real time [Dozio and Mantegazza, 2003]. In case the system runs in hard real time the shared memory server allocate a chunk of memory to be shared as inter-intra kernel module and Linux process, which enables symmetric hard real time services, i.e. hard real time for all Linux schedulable objects, like processes/threads/kthreads and also for Real Time Application Interface (RTAI) own kernel tasks [Dozio and Mantegazza, 2003]. When the system runs in soft real time the XSI[2] shared memory will be allocated. This enables different programs to read/write to the same memory, but not in real time. As AAU-BOT1 is based on a soft target, soft real time us utilized and XSI memory is used. The disadvantage of using shared memory is that it is possible to change data in shared memory while reading the data. This could result in faulty sensor data if e.g. MSB is updated, but the LSB of the data has not while the S-function reads data. To check whether data have changed during reading, a special `locked` flag indicates if data is being altered in the memory. In case the flag is set, data must not be read from the shared memory otherwise corrupted might be read.

Using shared memory makes it possible to access the memory from several threads simultaneously. The shared memory can also accessed from an external computer via TCP/IP connection and thereby get an exact copy of the shared memory which is very useful during test of the controllers and sensors.

4.4 Sensor Servers

The sensor servers section describes how the computer handles inputs from the different sensors on **AAU-BOT1** and forwarding these to the shared memory. This section gives the reader an overview of how the sensor servers are designed, works and tested. The two sensor servers are designed as identical as possible and therefore only software for one sensor server is described. Due to the fact that the IMU is not necessary for static gait, the sensor server for it has not been created in this project.

4.4.1 General Description of the Sensor Servers

The sensor server consists of sensor server 1 and sensor server 2 as shown in Figure 4.5. Since there are 12 USB units, i.e. 5 CAN busses, 6 RS485 busses and 1 RS232 bus, the sensors have been distributed on two sensor servers. This gives the opportunity to use 2 software threads to get data, and thereby minimize the round trip time. This will be described in 4.4.2. The sensor servers are the link between each sensor driver and the shared memory. In Linux all communication to and from external hardware are done via files, i.e. USB0 can be found in file `/dev/ttyUSB0`. The path of the file is entered when starting the sensor server. The path of each sensor file is used to get a file descriptor which is used to send and receive data. The received data will be saved in the shared memory.

[2]XSI System Interfaces: The system shall support all the functions and headers defined in IEEE Std 1003.1-2001 as part of the XSI extension.

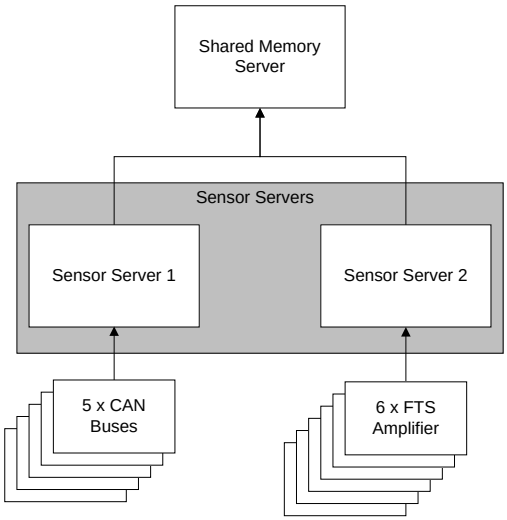


Figure 4.5: Sketch of the information flow in the Sensor Servers.

4.4.2 General Software Description of the Sensor Servers

The two sensor servers are designed as 2 threads which enables the computer to run them individually. The paths to the device files are input arguments to the sensor server program. In the initialization phase it connects to the shared memory and the device file is given as input. At this point the server goes into operational mode and enters a while(true) loop. The data from the device file is retrieved in different ways:

Sensor Server 1

Sensor server 1 serves the CAN/EPOS driver, and uses the `CAN_READ()` command to get data from the CAN busses. `CAN_READ()` waits for a message to arrive, and returns the message when it comes. See Figure 4.6 for a flowchart of sensor server 1.

Sensor Server 2

Sensor Server 2 serves the FTS amplifiers over the RS485 bus. To ensure that the computer gets new data from each sensor, a counter counts the while loops. The counter is reset every time data is received from the sensors. If no data is received before the counter reaches a threshold, a message is printed to the command prompt, this indicates if all sensors are operational.

The next task is to check whether there are any changes in any of the device files in the last 5 seconds. If this is not the case in returns an error and terminates the sensor server. If there are new data within the last 5 sec. a sensor handler function is called with the file descriptor and a pointer to the shares memory. Only sensor handles with changes on its respective device file will be called. After this point the sensor server will restart the while(true) loop. Figure 4.7 shows a flowchart of how Sensor Server 2 works.

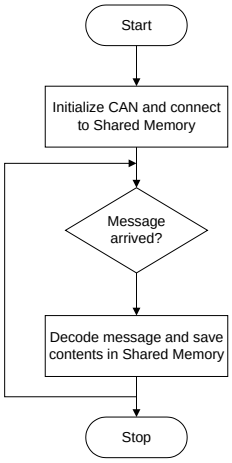


Figure 4.6: Flowchart of Sensor Server 1.

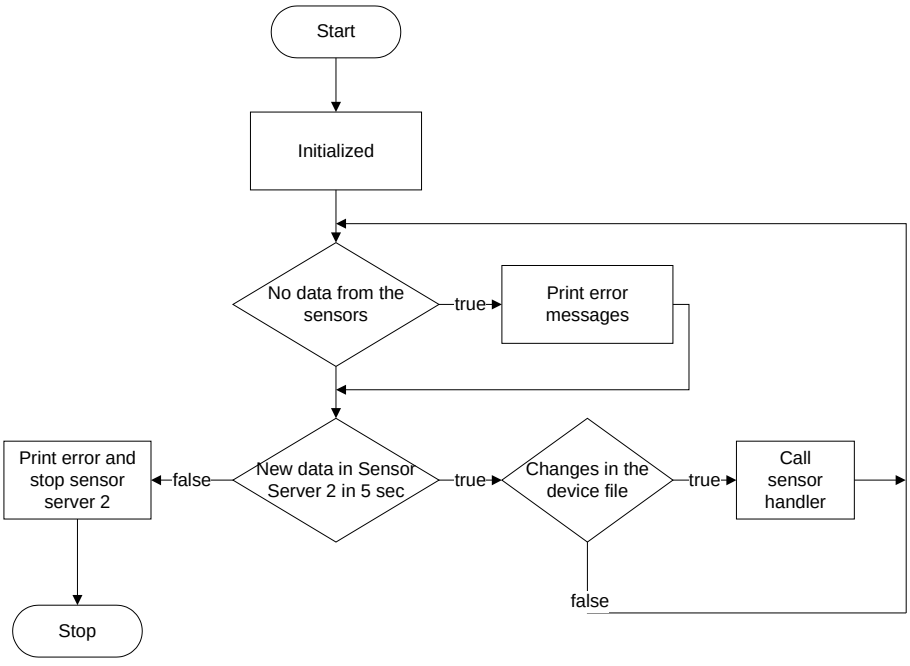


Figure 4.7: Flowchart of Sensor Server 2.

4.4.3 Test of the Sensor Servers

Both sensor servers has be tested by enabling the sensors and verifying that the data changes in the output of the Simulink S-functions described in Section 4.2. Via the USB/RS485 converter it is possible to see whether data is received or not since it enables a red LED when it read or write data. Furthermore it has been tested that the program can not start if the paths for the device file are not correct. It has also been tested that is possible to start the sensor in different USB plug-in as long as the path for the device file is corrected entered during startup of the sensor server. The result of the tests are that the sensor servers works as designed.

4.5 Actuator Server

The purpose of the actuator server is to transmit the commands in the command queue via the CAN networks to the EPOS amplifiers. This section gives an overview of how the actuator server is designed, and how it works.

4.5.1 General Description of the Actuator Server

To remove the sleep cycles necessary when communicating to the PEAK CAN driver, the command transmission is separated from the Simulink S-function. As the soft real time target requires that the memory is allocated in the initialization phase, a ring buffer is used to contain the commands.

4.5.2 General Software Description of the Actuator Server

The actuator server has 3 phases:

1. **Initialize:** The actuator server connects to the 5 CAN busses.
2. **Running:** If there is a new message in the ring buffer (i.e. the head pointer and the tail pointer are not equal), a `switch` command determines to where the message in the queue is to be sent, and sends it to the appropriate CAN using the PEAK driver. If there is not a new message in the queue, the actuator server sleeps for 50 μs, and looks if there are any new messages in the queue again.
3. **Exit:** The actuator server disconnects from the 5 CAN busses.

4.5.3 Test of the Actuator Server

The actuator server is tested by connecting to the EPOS amplifiers via RS232 and checking whether the commands in the queue is received by the EPOS amplifiers. This test is described in Appendix I.1, and concludes that the actuator server works as planned.

4.6 EPOS/CAN Driver

To enable the actuator server and sensor server 1 to communicate with the amplifiers, the EPOS/CAN driver is used. This section will describe how the EPOS/CAN network is setup, how it works and how to handle the protocol that is used to communicate with the EPOS amplifiers. Furthermore the section will describe how to initialize and run the driver. In the end of this section a description of how the driver is tested can be found.

4.6.1 General Description of the EPOS/CAN Driver

The primary idea of the EPOS/CAN driver is to enable communication with the CAN network and maximize the throughput as much as possible. The CAN units uses several types of data objects, and can be situated in different modes, this is elaborated on in the following:

- Service Data Object (SDO):
  SDO packages requests or sets a specific measurement/setting. As the SDO is not predefined, the setting or measurement in the frame is specified in the frame. This yields a high degree of overhead(see Figure 4.8 for a comparison of overhead).

- Process Data Object(PDO):
  Is a predefined data object and can contain measurements and settings. There are two kinds of PDO's: transmit and receive PDO's (TxPDO and RxPDO). The former is for data transmitted from the device and the latter is for data transmitted to the device, ie. with RxPDO the driver can send data to the device and with TxPDO the driver can read data from the device. The PDO's has a smaller overhead and can contain more user data than SDO packages, as the PDO's are predefined.

- sync:
  The sync makes all the nodes sample their measurements and transmit them using the predefined PDOs. (see Figure 4.9 for a timeline of a `sync` message command).

- Pre-operational mode:
  When the EPOS CAN unit is switched on it automatically enters pre-operational mode with is a state where the equipment can be configured via the SDO packages. In pre-operationel mode the actuators are offline.

- Operational mode:
  After the configuration of the EPOS amplifier is done, the unit can be set into operational mode. Here both PDO and SDO packages can be transmitted. The actuators are now online and can be used.

To increase the throughput of the CAN network and reduce the time between a command is sent from Simulink to it is received by the EPOS amplifiers, the following strategies are used:

- By using PDO frames for the communication with the amplifiers during the operational mode, the following is achieved:
  - Reduction of overhead, as the content of the PDOs are defined in the pre-operational mode, and several amplifiers can react to the same PDO. The designed PDO frames can be seen in Appendix J.
  - Synchronous sampling, due to the sync command.
  - Synchronous control signals, as it is possible to activate the control input with one activate command.

- Multiple amplifiers are set to react to the same PDOs, this reduces the amount of PDOs sent per sample especially in the case of the double actuated joints.

Adr	Len	Info1 [4B]				Info2 [4B]			
200 + id	8	$u_{\omega 1}$				$u_{\omega 2}$			
201	8	0x64	0x00	0x00	0x00	0x00	0x00	0x00	0x00

(a) RxPDO frame.

Adr	Len	Command type				parameter			
600 + id	8	Velocity command				$u_{\omega 1}$			
601	8	0x23	0xff	0x60	0x00	0x64	0x00	0x00	0x00

(b) RxSDO frame.

Adr	Len	Measurement type	parameter
580 + id	8	Velocity	ω_1

(c) TxSDO frame.

Adr	Len	Measurement1	Measurement2
180 + id	8	ω_1	θ_1

(d) TxPDO frame.

Figure 4.8: Comparison between PDO and SDO frames. Both Rx frames transmit a velocity command (100) to EPOS 1, however the RxPDO can transmit a velocity command to EPOS 2 at the same time. In addition, any EPOS can be set to listen to any PDO, which will be utilized at the dual axis motors. The TxPDO frames can contain both the velocity and position in one frame, whereas the TxSDO has to use two frames for the same amount of information.

4.6.2 Software Description of the EPOS and CAN Driver

To communicate with the EPOS via CAN, an open source library called CanFestival [LOLITech, 2007] was considered. However, due to the lack of EPOS driver to CanFestival another solution to the problem is pursuited. This is the ASL-HWlib from the Autonomous Systems Lab in Zürich [ASLSOFT], here a partial EPOS driver is developed. The ASL-HWlib does not fulfill the demands of this project, and it has been rewritten to support PDO packages and specific EPOS SDO commands. Furthermore it is enabled to read and transmit data to the shared memory. The CAN driver used to communicate over CAN is provided by PEAK systems which are the manufacture of the USB/CAN converters.

The EPOS driver and the CAN driver is described in the following:

- EPOS driver:
  Contains the basic commands for communicating with the EPOS amplifiers, by translating commands into CAN frames. See Figure 4.8 for an example of the individual frames.

- CAN driver:
  The CAN driver must transmit the commands to and from the CAN bus to the EPOS driver.

Both the actuator server and the sensor server connects to the CAN busses using the `LINUX_CAN_Open()` command. The CAN driver and EPOS driver is compiled and included in the actuator server and sensor server 1, which utilize the commands when needed.

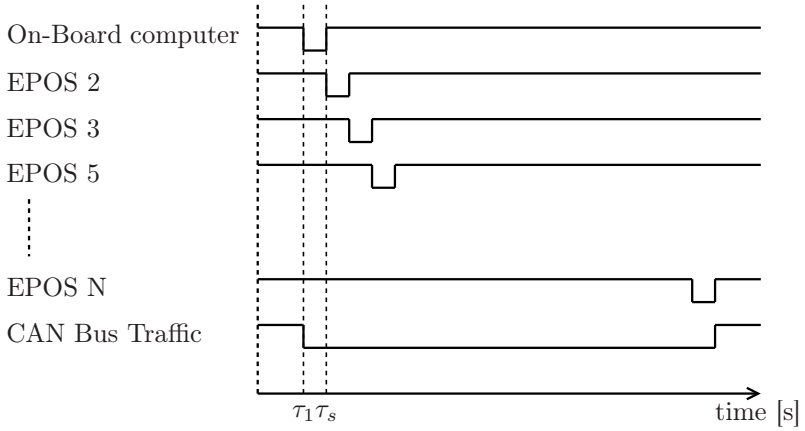


Figure 4.9: Timeline of execution of a `sync` command. The On-board computer transmit a `sync` command, which makes the EPOS' transmit their PDOs in turn. τ_1 is the sync time, τ_s is the sample time thus all the EPOS' sample at the same time.

4.6.3 Test of the CAN/EPOS Driver

The CAN/EPOS driver is tested by transmitting commands to and from a number of EPOS amplifiers. These tests are completed successfully, and will be elaborated on in the DC motor model verification in Appendix A.1.

4.7 FTS driver

The FTS driver enables measurements of forces and torques via the strain gauge bridges in the ankles, 6 digital amplifiers have been bought from Lorenz Messtechnik GmbH. This section will describe how the FTS amplifier is setup, how it works and how to handle the protocol that is used to communicate with the amplifiers. Furthermore the section will also describe how to initialize and run the driver. Finally the section will contain a description of how the driver is tested. The details of the FTS amplifier can be found in [Lorenz Messtechnik, 2001].

4.7.1 General Description of the Interface to FTS Amplifier

Each amplifier is able to measure 2 strain gauge full bridges at the same time. It means that the FTS driver has to handle 6 amplifiers. The driver is designed in such a way that the amplifier can be initialized and operate in streaming mode via 2 function calls. This makes it very easy to use the driver via Sensor Server 2. To communicate with the amplifiers 6 USB/RS485 converters are used. The converters are from the company 4N-GALAXY. These converters use the standard Linux serial driver, `ftdi_sio`, so it is possible to access the RS485 bus as an asynchronous serial bus.

One of the important issues when designing a driver for the FTS amplifier is to have a high sample rate and at the same time minimize the number of interrupts in the computer. The amplifier features 3 different protocols. The first protocol is the normal Lorenz mode protocol which enables the user to setup and calibrate the amplifier. The normal Lorenz mode protocol calculates a check sum of the payload that is showed in Figure 4.10. The check sum is a sum of all bytes in the payload and to be extra safe the

protocol also specifies a weighted check which is a sum of sums of the data. The normal Lorenz protocol also specifies that `0x02` is the start byte of all messages, this means that all `0x02` in the payload, checksum and in the weighted checksum is doubled, i.e. if e.g. the command byte is `0x02`, an extra `0x02` is inserted in the messages. The checksum should be calculated before the `0x02`'s are doubled.

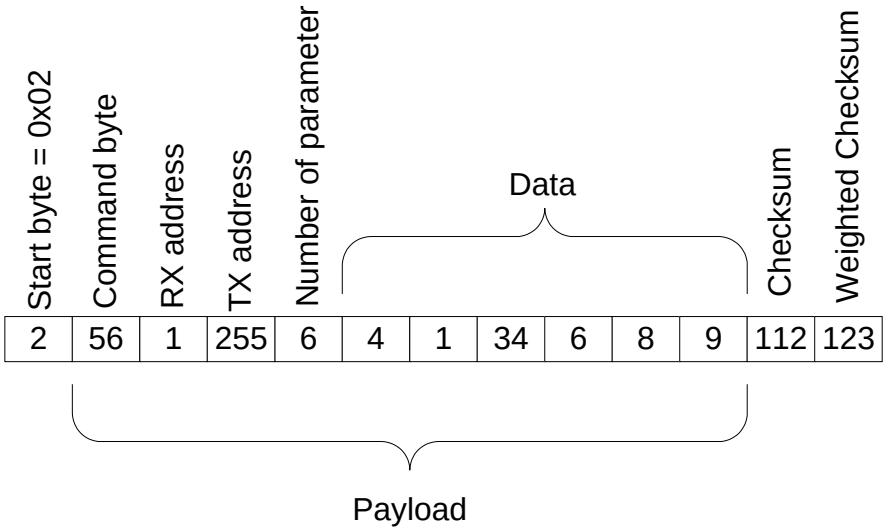


Figure 4.10: Sketch of the normal Lorenz mode protocol.

The two other protocols are a speed optimized polling protocol and a Speed Optimized Streaming Mode protocol(SOSM). Both protocols are designed to have low overhead. Since it is important to reduce the number of interrupts and minimize workload on the computer, the SOSM protocol is used. The SOSM protocol enables high sample rate of up to 2,500 samples/sec. The message in the SOSM protocol consist of only 5 bytes, 2 bytes for channel A and 2 bytes for channel B at the amplifier. The last byte is a synchronization byte which is a number between 0 and 255. The sync number is increased by 1 each time a messages is sent, when the number reaches 255 it becomes 0 the next time. As in the normal Lorenz mode protocol, the SOSM protocol also double '0x02' in the message from the amplifier. Figure 4.11 illustrate how the raw input data looks like and how the modified input buffer where the extra `0x02`'s are removed.

Even though the SOSM protocol has many advantages, there are also three main disadvantages. The first is that data is received asynchronously, i.e. there is not always a whole message in the input buffer and it is also possible to have several messages in the input buffer. This makes it hard to identify what the data exactly is. It is therefore necessary to shift the entire buffer when the last message has been found, so the latest whole message is in the beginning of the buffer.

The second disadvantage is that no checksum is calculated which makes it difficult to detect corrupted data. The risk for corrupted data is minimal because the amplifier and USB/RS485 converter both are shielded, the RS485 bus is very good at suppressing common mode noise and the amplifier is designed to use a relative slow baud rate at 115,200.

The third disadvantages by using the amplifier in SOSM, is that jitter can occur when

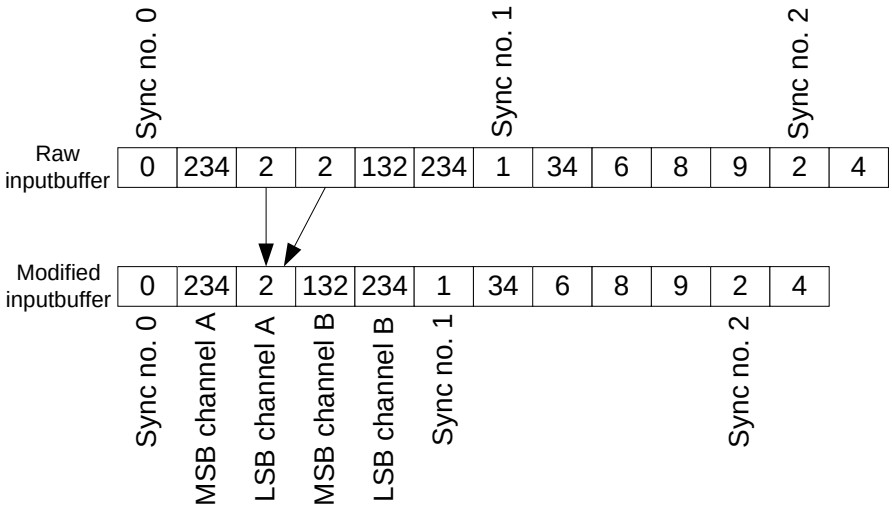


Figure 4.11: Sketch of the speed optimized Lorenz mode protocol.

the data is received from the 6 amplifier. Since it is not possible to hardware trigger the sampling on the amplifier the jitter will always be there. In order to reduce the jitter the sample rate is set to 1,250 samples/sec. This reduces the worse case jitter time to maximum 800 μs (1/1,250). Figure 4.12 illustrate how the timestamps of the received messages jitter between each amplifier.

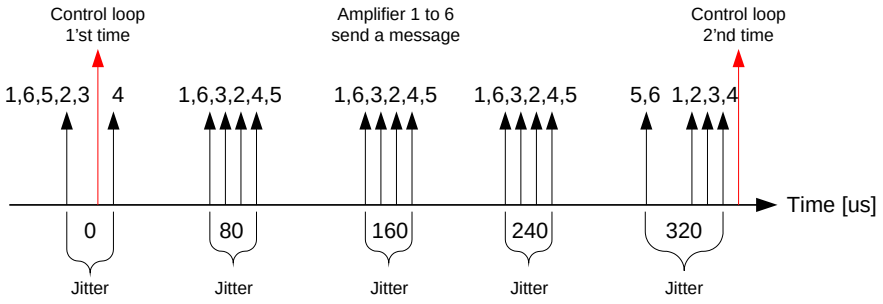


Figure 4.12: Sketch of how the timestamps of the received messages jitter and how the FTS is oversampled.

Since USB adresses are dynamical assigned, the FTS amplifier is set up with a uniqe SW ID such that it can be identified when connected to the USB port through the USB/RS485 converter. This means that the USB/RS485 converter can be connected to an arbitrary USB port. The hardware ID (HW ID) is unique for each sensor and it can be read via the Lorenz program. The software ID (SW ID) is a number that the amplifier is given via the Lorenz program. The shear and bending bridges must be connected as in Table 4.7.1. The shear bridge must always be mounted at channel A (CHA) and the bending bridge must always be mounted at channel B (CHB).

Table 4.1: Hardware and software ID' for each shear and bending bridge on the FTS amplifier.

Ankle	HW ID	SW ID	Shear no.	Bending no.
Left	14729	1	CHA 1	CHB 1
Left	14730	7†	CHA 2	CHB 2
Left	14692	3	CHA 3	CHB 3
Right	14690	4	CHA 1	CHB 1
Right	14691	5	CHA 2	CHB 2
Right	14693	6	CHA 3	CHB 3

†: The SW ID 7 is used instead of 2 since it otherwise will cause problems during initialization because 2 is doubled.

4.7.2 Software Description of the FTS driver

Start the FTS driver

The function `startFtsAmpl()` is used to start the FTS driver from sensor server 2. In order to start the FTS driver each USB port must be initialized to communicate as a serial port. The used USB port settings, for the USB/RS845 converter, can be viewed in the following:

Data bit	=	8
Stop bit(s)	=	1
Baud rate	=	115,200
Flow control	=	None
Parity check	=	None

When the function `startFtsAmpl()` is called, it gets a file path to a USB-port which can be associate to a file descriptor(fd). When the USB-port is initialized the FTS amplifier is setup. To be sure that the FTS amplifier is ready to be initialized and to identify the FTS amplifier's SW ID, the FTS amplifier is restarted via a software command. When the FTS amplifier restarts it sends a 'Hello world' message with SW ID and the amplifier can be identified. After resetting the FTS amplifier it is setup to Speed Optimized Streaming Mode (SOSM). The settings for the FTS amplifier in SOSM can be viewed in the following:

Sample rate	=	1,250 Samples/sec.
Number of packages	=	Infinity.
Channel(s) to read from	=	Channel A and Channel B.

The FTS amplifier should respond with acknowledge when it enters the SOSM else the `startFtsAmpl()` function returns a `-1` which causes sensor server 2 to terminate.

Normal operation of the FTS driver

Sensor Server 2 uses the function `FtsAmplHandler()` after initialization. The function has 2 states, a state where it is out of `sync` and a state where it is in `sync`. First time the

function is running the FTS driver will not be synchronized, i.e. a known `sync` number is not in the beginning of the input buffer. In order to get synchronized the function looks through the input buffer to see whether 3 succeeding sync numbers have been received. I.e. at least 11 bytes must be received before it is possible to determine whether the driver is in sync or not. When the driver is in sync it will be possible to receive data. In case several messages are in the input buffer, only the newest whole message will be used. If the received sync number is equal to the expected sync number the messages is correct and it will be saved in the shared memory. In case anything goes wrong and the input buffer starts to grow, the buffer will be flushed and the function is not in sync. The entire driver is designed universal so it is possible to use the same driver for all the FTS amplifiers.

4.7.3 Test of the FTS driver

In order to verify the FTS driver it has been tested that data is send to Simulink from the amplifier. Furthermore the data in the input buffer has been analyzed when the `FtsAmplHandler()` is in sync and when it is not. In case the function is not in sync, an error messages will be printed in the command prompt.

Is has not been possible to test the throughput since one of the FTS amplifier in the left ankle is defect, i.e. it only changes its values with approximately ± 40 with a sample rate approximately on 0.2 Hz, where the other changes their values with severally thousands with a sample rate of 1250 Hz. Figure 4.13 is a graph of the sampled strain gauge signals in the 6 channels in the right FTS. Note that the sampled values are between $\pm$ 32768. The graph prove that the driver is able to configure the FTS amplifier such that data can be received.

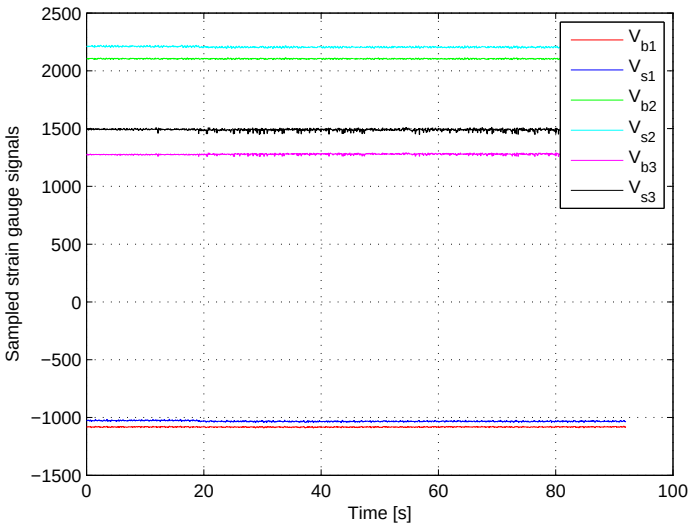


Figure 4.13: Sampled strain gauge signals in the right FTS

The throughput of the FTS is not verified, but to test the throughput it can be done by applying fast sine waves to the 12 channels, such that changes of the sampled strain gauge signals are ensured. Here after it is checked whether the values are changed for

each time step. If that is the case the throughput is ensured for that particular sampling rate.

4.8 Visulisation

When the model and controllers are developed they have to be tested in order to examine their performance. Mostly xy-graphs showing the joint angles of the **AAU-BOT1** is insufficient to determined whether the behavior is acceptable. Therefore the geometry orientation and position of the **AAU-BOT1** has to visualized. To accommodate this feature two methods are developed and is described in the following.

4.8.1 MATLAB^{TM} Plot Function

The first solution to the visualization problem was to use MATLAB^{TM} and use the plot function. The plotted data is generated by using the kinematic model developed in Section 5.4. This plot gives a nice overview of the **AAU-BOT1** and can be seen i Figure 4.14 where it is in an upright position. The black circles are the CoM for the individual links and the red circles are the position of the joints.

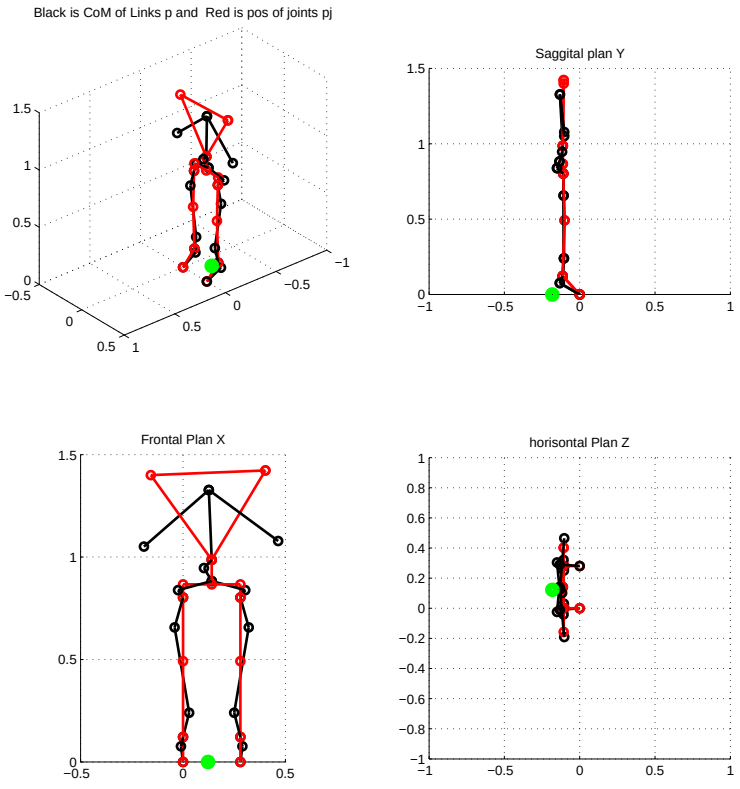


Figure 4.14: Matlab plot of the kinematics.

The Matlab plot works very well when inspecting the kinematics of the **AAU-BOT1**, but when the dynamical and real world aspect is introduced it gives problems. This is because it is very time consuming to e.g. program the dynamics that will enable the robot to slide over the floor or tilt over the edge of a foot. Different physics engines has already been developed by others e.g. for 3D games one of these engines will now be described.

4.8.2 Webots

Webots is a three dimensional robot simulator. It features its own physics engine, and it is possible to build a robot with different shapes, add motors and do all the control in Webots. The Webots program was initially developed as a research project, but is now a commercial product that can be bought.Figure 4.15 shows a screen shot of the Webots robot simulator with the custom build virtual **AAU-BOT1**.

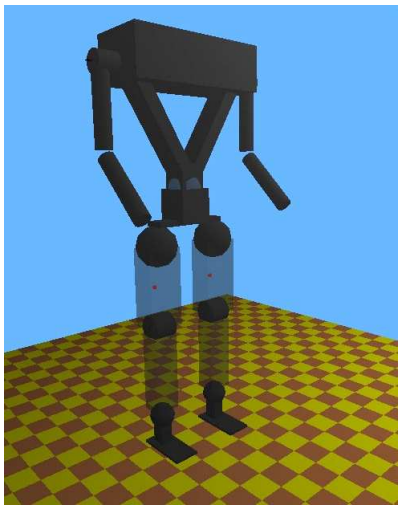


Figure 4.15: Screenshot from Webots of the virtual **AAU-BOT1**.

Interface

As Webots has its own robot simulator tool, the intention is to program the controllers for the robot in C code and thereby have it all as one embedded unit. Since the preferred controller development tool is Matlab^{TM} and Simulink, it is chosen to develop an interface between Webots and Simulink. The interface has to enable communication with Simulink and Webots to be synchronized. Furthermore it has to support two way communication.

Two method are considered when developing the interface. The first method is to program the interface as a C-code interface with shared memory. The second method considered is using a TCP/IP connection. Both interfaces are applicable, but the last solution is chosen since it features more opportunities which are elaborated on in the following. Webots is computational heavy and it is practically impossible to run both the Webots robot simulator program together with the main control loop in Simulink on one computer, unless the computer features a multi kernel CPU. A TCP/IP connection

enables communication via network and this has a big advantage as the Webots robot simulator can be located on a different system and be interfaced by the computer running the main control loop.

The interface between Webots and Simulink is developed as a master slave setup, where Webots act as the server and is waiting for connections. Simulink is the client and connects to Webots when the simulation is initiated. During each simulation step all the necessary data as joint angles and control input are transmitted. Each simulation step is ended by a message from Simulink to Webots, that initiates the next simulation step. This ensures that the robots simulator and the control loop are always synchronized.

The first part of the interface is the TCP/IP connection in Simulink. To enable Simulink to connect to a TCP/IP an S-fuction developed by [Rydesäter, 2003] is used. This is a TCP/IP Toolbox for MATLAB^{TM} and can be downloaded from MathWorks' homepage, but is also available on the enclosed CD-ROM together with the supervisor thread, the controller thread and the virtual **AAU-BOT1** in Webots.

In Figure 4.16 the TCP/IP connection between Webots can be seen. Furthermore the internal communication in Webots are displayed.

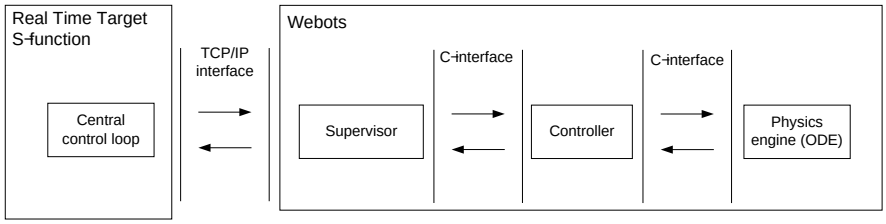


Figure 4.16: Interface between Webots and Simulink.

It is chosen to program a Supervisor and a Controller in the Webots environment theses are both programmed in C and are constructed as two separate threads. This is done since the controller thread only runs each simulation time step, but the supervisor thread runs as many times as needed to handle all the TCP/IP communication between Webots and Simulink. Furthermore the supervisor fetches all the necessary sensor data from the controller thread and forwards it to Simulink. It also handles the actuator data received from Simulink, this is transfered to the controller thread at each simulation step.

The controller thread updates the physics simulator with new actuator input and retrieves the sensor data from the physics environment. Furthermore the controller thread calculates the ZMP position which is displayed in the simulation environment such that stability can be viewed during simulation.

The programmed functions for Simulink are shown in Table 4.2, and can be found on the enclosed CD-ROM.

With these functions programmed a complete interface to the Webots simulation tool has been developed. Furthermore the configuration of the virtual **AAU-BOT1** in Webots has been carried out, a supervisor and a controller have been programmed such that the developed Simulink functions are supported. It all runs synchronized and Webots restarts the virtual environment each time Simulink is invoked for simulation.

Table 4.2: Simulink commands for Webots

Simulink function:	Explanation:
`get_theta_pos()`	Gets the angular position of joints
`get_theta_vel()`	Gets the angular velocity of joints
`get_theta_acc()`	Gets the angular acceleration of joints
`get_Pj_pos()`	Gets the position of links
`get_Pj_vel()`	Gets the velocity of links
`get_Pj_acc()`	Gets the acceleration of links
`get_P_pos()`	Gets the CoM position of links
`get_P_vel()`	Gets the CoM velocity of links
`get_P_acc()`	Gets the CoM acceleration of links
`get_zmp()`	Gets the position of ZMP
`get_torq()`	Gets the torque used by motors in Webots
`set_theta_pos()`	Set the angular position of joints
`set_theta_vel()`	Set the max allowable angular velocity of joints
`set_theta_acc()`	Set the max allowable angular acceleration of joints
`set_torq()`	Set the torque on the joints
`revert_webots()`	Reverting Webots for a new simulation
`step_forward()`	Stepping Webots one time step forward

4.9 Summary of Software

In this chapter, the designed software architecture for **AAU-BOT1** is documented. The first part of the software use a module based approach, where each program connects to a shared memory server. Two S-functions transmit and receive data from the shared memory to Simulink. An actuator server and two sensor servers interfaces with the hardware. This allows the Simulink model to run in real time, with the servers taking care of the communication with the hardware.

Additionally, a simulation program called Webots has been implemented, that enables testing of control algorithms in a risk-free environment.

A throughput test has been conducted. This showed that there was a loss in packages of 13.6% when using 250 Hz as sample frequency. Lowering the sample rate to 200 Hz lowered the packet loss to 1.9 % which is considered acceptable. If a packet is missing the previous packet will be used. Furthermore it should be noted that the packages loss is distributed over the whole timeframe of the test. This is appreciated as it does not influence the system as much as if the system lost all 13.6% packages in a row.

This software architecture is utilized in the control of **AAU-BOT1**.

Chapter 5

Modeling

*This chapter deals with the model derivation for **AAU-BOT1**. The purpose with the model is to give the necessary insight of the system to design, test and simulate controllers for **AAU-BOT1**. The models developed are: a DC motor model which are used for parameter estimation, a kinematic model is derived such that it is possible to determine the position, velocity, acceleration of the links and the CoM of links on the **AAU-BOT1**, given by the joint angles. A support phase estimator is proposed such that it can be determined in which of the phases described in Section 2 on page 23,**AAU-BOT1** is in. A foot model is also proposed such that forces and moments can be determined and used in the dynamical model. The dynamical model are derived in order include the dynamics of the **AAU-BOT1**. Finally an inverse kinematic model is derived to enable the derivation of walking trajectories.*

5.1 Introduction to Modeling

When modeling a biped robot many factors come into play. The two most crucial sub-models are the kinematic model and the dynamical model which constitutes the main part of the complete model. The kinematic model gives the positions, velocities and accelerations of the CoM for all the links according to changes of the joint angles. The dynamical model can be used to calculate the angular acceleration, velocity and position of the joints based on a input torque. This is based on the known forces, as gravity and coriolis force acting on the robot, but it does not include external unexpected forces.

Since the dynmic model is linear, only the viscous friction is included. In the DC motor model the stiction, the viscous and the coulomb friction is included.

5.2 Elements in the Model

The modeling are divided into several parts. The complete model of the **AAU-BOT1** can be seen in Figure 5.1.

It consist of sub models where the input and output can be seen for each model. The division of the models ensure that these can be developed and tested individually.

DC Motor model The DC Motor model is proposed such that it is possible to perform parameter estimation of **AAU-BOT1**. This is needed as the friction otherwise

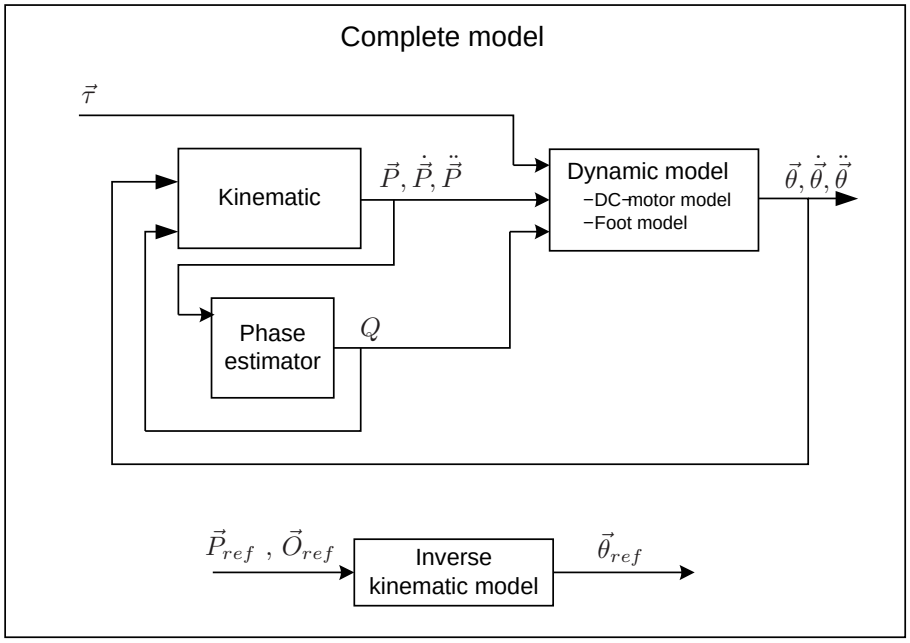


Figure 5.1: Block diagram of the complete model showing all the sub models with input and output.

would be unknown. The obtained parameters are used in the derivation of the Dynamical model.

Kinematic model The kinematic model is developed to obtain the position of link and the CoM of the links. This is based on the joint angles since the dynamical model

Dynamical model The dynamical model can be used to calculate the angular acceleration, velocity and position of the joints based on a input torque.

Phase Estimator The phase estimator is proposed as it is important for the kinematic model and Dynamical model to know in which phase it is in. Also a weight distribution between the feet is calculated as the Dynamical model uses this in the Dual support phases

Foot model The foot model calculates the forces and the torques that the ground exerts on the foot. The foot model is used in the dynamical model.

Inverse Kinematics The inverse kinematics is not an active part of the complete model seen in Figure 5.1, but is necessary in order to calculate trajectories for the joints.

In the following sections the models, except the foot model, will be derived starting with the DC Motor model. Since it is small part of the foot model that is used, it will be derived in Appendix D on page 188.

5.3 DC Motor Model

This section documents the model of the Servo Amplifiers and DC motors. The DC motor model is derived from physical and electrical considerations, and will be utilized in the dynamic model.

5.3.1 General Model

A DC motor can be modeled the electrical model of a DC motor and the physical model of a DC motor, see Figure 5.2(a) and 5.2(b) [Andersen and Pedersen, 2007, p. 8-10]. By using the Kirchoffs Current law, the motor current (i_M) can be calculated using (5.1). The angular velocity can be calculated using (5.2) and the torque on the motor can be found via Equation (5.3).

$$i_M = \frac{u - K_{\text{emf}}\,\dot{\theta}_M}{R_M + L_M s} \tag{5.1}$$

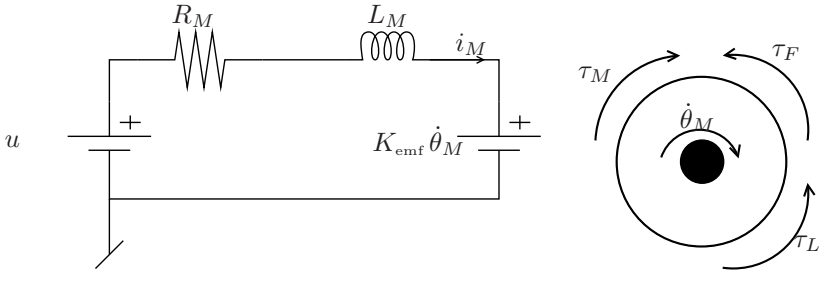
$$\ddot{\theta}_M = \frac{\tau_M - \tau_F - \tau_L}{J} \tag{5.2}$$

$$\tau_M = K_T \cdot i_M \tag{5.3}$$

where:

u	is the input voltage [V]
R_M	is the terminal resistance [Ω]
L_M	is the terminal inductance [H]
i_M	is the motor current [A]
J	is the total inertia of both the motor and the load [kg cm^2]
K_T	is the motor torque constant $\left[\frac{\text{Nm}}{\text{A}}\right]$, which has the same value as K_{emf}
K_{emf}	is the motor back emf voltage constant $\left[\frac{\text{Vs}}{\text{rad}}\right]$, which has the same value as K_T
τ_M	is the motor torque [Nm]
τ_F	is the friction torque [Nm]
τ_L	is the load torque [Nm]
$\ddot{\theta}_M$	is the angular acceleration of the DC motor shaft $\left[\frac{\text{rad}}{\text{s}^2}\right]$
θ_M	is the angle of the shaft [rad]
θ	is the angle of the link [rad]
G	is the gear ratio [ ]

Equation (5.1), (5.2) and (5.3) can be combined into the block diagram seen in Figure 5.3.



(a) Electrical diagram of a DC motor. (b) Model of the shaft.

Figure 5.2: DC motor model.

L_M has a relatively small value compared to R_M, i.e. the data sheets for the motors sets it to 0.513 mH, 0.870 mH and 0.329 mH for the 60 W, 90 W and 150 W motor, respectively [Maxon Motors, 2007d,e,c]. The value of L_M is set to 0 to prevent overmodelling as it has a negligible influence on the DC motor model at the frequencies used in this project. τ_L is the influence the physical system has on the motor, this will be found in the dynamic model. τ_F is elaborated on in the next section.

5.3.2 Friction of the DC Motor Model

The friction torque can be divided into three different states:

$$\tau_F = \begin{cases} \dot{\theta}_M \cdot \mu + \text{sign}(\dot{\theta}_M)\tau_c, & \text{if } \dot{\theta}_M \neq 0 \\ \tau_M - \tau_L & \text{if } (\dot{\theta}_M = 0) \wedge (|\tau_M - \tau_L| \leq (\tau_c + \tau_s)) \\ \text{sign}(\tau_M - \tau_L)(\tau_c + \tau_s) & \text{if } (\dot{\theta}_M = 0) \wedge (|\tau_M - \tau_L| > (\tau_c + \tau_s)) \end{cases} \tag{5.4}$$

where:

μ	is the viscous friction coefficient	$\left[\frac{\text{Nm s}}{\text{rad}}\right]$
τ_c	is the Coloumb friction torque	[N m]
τ_s	is the Stiction torque	[N m]

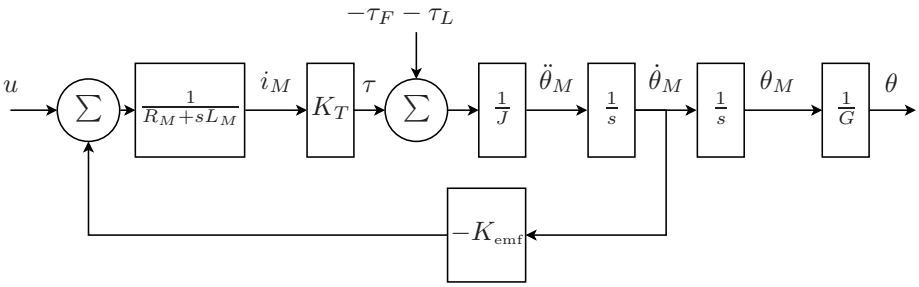


Figure 5.3: Block diagram of full model of DC motor.

5.3.3 Feedback Control of the Amplifiers

Due to the fact that the EPOS amplifiers allow several methods of control, some or most of the DC motor model can be simplified. As the amplifiers allow direct control of the current, the model can be reduced to Figure 5.4.

5.3.4 Gearing and Motor Constants

Since the gearing between the motor and the joints are taken into consideration in the motor model, the output from the motor models will be the angles of the joints and not the angle of the motor. The motor constants for the different motors can be seen in Table 5.1

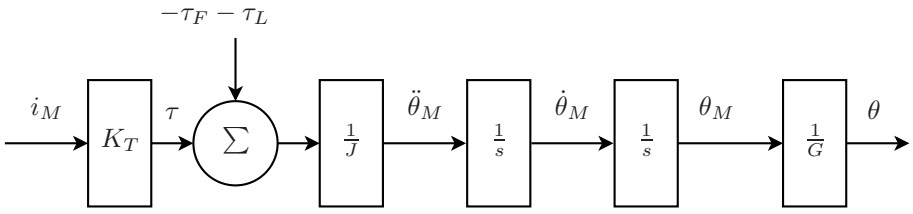


Figure 5.4: Block diagram of full model of DC motor and amplifier, with the EPOS amplifier in current mode.

Table 5.1: Motor specifications.

Motor size [W]	K_T **[$\frac{\mathbf{mNm}}{\mathbf{A}}$]**	J_m **[gcm^2]**	R_m **[Ω]**	**Nom. speed [rpm]**
60	53.8	34.5	2.52	7750
90	62.2	67.4	3.09	6490
150	60.3	138	1.16	7000

5.3.5 Double Actuated Joints

Six of the joints are actuated by two identical DC motors working together by pulling one belt which is connected to a gear-joint. In Figure 5.5 the setup of the hip roll joint actuation can be seen. To the left in the picture the SolidWorksdrawing of the hip dual motor system and to the right is the forces and moments on this setup.

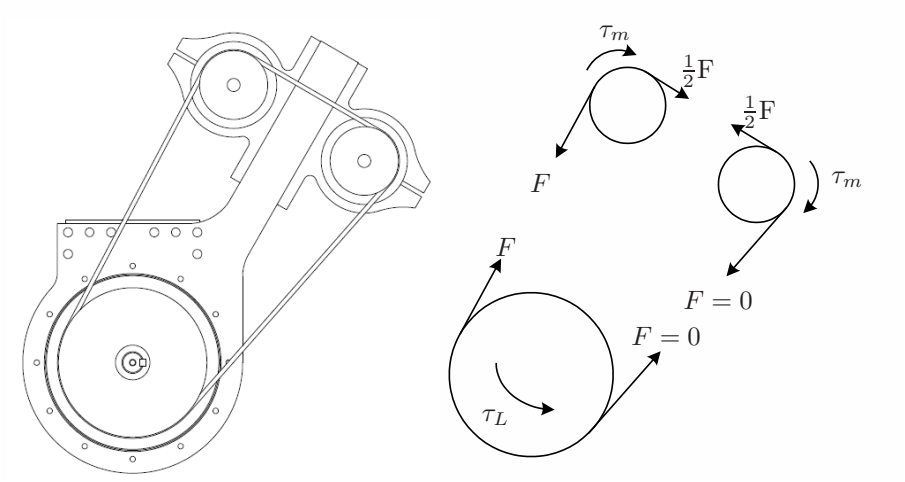


Figure 5.5: Force distribution in the double actuated joint system. τ_m are the torques from the motors and τ_L is the load torque or output torque in the joint

The other joints actuated by this dual motor system have a setup which is similar

to the one shown in Figure 5.5. In Table 5.2 it can be seen which joints are actuated by the double actuated joint system and which are not. To get an overview of the joint numbers Figure 5.9 on page 79 can be used. The reason why this double actuated joint system is chosen is simply to reduce weight. The weight of two smaller motors working in parallel is smaller than one large motor doing the same amount of work.

Table 5.2: Motor setup and joint speed.

Joint #	**Motor [W]**	**Dual actuated joint**	**Gear ratio**	**Max link vel.[$\frac{\mathbf{rad}}{\mathbf{s}}$]**
2	60	no	200	4.05
3	150	yes	320	2.29
4	150	yes	133	5.51
5	150	yes	288	2.55
6	150	no	257	2.85
7	90	no	250	2.72
8	90	no	250	2.72
9	150	no	257	2.85
10	150	yes	288	2.55
11	150	yes	133	5.51
12	150	yes	320	2.29
13	60	no	200	4.05
15	60	no	300	2.71
16	90	no	300	2.27
17	150	no	300	2.44
18	60	no	111	7.31
19	60	no	111	7.31

In order to model this double actuated joint, the force is considered as shown in Figure 5.5 on the previous page and some certain assumptions has to be made. In order to divide the total force by two, and thereby split the work evenly between the motors, the belt has to be nonelastic and no flexion must occur during operation. It is possible to tension the belts as wished and the belts are made for high precision servomotor drives [Pedersen et al., 2007]. This assumes that τ_L to the motors in the double actuated joints can be described as in Equation 5.5.

$$\tau_{L_{dual}} = \frac{\tau_L}{2} \tag{5.5}$$

This has to be included in the DC motor model for **AAU-BOT1**. The six joints that are actuated by two motors have the model as seen in Figure 5.6.

The double actuated joint model seen in Figure 5.6 on the facing page is similar to the regular model seen in Figure 5.4 on the previous page. The difference is that the system now has two inputs i_{M1} and i_{M2} and two output torques from the motors, τ_1 and τ_2. The intention with the system is that τ_1 and τ_2 are equal when in operation.

5.3.6 Actuation Limits

Since this is a model of a physical system with physical limitations, these considerations has to be included in the model. There are two different limitations added to the

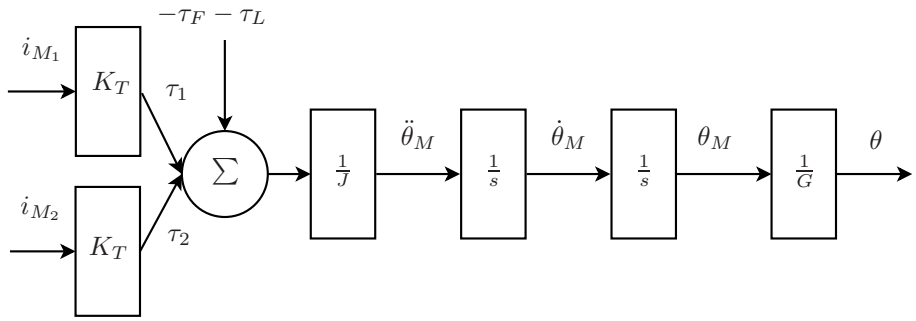


Figure 5.6: Block diagram of DC motor model and amplifier in the double actuated joints system.

motor model. The first limitations is based upon the datasheets for the maxon DC motors [Maxon Motors, 2007c], [Maxon Motors, 2007d] and [Maxon Motors, 2007e]. These limitations concern the maximum allowable velocities of the motors. These are converted to maximum angular velocities of the links and can be seen in Table 5.2

The other limitations are the maximum actuation of the joints. The physical constraints were initially defined by [Pedersen et al., 2007] and further limitations for the toes are added here as they where missing. The limits can be seen in Table 5.3. To get an overview of the joint numbers Figure 5.9 on page 79 can be used. With models and all the parameters and limitation in place the verification can be carried out.

5.3.7 Verification of DC Motor Model

The DC motor model is verified and parameter estimated using SENSTOOLS [Knudsen, 2004] on the left arm, see Appendix A.1. The parameters that have been estimated are the viscous friction, coulomb friction, inertia of the arm and τ_L. The stiction has not been possible to estimate, but this should not affect the final result, due to that stiction only affects the result in the start of the test.

The result from the test was that the final model has a mean squared error on 12.1%. According to [Knudsen, 2004], a mean squared error of only 5-8% should be obtainable. Due to the fact that the parameters are highly interconnected, the found parameters are perceived as correct and will be used in the model. For the same reasons the DC Motor Model is considered verified as being correct. See Figure 5.7 for a comparison of model output and system output.

Table 5.3: Maximum joint actuations.

Joint #	**Max pos. rotation [degree]**	**Max neg. rotation [degree]**
1	0°	-90°
2	15°	-10°
3	19°	-19°
4	1°	-95°
5	11°	-17°
6	11°	-81°
7	27°	-19°
8	27°	-19°
9	81°	-11°
10	11°	-17°
11	95°	-1°
12	19°	-19°
13	15°	-10°
14	0°	-90°
15	28°	-19°
16	11°	-13°
17	56°	-1°
18	6°	-33°
19	6°	-33°

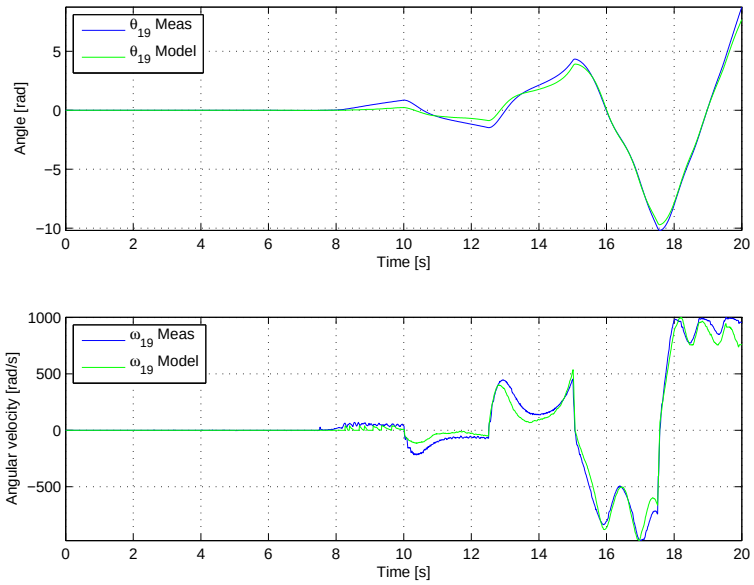


Figure 5.7: Output of parameter estimation test. The model has a mean squared error of 12.1%.

5.4 Kinematic Model

The kinematic concerns the relations between different joints on the robot and the position of the individual links. This is done by transforming from joint space to cartesian space. With this transformation the position of the links' CoM is calculated from the given rotation of the joints. Since the joints of the **AAU-BOT1** are not actuated directly with a DC-motor but through a certain gearing, a transformation from actuator space to joint space is necessary. This has been elaborated on in the DC-motor model in section 5.3. In Section 2.2 the different phases of the system is found. Even though the system can be in 14 different phases, only two phases are needed to describe the kinematic of the system and those are: Single Support Phase Left (SSP-L), Single Support Phase Right (SSP-R), this means that the robot is standing on one or the other foot. One must understand that the kinematic chains deduced for the two phases SSP-L and SSP-R can be used to describe the kinematic in all the other phases as well. This is because the kinematic always start from one toe and out to all limbs. In Dual Support Phase (DSP), the kinematic chain is described in SSP-L. The input to the kinematic model is the angular position of the joints θ together with the phase of the system $\vec{Q}$. The output is the globalized coordinates $\vec{P}$ which consist of a matrix $[\vec{x} \quad \vec{y} \quad \vec{z}]^T$. The transformation can be seen in figure 5.8. It is important to have a consistent notation, and representation of the mechanical system, the following will deal with representation of the mechanical system.

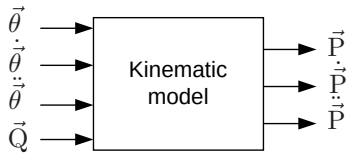


Figure 5.8: Kinematic model with joint angles as input and generalized coordinates as output.

5.4.1 Representation of the AAU-BOT1

The **AAU-BOT1** is a mechanical system with many joints and links. All the links are connected via the joints in a unique way such that it represents a humanoid robot. Since the **AAU-BOT1** is developed and manufactured at the section for mechanical systems, the complete mechanical layout is available in SOLIDWORKS made by [Pedersen et al., 2007]. SOLIDWORKS features a method to extract link vectors, CoM vectors, masses and moments of inertia for the desired links. In the SOLIDWORKS representation there exist a total of 32 body parts. This is because all the motors, joints and links are counted as body parts. The aim is to reduce the number of body parts by collaborating motor frames with the respective link and thereby reduce calculations.

Since the **AAU-BOT1** has been designed in SOLIDWORKS all the distances from CoM and to the joints can be extracted from this model. This data is already extracted by the former group working on the **AAU-BOT1**. The original data from [Pedersen et al., 2007] can be seen in Appendix C.2. By using this data and adding the right CoM vectors together the vectors from link to link can be calculated. The calculated data for the link-vectors are displayed in Table 5.4 together with the already existing CoM vector.

Table 5.4: Link- and CoM vectors extracted from SOLIDWORKS.

Link vectors [m]	**CoM vectors [m]**
$\vec{a}_1$= [0 0 0]^T	$\vec{b}_1$= [0 0 0]^T
$\vec{a}_2$= [-0.113 0 0.122]^T	$\vec{b}_2$= [-0.132 -0.010 0.076]^T
$\vec{a}_3$= [0 0 0]^T	$\vec{b}_3$= [0 0 0]^T
$\vec{a}_4$= [0 0 0.370]^T	$\vec{b}_4$= [0.004 0.030 0.118]^T
$\vec{a}_5$= [0 0 0.311]^T	$\vec{b}_5$= [-0.002 -0.041 0.165]^T
$\vec{a}_6$= [0 0 0]^T	$\vec{b}_6$= [0 0 0]^T
$\vec{a}_7$= [-0.003 0 0.064]^T	$\vec{b}_7$= [-0.041 -0.024 0.036]^T
$\vec{a}_8$= [0 0.280 0]^T	$\vec{b}_8$= [-0.022 0.140 0.016]^T
$\vec{a}_9$= [0.003 0 -0.064]^T	$\vec{b}_9$= [-0.038 0.024 -0.028]^T
$\vec{a}_{10}$=[0 0 0]^T	$\vec{b}_{10}$=[0 0 0]^T
$\vec{a}_{11}$=[0 0 -0.311]^T	$\vec{b}_{11}$=[-0.002 0.041 -0.146]^T
$\vec{a}_{12}$=[0 0 -0.370]^T	$\vec{b}_{12}$=[0.004 -0.030 -0.252]^T
$\vec{a}_{13}$=[0 0 0]^T	$\vec{b}_{13}$=[0 0 0]^T
$\vec{a}_{14}$=[0.113 0 -0.122]^T	$\vec{b}_{14}$=[-0.019 0.010 -0.046]^T
$\vec{a}_{15}$=[0 0 0]^T	$\vec{b}_{15}$=[0 0 0]^T
$\vec{a}_{16}$=[0 0 0.120]^T	$\vec{b}_{16}$=[-0.004 -0.038 0.080]^T
$\vec{a}_{17}$=[0 0 0]^T	$\vec{b}_{17}$=[0 0 0]^T
$\vec{a}_{ra}$=[0.006 -0.280 0.425]^T	$\vec{b}_t$=[-0.019 0 0.341]^T
$\vec{a}_{la}$=[0.006 0.280 0.425]^T	$\vec{b}_{ra}$=[0.004 -0.048 -0.347]^T
	$\vec{b}_{la}$=[0.004 0.048 -0.347]^T

The joints that have zero distance to each other are called multiturnable joints and can be modeled as two separate joints [Craig, 2005]. Therefore the link vector between two single joints considered a multiturnable joint are zero and so are the CoM vectors. This is illustrated in Figure 5.9-(a) where J_i are the joint numbers and a_i are the link vectors. The multiturnable joints are shown as two single joints together. These joints have the numbers: (J_2-J_3), (J_5-J_6), (J_9-J_{10}), (J_{12}-J_{13}), (J_{16}-J_{17}). All other joints are located by the link vectors defining the links between them. The link vectors have their origin at the previous joint, thereby link vector a_i originates from J_{i-1}.

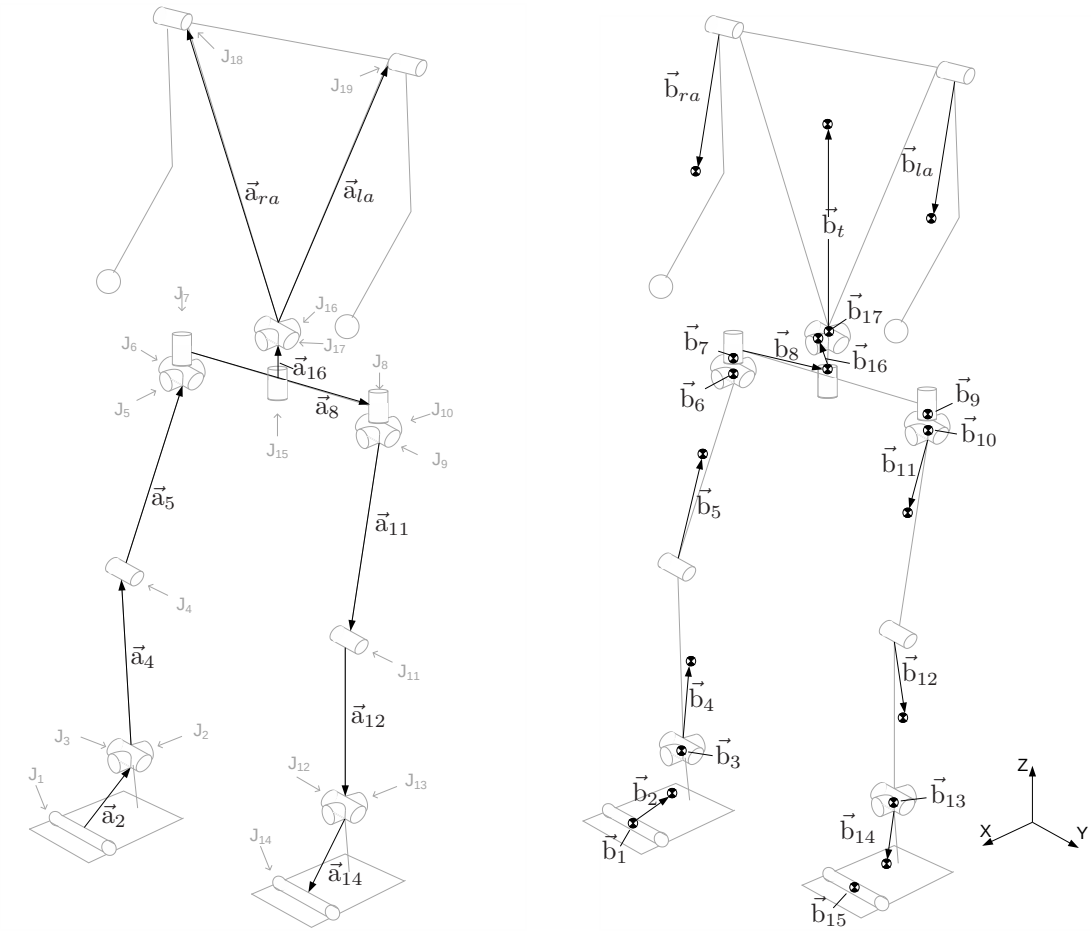
In figure 5.9-(b) the CoM of all the link bodies are illustrated. These have their origin in the previous joint just as the link vectors. The CoM vectors are used later when the inverse kinematics is described in Section 5.7 on page 91.

In Figure 5.10-(a) all the roll rotations around the$x$-axis is shown together with the zero angle and in which direction the rotation is positive. In Figure 5.10-(b) it is for all the rotations around the y-axis and in Figure 5.10-(c) it is the yaw rotations around the z-axis.

To find the position of all the links and their CoM, rotations have to be applied for every joint. This is done in the following.

5.4.2 Transformations

To be able to find which rotations that have to be applied through the model it is necessary to give every joint their own coordinate system. In [Christensen et al., 2006], the Denavit hardenberg method described in [Craig, 2005] is used, this is an old well-known method that is good for organizing all the rotations. The method is not always fully applicable without differing from the standard procedure described in [Craig, 2005].



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

(b) Showing the CoM vectors for the links.

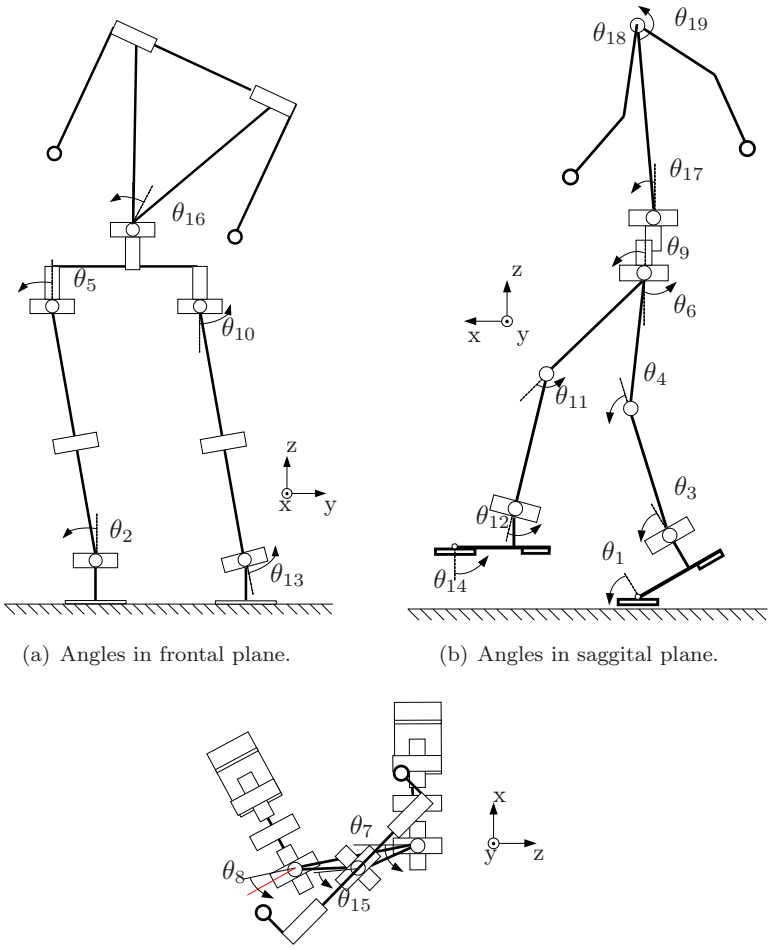
Figure 5.9: Joints, link vectors, CoM vectors.

Therefore it is chosen to use the same method as in [Christensen et al., 2007]. Here a coordinate system is placed in every single link with the origin in the previous joint. The origin of frame[i] is in that way placed in joint J_i, and is attached with the next link a_{i+1} and previous link a_i.

The frames are aligned with the global reference frame, when all the joints are in their zero state meaning the angles are zero ($J(\Theta_i) = 0$). This means that when all the angles of the joints are zero no rotations are needed to describe the system. This gives the advantage that when a rotation between two link-frames is carried out, it is described by the actual angle of the joint.

In order to keep track of the global position a ground frame is added to the model as it is seen on Figure 5.11. The position of the ground frame is fixed to the initial frame[0] such that the initial frame always contain the global position.

All the frames have the same direction when the angles are zero, but the joints are not always revolute in the same direction. This means rotations has to be made according to which direction the joint can be actuated. In Table 5.5 the different rotations for the



(a) Angles in frontal plane.

(b) Angles in saggital plane.

(c) Angles in transverse plane.

Figure 5.10: Angle of rotation in three three planes.

link-frames can be seen.

Where the rotations around x-, y-, z-axis is given as in equation (5.6), (5.7), (5.8) from [Craig, 2005, p. 49].

$$ {}^{i-1}_{i}\mathbf{R}_x(\theta_i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_i) & -\sin(\theta_i) \\ 0 & \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \tag{5.6} $$

$$ {}^{i-1}_{i}\mathbf{R}_y(\theta_i) = \begin{bmatrix} \cos(\theta_i) & 0 & \sin(\theta_i) \\ 0 & 1 & 0 \\ -\sin(\theta_i) & 0 & \cos(\theta_i) \end{bmatrix} \tag{5.7} $$

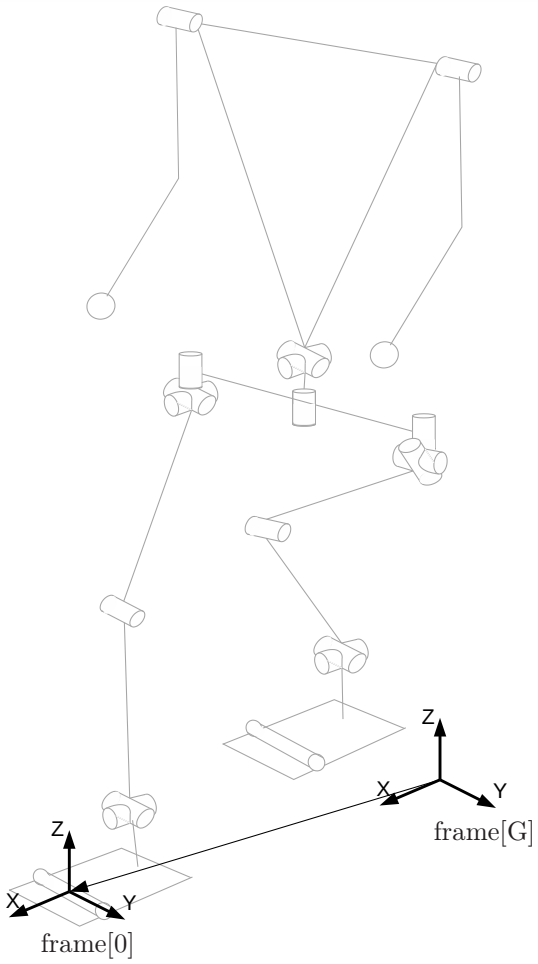


Figure 5.11: The earth frame[E] and the first frame[0] is the global coordinate system

Table 5.5: Rotations for the link-frames.

Rotation matrix:	**Link-frames$_i$:**
${}^{i-1}_{i}\mathrm{R}_x(\theta_i)$	2,5,10,13,16
${}^{i-1}_{i}\mathrm{R}_y(\theta_i)$	1,3,4,6,9,11,12,14,17,18,19
${}^{i-1}_{i}\mathrm{R}_z(\theta_i)$	7,8,15

$$ {}^{i-1}_{i}\mathbf{R}_z(\theta_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.8} $$

The robot is an open loop serial chain system that originates from the supporting foot as it is done in [Wu, 2003]. This means that the first rotation is in the supporting foot and the end rotations is in the non supporting foot, left arm and right arm. This gives three different open kinematic chains. The rotation matrices have to be multiplied together according to which of the three chains that are calculated. As elaborated on before the kinematics only needs to be defined in two different phases: SSP-L and SSP-R. Table 5.6 show the kinematic chains for SSP-R.

Table 5.6: Kinematic chains for SSP-R.

Kinematic chain	**for link no. i**
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	$i = 19$

When the system is in SSP-L the kinematic chains are as in Table 5.7.

Table 5.7: Kinematic chains for SSP-L.

Kinematic chain	**for link no. i**
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	$i = 19$

5.4.3 The Global Position

Every single link in the global position has to be determined. It is chosen to describe the global position as the position for the CoM of the links. The vectors containing the CoM position for all the links is denoted $\vec{P}$.

$$ \vec{P} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \qquad \text{where } i = 1, 2, ..., 20 \tag{5.9} $$

The global position of the CoM of the links can now be found, starting with the supporting foot and forward to the link needed just as described in [Craig, 2005, p. 28]. The position of CoM for each link$_i$ is found like:

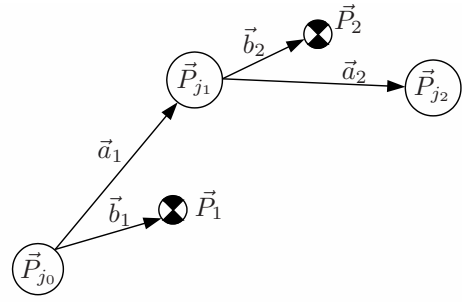


Figure 5.12: Example of rotation as used with the robot.

$$\vec{P}_i = \vec{P}_{j_{i-1}} +^0_{i-1} \vec{R}\, \vec{b}_i \tag{5.10}$$

where $\vec{P}_{j_{i-1}}$ is:

$$\vec{P}_{j_{i-1}} = \vec{P}_{j_{i-2}} +^0_{i-2} \vec{R}\, \vec{a}_{i-1} \tag{5.11}$$

where $\vec{P}_{j_{i-1}}$ is the global position of the joint $i-1$ and $\vec{b}_{i-1}$ is the local position of the CoM for link $i-1$.

From equation 5.10 and 5.11 the global coordinates can be found, just as it is shown in the motivating example on figure 5.12. Furthermore a detailed derivation of a motivating example with a simple robot with 3 DOF can be seen in appendix E.

Since the kinematic model also returns the global velocity vector and the global accelerations vector these need to be calculated and this is done in the following, desciped in [Craig, 2005, page 135]:

$$\dot{\vec{P}} = \frac{\partial \vec{P}}{\partial t} \quad , \quad \ddot{\vec{P}} = \frac{\partial \dot{\vec{P}}}{\partial t} \tag{5.12}$$

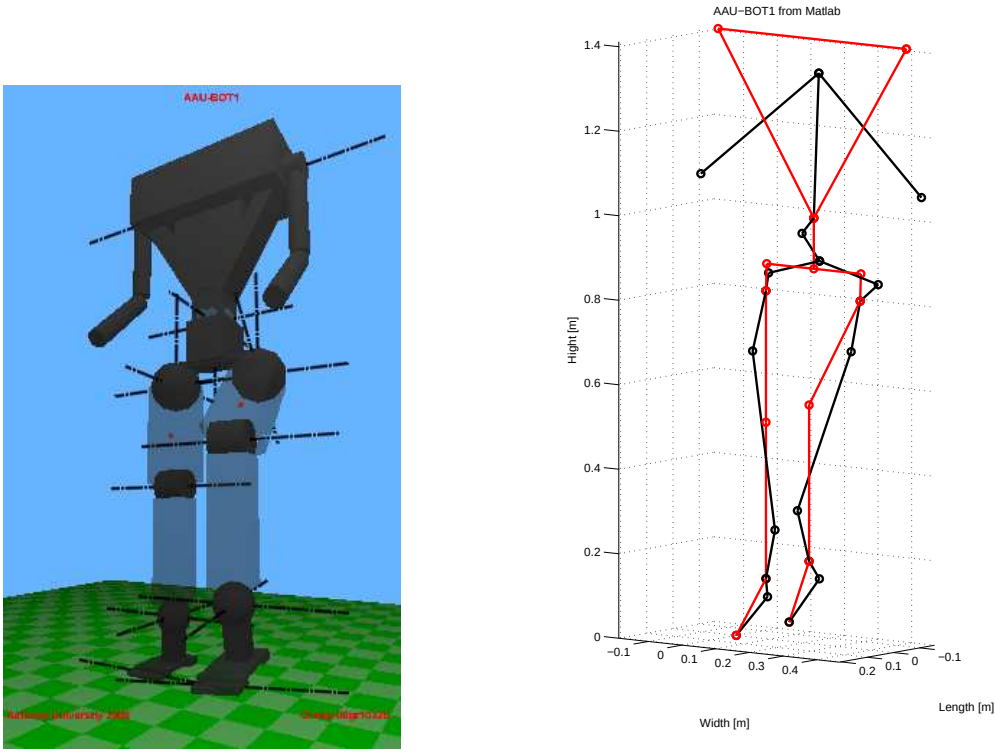
Now the kinematic model is derived, the global positions and their derivative can be found. Due to their size, these are found on the CD.

5.4.4 Verification of Kinematic Model

The Kinematic model is verified in Appendix A.2, using Webots. The largest error in the measurements where 0.004 m, which is regarded as an error in Webots. The output of Webots is compared to the Kinematic Model in Figure 5.13.

5.5 Dynamic Model

The purpose of the dynamic model is to determine the angle acceleration $\ddot{\vec{\theta}}$, in the individual joints. By integrating $\ddot{\vec{\theta}}$ it is possible to find $\dot{\vec{\theta}}$ and $\vec{\theta}$. $\ddot{\vec{\theta}}$ are used to obtain knowledge about the movement of **AAU-BOT1**, i.e. it can be used to determine the ZMP and the energy consumption. The dynamic model of **AAU-BOT1** is constructed as a hybrid model of the different support phases, as the basis for the model is different



(a) Webot's representation of AAU-BOT1 in SSP-R.

(b) Matlab plot of AAU-BOT1 in SSP-R. The red dots represent the position of joints and the black dots are the CoM.

Figure 5.13: Visual result by applying the test angles for SSP-R.

in each of the phases. The dynamic model for **AAU-BOT1** in SSP is derived first, and is used as the basis for the model of **AAU-BOT1** in DSP. In Figure 5.14 the input and output relation for the dynamic model is shown. Due to the size of the dynamic model, the model will only be treated symbolic, which is why a motivating example has been derived in Appendix E on page 193.

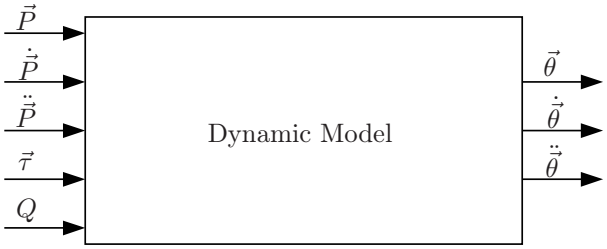


Figure 5.14: Block diagram of the Dynamic model.

5.5.1 Dynamic Model of AAU-BOT1 in SSP

The dynamics of **AAU-BOT1** in SSP are derived by using the Lagrange-d'Alembert equation[Craig, 2005, p. 183]:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = \mathcal{F}_i \tag{5.13}$$

where:
$\mathcal{L}$ is the Lagrangian for **AAU-BOT1** in SSP
q_i is system state i
$\mathcal{F}_i$ is external force i for **AAU-BOT1** in SSP
The state vector $\vec{q}$ consists of the position vector $\vec{P}$ and the angles θ are as:

$$\vec{q} = \begin{pmatrix} \vec{P} \\ \theta_1 \\ \vdots \\ \theta_{19} \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ \vdots \\ x_{20} \\ y_{20} \\ z_{20} \\ \theta_1 \\ \vdots \\ \theta_{19} \end{pmatrix} \tag{5.14}$$

The Lagrangian ($\mathcal{L}$) is defined as:

$$\mathcal{L} = E_{\text{kin}} - E_{\text{pot}} \tag{5.15}$$

where:
E_{kin} is the kinetic energy of the system
E_{pot} is the potential energy of the system

The kinetic energy is calculated using [Craig, 2005, Eq. (6.69) and (6.70), p. 182] to:

$$\begin{aligned} E_{\text{kin}} &= \frac{1}{2}\left(\sum_{i=1}^{N_{\text{Links}}} m_i \dot{P}_i^2 + \vec{\omega}_i^{\text{T}} \boldsymbol{J}_i \vec{\omega}_i\right) \\ &= \frac{1}{2}\left(\sum_{i=1}^{N_{\text{Links}}} m_i(\dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2) + \vec{\omega}_i^{\text{T}} \boldsymbol{J}_i \vec{\omega}_i\right) \end{aligned} \tag{5.16}$$

where:

N_{Links}	is the number of links
$\vec{\omega}_i$	is the angular velocity vector of link i
$\boldsymbol{J}_i$	is the inertia tensor of link i, around the CoM of the link
$\vec{P}_i$	is the position vector of the center of mass of the ith body
m_i	is the mass of the ith body

The toe spring is modeled as potential energy. The springs in the feet are described in Appendix D on page 188 where the foot model is derived. The total potential energy is calculated as:

$$E_{\text{pot}} = \sum_{i=1}^{N_{\text{Links}}} m_i g z_i + k_t(\theta_1 + \theta_{14}) \tag{5.17}$$

where:

g is the gravitational acceleration constant.

Inserting Equation (5.16) and (5.17) into Equation (5.15) yields:

$$\mathcal{L} = \frac{1}{2}\left(\sum_{i=1}^{N_{\text{Links}}} m_i(\dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2 - 2gz_i) + \vec{\omega}_i^{\text{T}} \boldsymbol{J}_i \vec{\omega}_i\right) \tag{5.18}$$

The angular velocities ($\vec{\omega}_i$) can be found using [Craig, 2005, Eq. (5.43), p. 146]:

$$\vec{\omega}_i = \vec{\omega}_{i-1} +_0^i \boldsymbol{R}\dot{\theta}_i \zeta \tag{5.19}$$

where:

$\vec{\zeta}$	is a vector that denotes the axis which $\dot{\theta}_i$ rotates about

As **AAU-BOT1** has several kinematic chains, that varies with the walking phase, the velocity vector of the previous link is found using table 5.6 and 5.7 on page 82. The external forces given in Equation (5.13) are calculated in steps, firstly, the Lagrangian is

differentiated with regards to the states:

$$\frac{\partial \mathcal{L}}{\partial \vec{q}} = \begin{pmatrix} 0 \\ 0 \\ m_1 g \\ 0 \\ 0 \\ m_2 g \\ \vdots \\ 0 \\ 0 \\ m_{20} g \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{5.20}$$

The second part of Equation (5.13) is derived and differentiated with respect to time:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\vec{q}}} = \begin{pmatrix} \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_1} \\ \vdots \\ \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_{79}} \end{pmatrix} \tag{5.21}$$

The result from this is omitted due to size. To map Equation (5.21) to the actuators, the Jacobian is used[Craig, 2005, p. 186]:

$$\boldsymbol{J}_F(\vec{\theta}) = \frac{\partial \vec{q}}{\partial \vec{\theta}} \tag{5.22}$$

$$\boldsymbol{J}_F(\vec{\theta}) = \begin{bmatrix} \frac{\partial q_1}{\partial \theta_1} & \cdots & \frac{\partial q_1}{\partial \theta_{19}} \\ \vdots & \ddots & \vdots \\ \frac{\partial q_{79}}{\partial \theta_1} & \cdots & \frac{\partial q_{79}}{\partial \theta_{19}} \end{bmatrix} \tag{5.23}$$

The kinematic model for the positions are inserted and the equation is differentiated partially. The mapping is then done by:

$$\begin{aligned} \vec{\tau} &= \boldsymbol{J}_F^{\mathrm{T}}(\vec{\theta})\mathcal{F} \\ &= \boldsymbol{J}_F^{\mathrm{T}}(\vec{\theta}) \left(\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\vec{q}}} \right) - \frac{\partial \mathcal{L}}{\partial \vec{q}} \right) \end{aligned} \tag{5.24}$$

where:

$\vec{\tau}$ is the torque excerted on the links by the DC Motors and the springs in the toes.

5.5.2 State Space Formulation

If the equation of motion is brought into a state space formulation state space control theories can be applied. The equation of motion from Equation (5.22) can also be represented in another form. This form of the dynamic equation can be seen in equation (5.25) as described in [Craig, 2005, p. 185].

$$\vec{\tau} = \boldsymbol{M}(\vec{\theta})\ddot{\vec{\theta}} + \boldsymbol{V}(\vec{\theta}, \dot{\vec{\theta}}) + \boldsymbol{G}(\vec{\theta}) \tag{5.25}$$

where:

$\vec{\tau}$	is the torque excerted by the DC motors
$\vec{\theta}$	is the angle of the joints
$\boldsymbol{M}(\vec{\theta})$	is the mass/inertia matrix
$\boldsymbol{V}(\vec{\theta}, \dot{\vec{\theta}})$	contains the centrifugal and coriolis terms
$\boldsymbol{G}(\vec{\theta})$	is the gravity terms

$\vec{\tau}$ consists of the torque added by the DC motors and the torque excerted by the friction of the system. The friction is modeled using Equation (5.26).

$$\vec{\tau}_F(\dot{\vec{\theta}}) = \text{diag}(\vec{\mu})\dot{\vec{\theta}} \tag{5.26}$$

where:

$\vec{\mu}$	is the viscous coefficient constant.

Inserting $\vec{\tau} = \vec{\tau}_M - \vec{\tau}_F$ into Equation (5.25) yields:

$$\vec{\tau}_M - \vec{\tau}_F(\dot{\vec{\theta}}) = \boldsymbol{M}(\vec{\theta})\ddot{\vec{\theta}} + \boldsymbol{V}(\vec{\theta}, \dot{\vec{\theta}}) + \boldsymbol{G}(\vec{\theta}) \tag{5.27}$$

Isolating $\ddot{\vec{\theta}}$ yields in Equation (5.28), where $\vec{\tau}_F$ is found via the DC motor model in Section 5.3.2 on page 72.

$$\ddot{\vec{\theta}} = \boldsymbol{M}^{-1}(\vec{\theta})\left(\vec{\tau}_M - \vec{\tau}_F - \boldsymbol{V}(\vec{\theta}, \dot{\vec{\theta}}) - \boldsymbol{G}(\vec{\theta}) - \boldsymbol{F}(\dot{\vec{\theta}})\right) \tag{5.28}$$

The expression in Equation (5.28) is inserted into a embedded MATLAB$^{\text{TM}}$ function in Simulink (see Figure 5.15, and is linearized numerically, using the MATLAB$^{\text{TM}}$ function `linmod`.

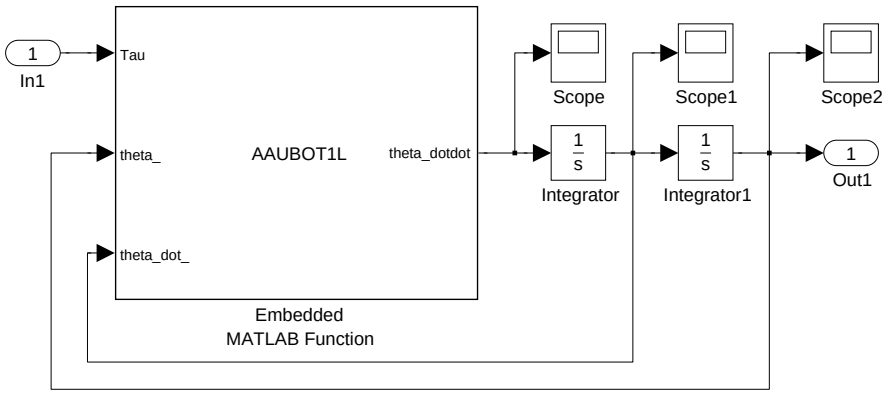


Figure 5.15: Nonlinear model in simulink.

The linearized system has the following general form:

$$\begin{bmatrix} \dot{\vec{\theta}} \\ \ddot{\vec{\theta}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}^{19\times19} & \boldsymbol{I}^{19\times19} \\ \boldsymbol{A}_{2,1} & \boldsymbol{A}_{2,2} \end{bmatrix} \begin{bmatrix} \vec{\theta} \\ \dot{\vec{\theta}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \tau_{\vec{M}} \end{bmatrix} \tag{5.29}$$

$$\vec{y} = \begin{bmatrix} \boldsymbol{I}^{19\times19} & \boldsymbol{0}^{19\times19} \end{bmatrix} \begin{bmatrix} \vec{\theta} \\ \dot{\vec{\theta}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0}^{19\times19} \end{bmatrix} \begin{bmatrix} \tau_{\vec{M}} \end{bmatrix} \tag{5.30}$$

where $\boldsymbol{A}_{2,2}$ is negative definite and $\boldsymbol{B}$ is positive definite, which is as expected.

5.5.3 Dynamic Model of AAU-BOT1 in DSP

The dynamics of **AAU-BOT1** in DSP are computed as a combination of SSP-L and SSP-R, by using Equation (5.31):

$$\vec{\tau}_{\text{DSP}} = \rho\vec{\tau}_L + (1-\rho)\vec{\tau}_R \tag{5.31}$$

where:

$\vec{\tau}_{\text{DSP}}$	is the torque of the individual links of **AAU-BOT1** in DSP
$\vec{\tau}_L$	is the torque of the individual links of **AAU-BOT1** in SSP-L
$\vec{\tau}_R$	is the torque of the individual links of **AAU-BOT1** in SSP-R
ρ	is the weighting between the SSP-L and SSP-R

ρ is calculated by calculating the distance from the origo of each foot to the ZMP (ZMP is calculated using Equation (2.7) and (2.8) on page 30):

$$\rho_1 = \frac{\left\|\vec{P}_R - \vec{P}_{\text{ZMP}}\right\|}{\left\|\vec{P}_L - \vec{P}_{\text{ZMP}}\right\| + \left\|\vec{P}_R - \vec{P}_{\text{ZMP}}\right\|} \tag{5.32}$$

$$\rho = \begin{cases} 1 & \rho_1 \geq 1 \\ \rho_1 & 0 < \rho_1 < 1 \\ 0 & \rho_1 \leq 0 \end{cases} \tag{5.33}$$

where:

$\vec{P}_{\text{ZMP}}$	is the position of the ZMP
$\vec{P}_L$	is the position of the left foot
$\vec{P}_R$	is the position of the right foot

5.5.4 Summary of Dynamic Model

A dynamic model of **AAU-BOT1** was derived using Lagrangian mechanics, Jacobian transformations, the kinematic model and the DC motor model. The result is a hybrid state space model with 38 states, 17 inputs and 19 outputs. This model will be used as basis for the designed controllers. The dynamic model has not been possible to verify since there are parameters at the physical **AAU-BOT1** that yet are unknown, and the safety system has not been built in such extent that it is justified to verify the dynamic model on it. Furthermore the representation of **AAU-BOT1** in Webots does not work as expected, which is why the dynamic model neither is verified in Webots.

5.6 Support Phase Estimator

In Chapter 2 the different phases of the system is specified. It is important for the system to know which leg is the supporting one. Furthermore it makes a difference whether the system is standing on the toe or not or the system is in a heal impact phase. Especially for the kinematic model it is important to know which leg is the supporting one, so the correct kinematic chains can be chosen.

5.6.1 Hybrid Systems

The hybrid system section is based on [Bak and Izadi-Zamanabadi, 2004, p. 11]. A hybrid system can be formulated as a hybrid automaton H:

$$H = (\vec{Q}, \vec{X}, \text{Init}, \text{f}, \text{Dom}, \text{E}, \text{G}, \text{R}) \tag{5.34}$$

$\vec{Q}$ contains the discrete events which is defined as the different phases of the system. Even though all possible phases have been listed only q_1, q_2,q_3, q_4, q_5 and q_6 are implemented in this thesis.

$$\vec{Q} = \begin{cases} q_1 \triangleq \text{SSP-L} & : \text{Single Support Phase left} \\ q_2 \triangleq \text{SSP-R} & : \text{Single Support Phase right} \\ q_3 \triangleq \text{SSP-L-T} & : \text{Single Support Phase left toe only} \\ q_4 \triangleq \text{SSP-R-T} & : \text{Single Support Phase right toe only} \\ q_5 \triangleq \text{DSP-L} & : \text{Double Support Phase left} \\ q_6 \triangleq \text{DSP-R} & : \text{Double Support Phase right} \\ q_7 \triangleq \text{DSP-L-T} & : \text{Double Support Phase left toe} \\ q_8 \triangleq \text{DSP-R-T} & : \text{Double Support Phase right toe} \\ q_9 \triangleq \text{DSP-L-TH} & : \text{Double Support Phase toe and heel left} \\ q_{10} \triangleq \text{DSP-R-TH} & : \text{Double Support Phase toe and heel right} \end{cases} \tag{5.35}$$

$\vec{X}$ contains the continuous variables for the joint angles:

$$\vec{X} = \vec{\theta_{19}} \ , \ \vec{X} \in \Re^{19} \tag{5.36}$$

Init contains the initial condition of the system at time zero:

$$Init = \theta_1 = 0 \ , \ \theta_2 = 0 \ , \ \ldots \ , \ \theta_{19} = 0 \ , \ \vec{Q} = \text{DSP-L} \tag{5.37}$$

All the Double Support Phases are a combination of two single support phases. The phase is determined by evaluating the states in the model, using Equation (5.38). This equation is made from simple observations of the individual states.

$$\vec{Q} = \begin{cases} q_1 & \text{if} \quad z_r > \epsilon_1 \wedge z_l < \epsilon_2 \wedge O_{llp} < \epsilon_4 \wedge \theta_{14} < \epsilon_6 \\ q_2 & \text{if} \quad z_r < \epsilon_1 \wedge z_l > \epsilon_2 \wedge O_{rlp} < \epsilon_4 \wedge \theta_1 < \epsilon_5 \\ q_3 & \text{if} \quad z_r > \epsilon_1 \wedge z_l < \epsilon_2 \wedge O_{llp} < \epsilon_4 \wedge \theta_{14} > \epsilon_6 \\ q_4 & \text{if} \quad z_r < \epsilon_1 \wedge z_l > \epsilon_2 \wedge O_{rlp} < \epsilon_4 \wedge \theta_1 > \epsilon_5 \\ q_5 & \text{if} \quad z_r < \epsilon_1 \wedge z_l < \epsilon_2 \wedge O_{rlp} < \epsilon_3 \wedge O_{llp} < \epsilon_4 \wedge \theta_1 < \epsilon_5 \wedge \theta_{14} < \epsilon_6 \wedge x_r < x_l \\ q_6 & \text{if} \quad z_r < \epsilon_1 \wedge z_l < \epsilon_2 \wedge O_{rlp} < \epsilon_3 \wedge O_{llp} < \epsilon_4 \wedge \theta_1 < \epsilon_5 \wedge \theta_{14} < \epsilon_6 \wedge x_r > x_l \\ q_7 & \text{if} \quad z_r < \epsilon_1 \wedge z_l < \epsilon_2 \wedge O_{rlp} < \epsilon_3 \wedge O_{llp} < \epsilon_4 \wedge \theta_1 > \epsilon_5 \wedge x_r < x_l \\ q_8 & \text{if} \quad z_r < \epsilon_1 \wedge z_l < \epsilon_2 \wedge O_{rlp} < \epsilon_3 \wedge O_{llp} < \epsilon_4 \wedge \theta_{14} > \epsilon_6 \wedge x_r > x_l \\ q_9 & \text{if} \quad z_r < \epsilon_1 \wedge z_l < \epsilon_2 \wedge O_{llp} > \epsilon_4 \wedge \theta_1 > \epsilon_5 \wedge x_r < x_l \\ q_{10} & \text{if} \quad z_r < \epsilon_1 \wedge z_l < \epsilon_2 \wedge O_{rlp} > \epsilon_3 \wedge \theta_{14} > \epsilon_6 \wedge x_r > x_l \end{cases} \tag{5.38}$$

where:

ϵ_n	is threshold n, determined experimentally with **AAU-BOT1**
z_r	is the z-value of the right foot.
z_l	is the z-value of the left foot.
O_{llp}	is the pitch of the left foot.
O_{rlp}	is the pitch of the right foot.
θ_1	is the angle of the right toe.
θ_{14}	is the angle of the left toe.

The Phase Estimator will be used with the kinematic model and the dynamic model, to create a hybrid model. The verification is done concurrent with the system test described in Chapter 8 on page 143.

5.7 Inverse Kinematics

The derivation of the kinematic model was done in Section 5.4 and the purpose was to describe the positions of all the links in cartesian space given the joint angles. The purpose of the inverse kinematic model is to determine the joint angles which yield a specific position of the limbs in cartesian space. The inputs to the inverse kinematic are all the positions and orientations of the legs $\vec{P}_{t_r}, \vec{O}_{t_r}, \vec{P}_{t_l}, \vec{O}_{t_l}$ and the orientation of the torso and arms $\vec{O}_w, \vec{O}_{a_r}, \vec{O}_{a_l}$ In Figure 5.16 the position vectors for the inverse kinematics can be seen.

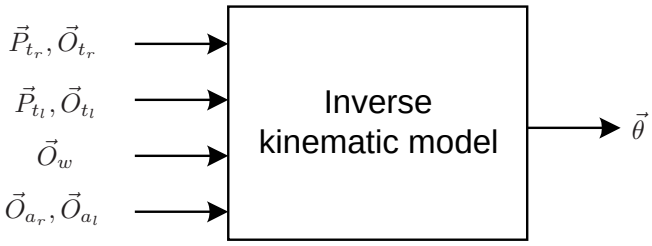


Figure 5.16: Block diagram of input and output for the inverse kinematic model.

Different methods have been developed to accommodate the inverse kinematic problem. Two widely used methods are the numerical solution and the analytical solution. One numerical solution is used by [Goldenberg et al., 1985] is done by solving the pseudo inverse Jacobian in order to transform the desired position of a limb into joint angles. As the first solution of this method does not always deliver an accurate outcome, this method is done in an iterative way to ensure a precise outcome. This method has been carried out by [Christensen et al., 2007] on a biped robot. They concluded that this method serves its purpose, but the downside is that the transformation of the position to joint angles on one leg with 6 DOF takes 209s for five iterations on a 2.4 GHz P4 computer. This is only for one leg and if the rest of the links are considered as well, it would result in an even higher calculation time. This demonstrates that this method is a very heavy computational task. Since it is assumed that the control loop of the **AAU-BOT1** is supposed to be much faster than the time it takes to calculate the inverse kinematic the numerical way, then this method is discarded.

The other method considered are an analytical method. This method is also called closed-form solution and can be solved directly as a resolution of the non-linear equations and is only supported on robot manipulators with few DOF [Craig, 2005]. Often robot manipulators are designed in such a way that a solution to the inverse kinematic problem exist. A great advantage is that it is very fast computational vise and moreover a robust method. The downside is that is only works on robot manipulators with up to 6 DOF. This method is preferable and with the right assumptions this method can be applied the **AAU-BOT1**, which is described in the following.

5.7.1 Closed Form Solution

The main idea with the closed form solution is to solve the inverse kinematic problem by geometric inspection. The position of the feet and orientation of the torso are all determined by the orientation of the legs. It is therefore crucial that the legs are in the

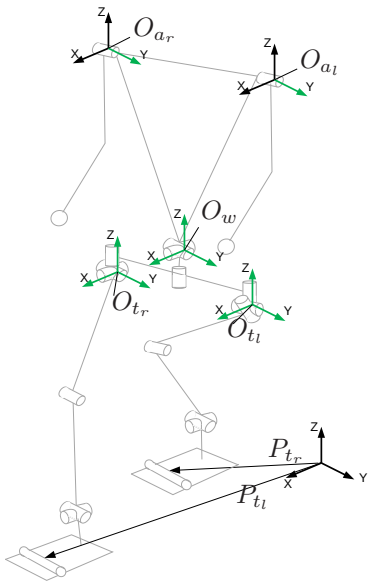


Figure 5.17: Position vectors and orientations in the inverse kinematic.

right position with the right orientation in order to obtain the right posture of torso. The different position vectors and orientations can be seen in Figure 5.17.

As mentioned before the closed form can only be used on kinematic chains of up to 6 links. Where the last three links are one revolute joint [Craig, 2005]. But as it can be seen in Figure 5.17 one leg consist of 7 joints and this includes the toe joint. Since the toe joint is unactuated and can be influence by e.g. moving the leg in a toe of phase trajectory such that the toe will be bended according to the angle between the foot and the ground, when the toe is not in contact with the ground or any other objects, is pulled back to its origin by a spring. To perform a toe of phase it is important to know the exact position and orientation of the concerning foot. This concludes that the toe is actuated indirectly by the movement of the leg, foot-base and interaction with the ground.

This means that in the inverse kinematic problem for the legs can be reduced to a inverse kinematic problem with 6 joints from the foot base and to the hip. Hence, the closed form solution to the inverse kinematic problem can be utilized for the legs. Besides from the legs the torso can also be orientated in the x-y-y axis and the arms can only be rotated around the y-axis. This method is called Pieper's solution and is described in [Craig, 2005]. In Figure 5.18 the axis of rotation for the torso, the arms and the hips can be seen and is indicated by green vectors. Furthermore the vectors used in the closed form solution to the inverse kinematics can be seen in the figure. The method of Pieper is applied in the following.

Inverse Kinematic for the Legs

The problem is split up into two main tasks. Assuming that the feet are on the ground the first task is to derive the angles of the first three joints which are the ankle roll, ankle pitch and knee pitch joints. The specific angles for the desired position of the hip can be obtained with this task. The next task is to give the hip the desired orientation

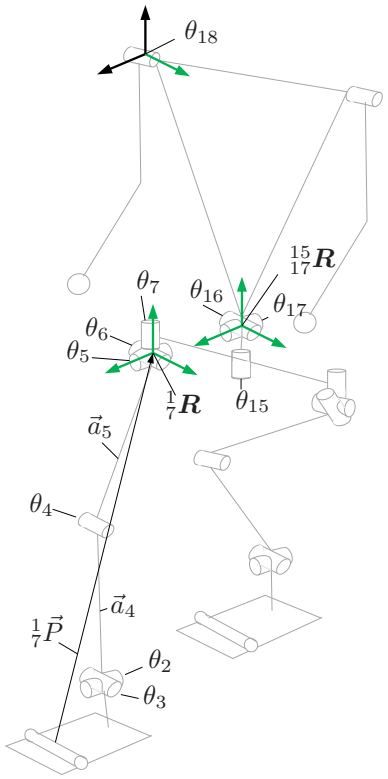


Figure 5.18: Used vectors and rotations in the inverse kinematic. The green axis are the axis of rotation in the concerned joint.

by a x-y-z rotation with the three intercepting joints at the top of the leg connecting the hip. The position from the toe to the revolute joint starting at θ_5 is given as in Equation (5.39).

$$ {}^1_5\vec{p} = {}^1_2\boldsymbol{T}\,{}^2_3\boldsymbol{T}\,{}^3_4\boldsymbol{T}\,{}^4_5\vec{p} \tag{5.39} $$

Here ${}^1_2\boldsymbol{T}$ describes the rotation and position of frame 2 relative to frame 1 and so on. The reason why the first transformation matrix is not ${}^0_1\boldsymbol{T}$ is because the rotation in the toe is not included as described before. Equation (5.39) can according to Pieper's solution to [Craig, 2005] be written as in Equation (5.40):

$$ {}^1_5\vec{p} = {}^1_2\boldsymbol{T}\,{}^2_3\boldsymbol{T} \begin{bmatrix} f_1(\theta_4) \\ f_2(\theta_4) \\ f_3(\theta_4) \\ 1 \end{bmatrix} \tag{5.40} $$

where f is a function that gives the position of joint 5 relative to joint 4. The function f only contains θ_4 and can be expressed as in Equation (5.41)

$$ \vec{f}(\theta_4) = {}^3_4\boldsymbol{T}\,{}^4_5\vec{p} \tag{5.41} $$

if $\vec{f}(\theta_4)$ is written out it will look as in Equation (5.42) - (5.44)

$$ f_1 = s_4 a_{5_z} \tag{5.42} $$
$$ f_2 = 0 \tag{5.43} $$
$$ f_3 = c_4 a_{5_z} + a_{4_z} \tag{5.44} $$

The squared magnitude of $\vec{f}(\theta_4)$ can be calculated as Equation (5.46) and is furthermore the length squared of ${}^1_5\vec{p}$:

$$ r^2 = f_1^2 + f_2^2 + f_3^2 \tag{5.45} $$
$$ = s_4^2 a_{5_z}^2 + c_4^2 a_{5_z}^2 + a_{4_z}^2 + 2c_4 a_{5_z} a_{4_z} \tag{5.46} $$

This can be rewritten to Equation (5.47)

$$ r^2 - a_{5_z}^2 - a_{4_z}^2 = c_4(2a_{5_z} a_{4_z}) \tag{5.47} $$

Now the first angle can be found by isolating θ_4 and Equation (5.48) is found

$$ \theta_4 = \arccos\left(\frac{r^2 - a_{5_z}^2 - a_{4_z}^2}{2c_4 a_{5_z} a_{4_z}}\right) \tag{5.48} $$

By writing out ${}^1_5\vec{p}$ in terms of ${}^1_2\boldsymbol{T}\,{}^2_3\boldsymbol{T}$ and $\vec{f}(\theta_4)$ the right side of Equation (5.49) is found. Note that ${}^1_5\vec{p}$ is the desired position of the leg.

$$ {}^{1}_{5}\vec{p} = \begin{bmatrix} x_{\text{desired}} \\ y_{\text{desired}} \\ z_{\text{desired}} \end{bmatrix} = \begin{bmatrix} c_3 f_1 + s_3 f_3 \\ s_2(s_3 f_1 - c_3 f_3) \\ -s_3 c_2 f_1 + c_3 c_2 f_3 \end{bmatrix} \tag{5.49} $$

Here it can be seen that the x component only depends of θ_3 and θ_4 in f_3, but since θ_4 is known θ_3 can be used by using the arctan2 function in MATLAB^{TM} and subtracting θ_3:

$$ \theta_3 = \left(\arctan2(f_3, f_1) \pm \arctan2\left(\sqrt{f_1^2 + f_2^2 - x_{\text{desired}}^2}, x_{\text{desired}} \right) \right) \tag{5.50} $$

Now θ_2 can be found since it is the only unknown factor in the y-component, θ_2 is found in Equation (5.51)

$$ \theta_2 = \arcsin\left(\frac{y_{\text{desired}}}{s_3 f_1 - c_3 f_3} \right) \tag{5.51} $$

Now the angles of the joints for the first three joints are found for the position of the hip. Now the angles of the last three joints has to be found in order to obtain the orientation of the hip. The last three joints are mounted as a x-y-z rotation given as ${}^{5}_{7}\boldsymbol{R}$ and can be calculated by:

$$ {}^{5}_{7}\boldsymbol{R} = {}^{1}_{5}\boldsymbol{R}^{-1}\, {}^{1}_{7}\boldsymbol{R} \tag{5.52} $$

The rotation of the last three links is a rotation around θ_5, θ_6, θ_7 and Equation (5.52) can be rewritten to Equation (5.53)

$$ \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c_6 c_7 & -c_6 s_7 & s_6 \\ s_5 s_6 c_7 + c_5 s_7 & -s_5 s_6 s_7 + c_5 c_7 & -s_5 c_6 \\ -c_5 s_6 c_7 + s_5 s_7 & c_5 s_6 s_7 + s_5 c_7 & c_5 c_6 \end{bmatrix} \tag{5.53} $$

It can be seen that θ_6 can be expressed as in Equation (5.54)

$$ \theta_6 = \arcsin(r_{13}) \tag{5.54} $$

Since θ_6 is known θ_7 can be found by using r_{11} and r_{12} of Equation (5.53) [Craig, 2005]:

$$ \theta_7 = \arctan2(\frac{r_{12}}{-c_6}, \frac{r_{11}}{c_6}) \tag{5.55} $$

θ_5 can be found using r_{23} and r_{33} from Equation (5.53) [Craig, 2005]:

$$ \theta_5 = \arctan2(\frac{r_{23}}{-c_5}, \frac{r_{33}}{c_5}) \tag{5.56} $$

Now all the angles for the joints are found such that it is possible to find position and orientation of the right hip. Note that if **AAU-BOT1** is in left phase it is the foot which is positioned and orientated. The same procedure is used when deriving the inverse kinematics for the left leg θ_8 - θ_{13}

Inverse Kinematics for Torso and Arms

Since the torso is connected to the legs through the waist and hip the overall position is determined by the inverse kinematics for the legs and the orientation of the waist. In Figure 5.18 the hip is located by ${}^{1}_{7}\vec{p}$ thereafter waist can be located by using the regular link vectors found in Table 5.4. With the waist located the x-y-z rotation can be performed according to the desired orientation of the torso O_t. From the waist the arm can easily be located again by using the regular link vectors. Both arms can only be rotated around the Y axis as show with the green axis in Figure 5.18.

5.7.2 Verification of Inverse Kinematics Model

The inverse kinematics are verified by setting the output of the inverse kinematics to the kinematics and seeing whether the resulting output is equal to the input. The largest error between the inputs and the output was smaller than 0.6 mm, meaning that the inverse kinematics are regarded as correct. This is described i detail in Appendix A.3.

5.8 Summary of Modeling

In this chapter a DC motor model has been derived for single actuated joints and for double actuated joints. In order to verify the DC motor model, parameter estimation for one arm has been derived as well. The DC motor model has been derived in such an extent that it only had a mean square error of 12.1%, and the DC motor model is assumed to be correct.

The kinematic model is derived such that the global positions, velocity and acceleration of the individual link's and their CoM's is found given the joint angles. The model has been verified towards the simulation tool Webots and the largest error in this test was 0.004 m. The error is considered as being uncertainties in Webots and the kinematic model is therefore assumed to be correct.

A dynamic model of **AAU-BOT1** has been derived using Lagrangian, Jacobian transformations, the kinematic model and the DC motor model. The result is a hybrid state space model with 38 states, 17 inputs and 19 outputs. Due to missing safety systems at **AAU-BOT1** and problems with Webots it has not been possible to verify the model.

In order to determine the support phase **AAU-BOT1** is in, a support phase estimator has been derived. The phase estimator is utilized by the dynamical model, foot model, and the kinematic model.

An inverse kinematic model has been derived and verified. By dividing the kinematic chain into parts that are only 6 links long, a unique solution to a position and orientation of a link is found. This has been utilized by the trajectory generation, which will be elaborated on in the following chapter.

Chapter 6

Trajectory generation

This section describes different approaches to generate trajectories, i.e. online, offline or in a combination. Hereafter different methods for trajectory generation are described, i.e. should the trajectory be generated from human trajectories, the highest stability, the lowest energy consumption or the CoM method. The outline for this master's thesis is to walk static gait and dynamic gait, so a method is developed to generate these two types of gait. However to avoid replications the dynamic gait trajectory is described in Appendix F on page 203. Last in this chapter, the result of the trajectory generation is presented and a summary of the entire chapter is given.

6.1 Trajectory Generation Requirements

One of the challenges by moving a humanoid robot is to find suitable trajectories for each joint. During the last 25-30 years many strategies have been developed for deriving trajectories. In order to choose the right strategy all the constraints/features of **AAU-BOT1** regarding to trajectory generation must be defined. The requirements are deduced from [Pedersen et al., 2007] and Chapter 3 on page 31:

- **AAU-BOT1** is dimensioned to be able to walk dynamical as a human.
- **AAU-BOT1** do not have the same energy efficiency and are not able to store energy as a human.
- **AAU-BOT1** should be able to walk with an average speed of 1 m/s.
- It should be possible to implement it on the on-board computer.

One of the more essential parts of generating trajectories is to investigate whether the generation of trajectories are done online or offline. This will be elaborated on the following section.

6.1.1 Online Trajectory Generation

Online gait trajectory generation is calculated on the fly during walk. One of the advantages of using online trajectories is that unexpected changes in the terrain will not affect the walk of the robot more than it will affect a human. Which means the trajectories will always be optimal according to energy or/and stability depending on the terrain.

Furthermore online trajectories will also be able to correct errors, i.e. if there is flexibility in the mechanical design or the terrain is soft. A soft terrain will cause the foot to strike the ground before or after it was expected. If offline generated trajectories are used, it will result in deviations from the intended trajectory and stability might also be reduced.

Many papers have been written about online trajectories gait generation, but many of them require high computational power [Kondak and Hommel, 2003a], they are presented on very simple models or they simple have failed to implement an algorithm that is able to work on a real systems. In [Kondak and Hommel, 2003a] a method is presented that generate online trajectories in simulation. However this method does not consider torque limits in the motors. In order to understand how difficult online generation of trajectories are, a study trip to Massachusetts Institute of Technology (MIT) was arranged. [Frazzoli, 2008] demonstrated how MIT's Darpa Urban Challenge[1] car generates online trajectories. This car has a very complex sensor system to get knowledge about the surroundings and 10 Quad-core computers to handle the sensor data, trajectory generation and control. Even though the cars is computational well equipped, it still has to slow down or stop some time in order to find and follow a correct trajectory. Based on the field trip to MIT and [Kondak and Hommel, 2003a] it has been concluded that it will be infeasible to use online generation of trajectories on **AAU-BOT1**, because it would require cameras or/and radars to get knowledge of the surroundings. The on-board computer do not either have enough computational processor power and an extra computer will be needed.

6.1.2 Offline Trajectory Generation

Offline trajectory generation is well known and often used in trajectory generation. Offline trajectory generation means that some or all of the trajectories for each joint are calculated before the walking experiment. One of the advantages of using offline trajectory generation is that it is possible to achieve an optimal trajectory, i.e. a trajectory that looks like human dynamic gait, which use minimum energy and still has high stability. However this requires a very accurately model of **AAU-BOT1** in order to get useful trajectories that can be implemented on **AAU-BOT1**. Furthermore offline generated trajectories sometimes result in unstable walking [Wollherr et al., 2003]. In [Wollherr et al., 2003] the problem was solved by making an online compensation. This type of trajectory generation will be discussed in Section 6.1.3. As mentioned in 6.1.1 on the preceding page it requires a controller structure that maintain stability to the system during walk, because flexibilities in the mechanical parts and noise in the sensors sometimes cause the robot to plant the foot before or after it was scheduled. This problem also increases the energy consumption since the trajectories would not be ideal.

Offline generated trajectories will be best suitable for **AAU-BOT1**. This will be a good way to test how the controller performs and it will be possible to combine these with adaptive generated trajectories as described in Section 6.1.3. Using offline generation also means that the trajectories have to be recalculated when the real **AAU-BOT1** model is known and when the hardware is mounted in order to get an accurate weight and inertia distribution for it.

6.1.3 Hybrid Trajectory Generation

A Hybrid Trajectory Generation combines offline and online trajectory generation. It is usually offline generated trajectories that is being modified online. Quite often the tra-

[1] An autonomous car which participate in the Darpa Urban Challenges.

jectory is tuned online, either to minimize the energy consumption, increase the stability or to be able to follow a rough terrain.

[Bebek and Erbatur, 2003] has presented a way, where the stability of a biped robot is tuned via neuro-fuzzy during walk to increase the stability. By training the neuro-fuzzy algorithm it was slowly able to increase stability on one joint, however the algorithm was slow and it was not possible to increase the stability on all joint simultaneously.

Another approach is a central pattern generator(CPG), which also often are used for adaptive, self-improving algorithms to realize human-like, energy efficient motion [Takahashi et al., 2005]. The adaptive property improves the robustness of the robot during environmental changes and the external disturbance, and a biped robot will be able to walk naturally on the sophisticated ground. The downside of using CPG is, that it is computational demanding and the presented method [Takahashi et al., 2005] is only applied on a 5 linked biped robot.

Improving trajectories by adaptive tuning would be a good solution to minimize the energy consumption and increase the stability. The focus in this project is to make to **AAU-BOT1** walk for the first time and not to optimize it, adaptive tuning will not be implemented in this iteration.

6.2 Different Trajectory Generation Approaches

6.2.1 Inverted Pendulum

Biped robots are often modeled as inverted pendulums [Wollherr, 2005], [Park and Kim, 1998], [Kajita et al., 2003], to reduce the complexity of the mechanical model. In [Wollherr, 2005] a 3D-Linear Inverted Pendulum Method is presented, where the mass of the robot is assumed to be concentrated in one point. Therefore the CoM is considered to remain in a fixed position relative to the main body of the robot, this assumption is justified when the main body, the head and the arms amount is more than 70% of the total robot mass. Furthermore the motion range of the legs is limited, hence the error caused by this assumption is treated as a disturbance and is suppressed by the controller traking the trajectoies. Using the inverted pendulum method also assumes that there is no ankle torque applied at the base of the pendulum. The 3D inverted pendulum method has also been used by [Pedersen et al., 2007]. Their result of the CoM and ZMP can be seen at Figure 6.1 on the following page.

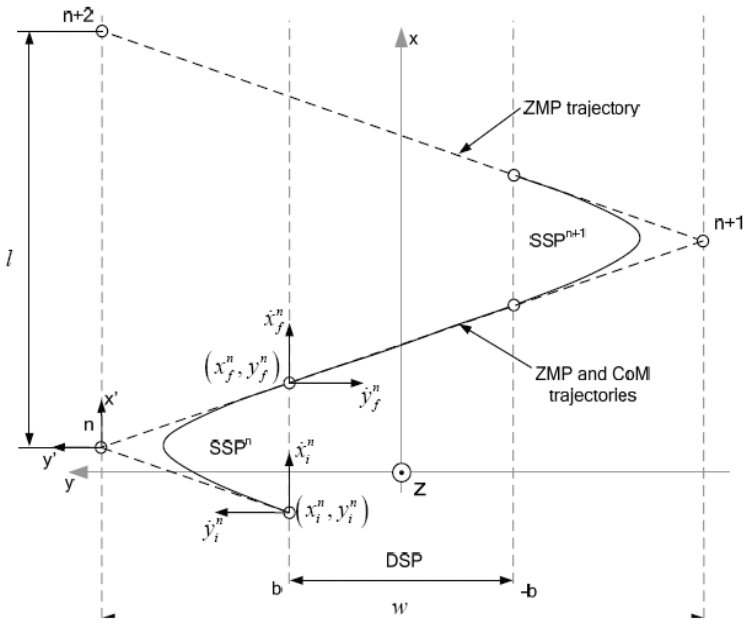


Figure 6.1: Sketch of CoM and ZMP trajectory [Pedersen et al., 2007].

AAU-BOT1 do not have 70% of its weight in the torso, arms and pelvis, which why the method is not used.

6.2.2 GCoM Method

The GCoM method is often used to generate trajectories. The idea with the GCoM methods is to walk very slowly, which means that GCoM and ZMP are the same. The method works by moving the CoM from one foot to the other one and vice versa. It is a relative simple way of getting a stable walk, but it has several disadvantages, e.g. this method only makes it possible to walk static and not dynamic gait as a human. In [Christensen et al., 2006] the technique was utilized on the first biped robot at Aalborg University. It resulted in a duck-like walk [Christensen et al., 2006]. This method also has high energy consumption. However for the first test of biped robots this method is often used since the trajectories are very slow, which makes it easier to follow the individual joints to verify whether they follow the trajectory correct.

6.2.3 Human Trajectories

One of the obvious ways of designing trajectories for biped robots is by using recorded human trajectories. Humans are said to have the most energy optimized walk and is it stable. In [Pedersen et al., 2007] experiment with motion capture of a human trajectory and use this for simulation. The results of these experiment were poor and the group ended up with using an own developed trajectory. Looking at human motion graphs in the book [Vaughan et al., 1992] supports the solution in [Pedersen et al., 2007] to used a different method than human trajectories, because humans are very different in size and weight. If human trajectories should be used for a biped robot trajectory, the biped robot should have the exact same dimensions and weights as the human, whose motions

are captured. Furthermore there is no guarantee that a trajectory based on a human would be energy optimal, since the human anatomy is very complex and it is impossible to make a biped robot exactly as a human with exiting technology.

6.2.4 Energy Efficiently Walk

One of the major concerns with autonomous biped robots is the amount of energy consumption during walk. Human walk is defined as the most energy efficiency walking type [Vaughan et al., 1992]. In [Pedersen et al., 2007] the **AAU-BOT1**is designed to operate in 15 minus and therefore **AAU-BOT1** energy consumption has to be low. [Tedrake, 2008] and [McGeer, 1990] have researched in passive walkers. The principle of passive walker is that it should be able to walk downhill of a slope by using the gravity and inertia to produce forward motion. This type of motion do not have any actuated joints, it must walk downhill and **AAU-BOT1** has high friction in the joints, which is why it cannot be implement on **AAU-BOT1**.

Another approach is a method develop by [Djoudi et al., 2005]. They have suggested a method where they use optimal cyclic joint reference trajectories for the walking of an under-actuated biped. The optimal trajectory is defined by a small number of parameters. The joint evolutions of the legs are fourth order polynomial functions of a scalar path parameter. By using a dynamical model, some constraints as limits on torque and velocity an optimal walking trajectory is found. It will be possible to use parts of this method on the **AAU-BOT1**. However **AAU-BOT1** have 17 actuated joints, where the presented method in [Djoudi et al., 2005] do not have actuated ankle joints, so the methods must be modified if it should be used.

6.2.5 ZMP Stability

Generation of trajectories, where the ZMP stability is considered, is one of the most popular trajectory generation methods. In [Choi et al., 2004] and in [Huang et al., 2000] methods to generate different trajectories with high stability are presented. The used principle is to simulate different trajectories and chose the trajectory with the highest stability.

Another popular method is seen in [Huang et al., 2001] which has focus on increasing the ZMP stability while walking. The paper describes a method for planning walking patterns, which includes the ground conditions, dynamic stability constraints, and relationship between walking patterns and actuator specifications. The method can be applied directly to a biped a robot, using constraints from human trajectories. The method use a stability margin, which ensure that the trajectory is not only marginal stable, but it can have a certain degree of stability. The principle of the technique is to develop one predefined trajectory for the feet, and then adjust the hip motion in the vertical and the horizontal direction, such that high stability is gained. It means severally changes of the hip parameters are simulated. The different trajectories are found via Cubic Spline Interpolations, which ensures that the trajectory is smooth and it do not result in any sudden high changes in the ZMP. Unfortunately this method only includes ZMP stability and might result in trajectory that is not energy optimal, even though the trajectory is created from a human trajectory. This is due to the mechanical design of a biped robot which is not as complex as the human anatomy.

6.3 Establishing Trajectories for AAU-BOT1

The previous section discussed different methods of generate trajectories. During this thesis dynamic and static gait is designed, dynamic in simulation and static gait at **AAU-BOT1**, which is why two trajectory generation strategies have been developed. It is decided to use offline trajectory generation, because the on-board computer does not have enough computational power to do online trajectory generation. The hybrid trajectory generation is not used because of lack of time, but it will be possible for future research groups working at the **AAU-BOT1** project to use the developed offline trajectory generation method to a hybrid solution.

The dynamic gait trajectory generation will combine several methods described in Section 6.2 on page 99. The ZMP-stability method is used to ensure stability, to ensure that the **AAU-BOT1** moves similar to a human. To make the trajectory as smooth as possible and to reduce the energy consumption, some parts of energy optimal gait is used. Real dynamic gait is hard to realize, that is why it has been decided only to implement the toe-off phase and not the heal-strike phase during dynamic gait, as described in Section 1.4 on page 18. To ensure that the legs are long enough to take a normal step when heal-strike phase is not used, the step length is reduced by 18 cm. However the dynamical model is not complete, and the trajectory cannot be simulated correct, but only the design method is derived. To avoid replication it will be described in Appendix F on page 203.

The static gait trajectory generation also combines several methods described in Section 6 on page 97. The CoM method (which is the same as the ZMP-stability method during static gait) and the human trajectory generation method are used to generate the trajectories. In order to design humanoid trajectories, the parameters have be chosen as seen in Table 6.1 on the next page and in Table 6.2 on the facing page. The parameters are found in [Huang et al., 2001], [Pedersen et al., 2007] and [Vaughan et al., 1992]. The mechanical design have angle limits in each joints, which why the static gait trajectories are very slow and the step length is small, otherwise it would be infeasible to create it.

Furthermore knowledge about **AAU-BOT1** has to be obtained before the speed is increased. The trajectory generation is calculated in Cartesian space, i.e. the origo is in the left toe. The following sections describe the developed method to obtain stable walk trajecories and the results are described in Section 6.4 on page 112. This section will also describe things that are common for static gait and dynamic gait.

6.3.1 ZMP Stability

The stability of **AAU-BOT1** is very important. In order to extract trajectories with the highest stability, more than 50000 different trajectories have been simulated, given the parameters in Table 6.1 on the facing page and Table 6.2 on the next page. Each simulation has small variation in the trajectory [Huang et al., 2001]. By using the inverse kinematic, the kinematic and the ZMP estimator it is possible to simulate walk trajectories. The stability definition for the biped robot can be seen below:

- During static gait, the GCoM must be within the support area. The GCoM and the ZMP is the same.

Table 6.1: Parameters defining a static gait trajectory

Parameters	**Values**
Step length[m], l_{step}	0.10
Averages speed[$\frac{m}{s}$], V_{speed}	0.002
Cycle time[s], T_{cycle}	50
Step time[s], T_{step}	0.001
SSP time[s], T_{SSP}	$T_{step} \cdot 0.6$
DSP time[s], T_{DSP}	$T_{step} \cdot 0.40$
Max ankle height[m], h_{amax}	0.20
Ankle height max time[s], T_{max}	$T_{step} \cdot 0.6$
Ankle height max length[m], l_{max}	$l_{step} \cdot 0.5$
Torso max[m], h_{tmax}	0.9452
Torso min[m], h_{tmin}	0.9152
Toe angle during toe off[degree]	0

Table 6.2: Parameters defining a dynamic gait trajectory

Parameters	**Values**
Step length[m], l_{step}	1.10
Averages speed[$\frac{m}{s}$], V_{speed}	1
Cycle time[s], T_{cycle}	1.1
Step time[s], T_{step}	0.55
SSP time[s], T_{SSP}	$T_{step} \cdot 0.8$
DSP time[s], T_{DSP}	$T_{step} \cdot 0.2$
Max ankle height[m], h_{amax}	0.22
Ankle height max time[s], T_{max}	$T_{step} \cdot 0.45$
Ankle height max length[m], l_{max}	$l_{step} \cdot 0.45$
Torso max[m], h_{tmax}	0.9702
Torso min[m], h_{tmin}	0.9102
Toe angle during toe off[degree]	16

- During dynamic gait, the ZMP must be within the support area. The GCoM is not the same as ZMP.

The area and the shape of the support area (SA) depends on how large step that **AAU-BOT1** has to walk with. Furthermore the shape also depends on what type of gait that is intended. As described in Section 1.4 on page 17 the scope for this thesis is to design **AAU-BOT1**, such that it is possible to walk dynamical in simulation and static gait with **AAU-BOT1**. This also results in a different support area. In order to maximize the support area, the toe feature has been disabled, i.e. the foot and the toe is considered as one plate. Figure 6.2 on page 105 is the SA of SSP. As it can be seen at the figure, support area in built of severally coordinates (x_i, y_i), which is marked with the black circles. Between each of the SA coordinates, there are vectors, which will be defined as edge vectors $\vec{v_i}$. In order to calculate whether the ZMP points are inside the SA or outside the SA as in Figure 6.2 on page 105, the perpendicular distance from the edge of the SA to a ZMP point is calculated via Equation (6.1) [WolframMathWorld, 2004]. Using this equation will always result in a positive distance, which is why Equation (6.2) is used to obtain negative distance as well.

$$d_i = \left| \vec{\hat{v}_i} \bullet \vec{r_i} \right| = \frac{|(x_{i+1} - x_i)(y_i - y_{zmp}) - (x_i - x_{zmp})(y_{i+1} - y_i)|}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}} \tag{6.1}$$

$$d_{i,stabMarg} = \frac{(x_{i+1} - x_i)(y_i - y_{zmp}) - (x_i - x_{zmp})(y_{i+1} - y_i)}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}} \tag{6.2}$$

where:

i	is 1,2,...,k
k	is the number of SA coordinates.
d_i	are the normalized perpendicular distance between the vectors $\vec{v_i}$ ZMP.
$d_{i,stabMarg}$	is the stability margin, positive if the ZMP is within the support area otherwise negative.
$\vec{v_i}$	are vectors in the edge around the support area.
$\vec{r_i}$	is a vector from the ZMP point to the start of edge vector.
x_{zmp}	is the x coordinate of the ZMP point.
y_{zmp}	is the y coordinate of the ZMP point.
x_i	is the x coordinate of the start of v_i.
y_i	is the y coordinate of the start of v_i.
x_{i+1}	is the x coordinate of the end of v_i.
y_{i+1}	is the y coordinate of the end of v_i.

The smallest distance $d_{i,stabMarg}$ is called the stability margin, positive if the ZMP is stable otherwise negative as described in Equation (6.3). The method has not been presented in any known papers, so in order to test the principle, 250 random ZMP points has been tested to see whether they are stable or not. Figure 6.2 on the next page shows that the principle is 100% able to calculate whether the ZMP points are stable or not. The size of the stability margin measure in which extent the ZMP point is stable, i.e. a large stability margin is the same as high stability.

$$\begin{array}{llll} \text{ZMP point} = \text{stable} & for & 0 \leq min(d_{i,stabMarg}) & i = 1, 2, ..., k \\ \text{ZMP point} = \text{unstable} & for & 0 > min(d_{i,stabMarg}) & i = 1, 2, ..., k \end{array} \tag{6.3}$$

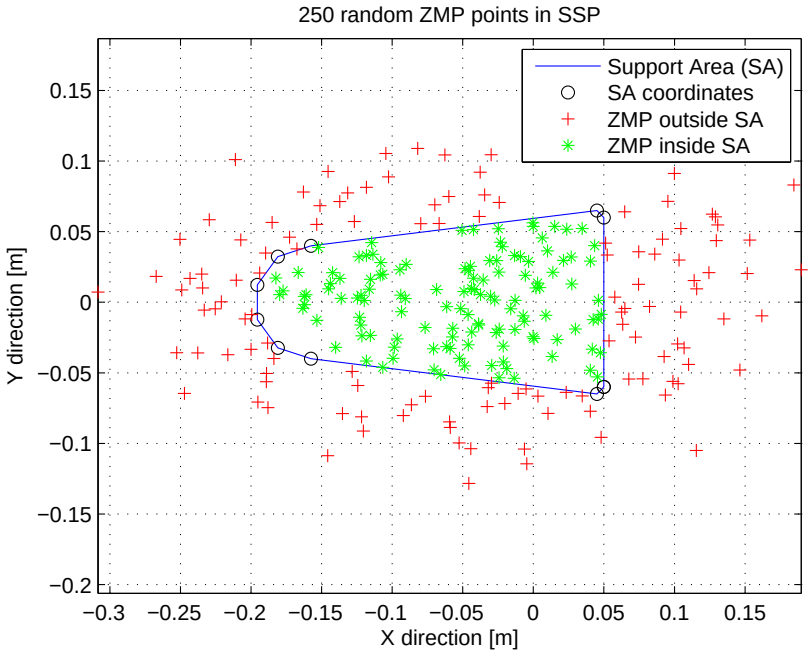


Figure 6.2: Test of the stability margin principle with 250 random ZMP points.

To determine whether the trajectories are stable or not, all ZMPs during one trajectory is calculated and tested whether they are inside or outside the support area. Each time the ZMP point is inside the support area a variable called stability index $stabIndex$ is increased by one. When all the different trajectories have been simulated, the trajectories with maximum stability index are the most stable. When the extent of stability is increased, a lower boundary for the stability margin is used. If the stability margin is larger than the lower boundary, the stability index will be increased with 1 otherwise it will not be increased. This technique is universal, and it can be used for dynamic gait and static gait.

$$stabIndex_{max} = max(stabIndex(i)) \qquad i = 1, 2, 3...n \tag{6.4}$$

where:

n	is the number of simulations.
$stabIndex_{max}$	is the maximal number of stable ZMP's is all simulations.
$stabIndex$	is the array of stability indexes for all simulations.

Another thing that has to be taken in to consideration is the energy consumption during walk. High stability often requires more power to maintain the stability. In [Arakawa and Fukuda, 1996] a method is presented to obtain energy optimized gait trajectory. The idea behind this method is that a walk with high energy efficiency will be similarly to human walk. In [Arakawa and Fukuda, 1996] the method was used with success, but they did not verified the results against a human. In [Ogino1 et al., 2001] another method presented where the trajectory was optimized such that it will minimize the energy consumption. The result of the energy optimized trajectory was similarly to

human gait. The result of all simulations with **AAU-BOT1** might result in severally stable trajectories which have $stabIndex_{max}$ of stable ZMP's. If this is the case, the one with the lowest energy consumption will be used. This will also result in a smoother trajectory. However since a complete dynamical model is not found, it has not possible to obtain an accurate ZMP and an accurate torque. Equation (6.5) describes the energy consumption during a trajectory.

$$P = \frac{1}{T_{sim}} \int_0^{T_{sim}} \tau \dot{\theta} dt \tag{6.5}$$

where:

P	is the energy consumption during a simulation.
T_{sim}	is the simulation time.
τ	is the used torque.
$\dot{\theta}$	is each joint rotational speed.

Using the trajectory with the lowest energy consumption and a satisfying stability margin, would result in the most optimal trajectory for **AAU-BOT1**. In order to obtain a static gait trajectory, the trajectory with the highest stability index and highest stability margin is used, since the ZMP is the GCoM when using static gait.

6.3.2 Cubic Spline Interpolation

In order to calculate a smooth trajectory, Cubic Spline Interpolation is used. This minimize the accelerations, which result in a smooth ZMP trajectory. [Wolfram MathWorld, 2004] describe a cubic spline function as the following:

A cubic spline is a spline constructed of piecewise third-order polynomials which pass through a set of m control points. The second derivative of each polynomial is commonly set to zero at the endpoints, since this provides a boundary condition that completes the system of m-2 equations. This produces a so-called "natural" cubic spline and leads to a simple tridiagonal system which can be solved easily to give the coefficients of the polynomials. However, this choice is not the only one possible and other boundary conditions can be used instead.

The trajectories for the static gait and the dynamic gait are designed via the MATLAB™ function `cspline`.

6.3.3 Arm Trajectory

Common for static and dynamic trajectories are that the arms do not affect the stability, because of the low weight. However a trajectory is created in order to make the walk look natural. Different variations of the arm trajectory have been simulated. The arm trajectory for the right arm is the same as for the left arm. The Trajectory has been created from the constraints in Equation (6.6) and in Equation (6.7).

$$\begin{array}{rll} -30 & \leq \theta_{arm}(t) \leq -60 & , t = t1 \\ 30 & \leq \theta_{arm}(t) \leq 60 & , t = t4 \end{array} \tag{6.6}$$

$$\begin{array}{ll} \dot{\theta}_{arm}(t1) & = 0 \\ \dot{\theta}_{arm}(t4) & = 0 \end{array} \tag{6.7}$$

6.3.4 Foot Trajectory

The foot trajectories are designed via the method used in [Huang et al., 2001]. The biped robot is symmetric and it will only be necessary to design a trajectory for one foot, and copy it to the other foot with a delay of T_{step}. In the following section a foot trajectory for static gait will be designed and dynamic gait trajectory for the foot is described in Appendix F on page 203. Parameters for a step can be found in Table 6.1 on page 103.

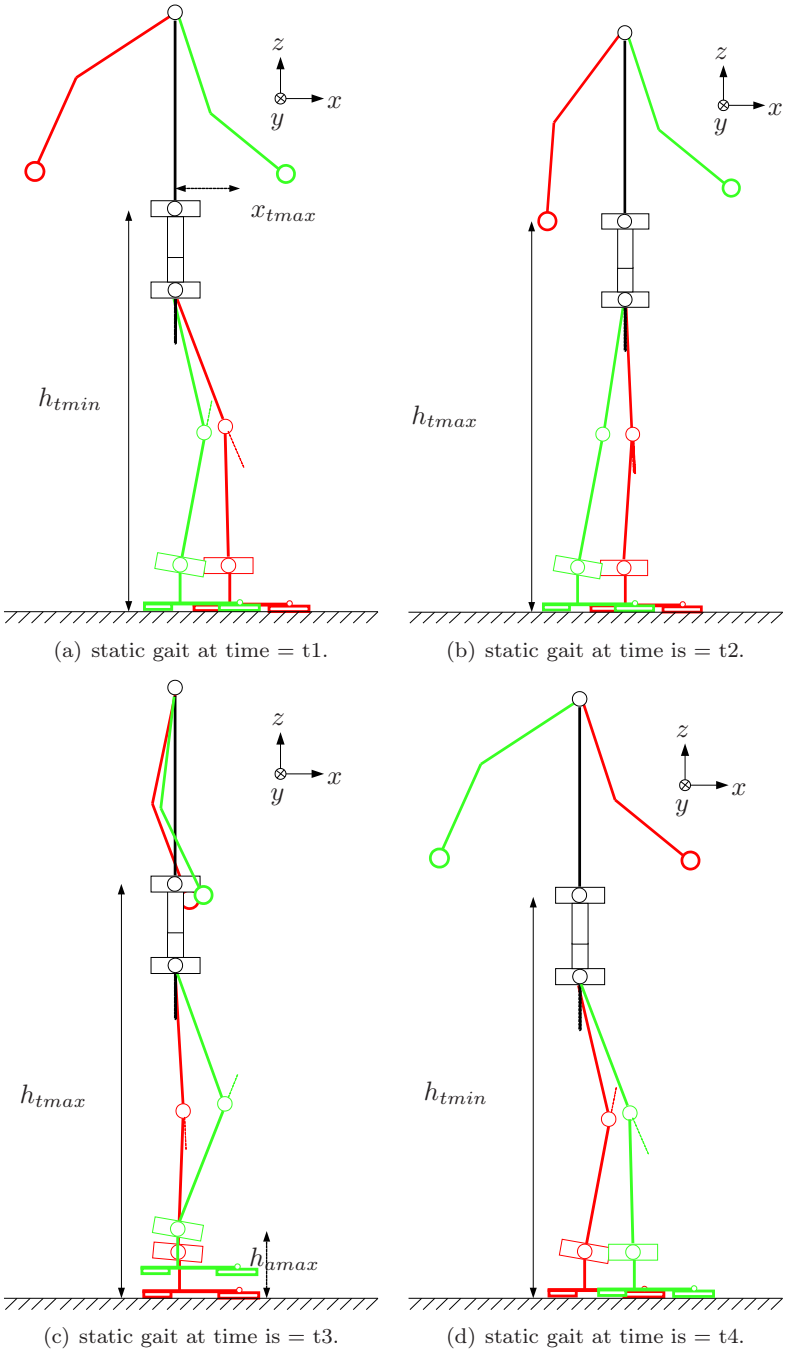
Foot Trajectory for static gait

The trajectory has been designed for the right foot. In order to create the trajectory for the foot, a time line is defined as shown in Equation (6.8). The time line is only for T_{step}, during the second part of T_{cycle} the right foot is on the ground. The distance between the feet is set to 0.28, which is the same as the hip width. Figure 6.3 on the following page shows a sketch of the static gait during T_{step}, it will be described in the following sections. Note that the left side of **AAU-BOT1** is marked with red color and the right side is marked with green color.

$$\begin{aligned} t1 &= (k-1)T_{step} \\ t2 &= kT_{step} - T_{SSP} \\ t3 &= (k-1)T_{step} + T_{max} \\ t4 &= kT_{step} \end{aligned} \tag{6.8}$$

where:

T_{step}	is the time it takes to move right foot in front of the left.
T_{SSP}	is the time **AAU-BOT1** is in SSP during T_{step}.
T_{max}	is the time it takes to raise the ankle to max height h_{amax}.
k	is $1, 2...n$ where n is the number steps.



(a) static gait at time = t1.

(b) static gait at time is = t2.

(c) static gait at time is = t3.

(d) static gait at time is = t4.

Figure 6.3: Sketch of the static gait during T_{step}

Movement of the foot can be divided in to three steps. Firstly the rotation of the foot should be defined, which is zero during static gait. Secondly the forward motion of the foot must be defined, this is done in Equation(6.9).

$$\begin{array}{lll} x_a(t) & = -l_{at} - 0.5 l_{step} & , t = t1 \text{ and } t2 \\ x_a(t) & = -l_{at} - 0.5 l_{step} + l_{max} & , t = t3 \\ x_a(t) & = -l_{at} - l_{step} & , t = t4 \end{array} \tag{6.9}$$

where:
$x_a(t)$ is the horizontal movement of the ankle during T_{step}.
l_{at} is the horizontal distance between the ankle and toe.

Third and last the vertical movement must be defined as in Equation (6.10).

$$\begin{array}{lll} z_a(t) & = l_{an} & , t = t1 \text{ and } t2 \\ z_a(t) & = h_{amax} & , t = t3 \\ z_a(t) & = l_{an} & , t = t4 \end{array} \tag{6.10}$$

where:
$z_a(t)$ is the vertical movement of the ankle during T_{step}.
l_{an} is the height of the ankle from the ground.
h_{amax} is the maximum ankle height during T_{step} .

In order to achieve static gaiting the following constraints must be applied. This will also secure a smooth movement of the foot and not suddenly change in the movement direction. All constraints are used to derive different 3^{th} order spline interpolation function as described in Section 6.3.2 on page 106.

$$\begin{array}{ll} \dot{x}_a(t3) & = 0 \\ \dot{x}_a(t4) & = 0 \\ \dot{z}_a(t3) & = 0 \\ \dot{z}_a(t4) & = 0 \end{array} \tag{6.11}$$

6.3.5 Torso Trajectory Generation

The torso trajectory has also been designed after [Huang et al., 2001]. However the used biped robot in that paper do not have a waist with 3 DoF as **AAU-BOT1**, that is one of the main differences compared to other biped robots. This makes possible to use a ZMP controller acting on joint θ_{16}, θ_{17} and θ_{18}, that does not affect the rest of the system. For further details see Section 7.3.3 on page 136. During trajectory generation the torso rotation is zero. The Yaw rotation in the pelvis is implemented in the trajectory generation. The trajectory is designed for the waist and not the hip as in [Huang et al., 2001]. The static gait trajectory is presented and the dynamical trajectory is described in Appendix F on page 203.

Torso Trajectory for static gait

The parameters in the following equations can be seen in Table 6.1 on page 103. The constraints for the vertical movement of the torso can be found in Equation (6.12) and

in Equation (6.13).

$$\begin{array}{lll} z_t(t) & = h_{tmin} & ,t = t1 \\ z_t(t) & = h_{tmax} & ,t = t2 \\ z_t(t) & = h_{tmax} & ,t = t3 \\ z_t(t) & = h_{amin} & ,t = t4 \end{array} \tag{6.12}$$

where:

$z_t(t)$	is the vertical movement of the torso during T_{step}.
h_{tmax}	is the maximum height of the torso.
h_{tmin}	is the minimum height of the torso.

$$\begin{array}{ll} \dot{z}_t(t2) & = 0 \\ \dot{z}_t(t4) & = 0 \end{array} \tag{6.13}$$

According to [Vaughan et al., 1992] normal yaw rotation in the pelvis is around $\pm 18°$. However this is on a normal human and the rotation affect the stability, which is why several different values have been tested. The constraints for the pelvis rotation can be seen in Equation (6.14) and Equation (6.15).

$$\begin{array}{rll} -4.5° & \leq \theta_{16}(t) \leq -18° & ,t = t1 \text{ and } t2 \\ 4.5° & \leq \theta_{16}(t) \leq 18° & ,t = t4 \end{array} \tag{6.14}$$

$$\begin{array}{ll} \dot{\theta}_{16}(t2) & = 0 \\ \dot{\theta}_{16}(t4) & = 0 \end{array} \tag{6.15}$$

The trajectory for the movement of the torso in the transverse plane is quite simple, since it is only moved during DSP, i.e. the CoM is moved from right leg to left leg. Figure 6.3(a) on page 108 and Figure 6.4 on the facing page shows sketches of how the torso is placed in the transverse plane.

The constraints for the torso movement in the sagital plane which is in the x-directions is listed in Equation (6.16) and in Equation (6.17).

$$\begin{array}{lll} x_t(t) & = -l_{at} - 0.5 l_{step} + x_{tmax} & ,t = t1 \\ x_t(t) & = -l_{at} x_{tmax} & ,t = t2 \text{ and } t3 \text{ and } t4 \end{array} \tag{6.16}$$

where:

$x_t(t)$	is the movement of the torso in the x-direction during T_{step}.
l_{at}	is the transverse distance between the ankle and toe.
x_{tmax}	is the length in the x-axis from the torso to the right foot.

$$\begin{array}{ll} \dot{x}_t(t1) & = 0 \\ \dot{x}_t(t2) & = 0 \end{array} \tag{6.17}$$

In order to obtain a good stability, x_{tmax} has been tested in a interval given in Equation (6.18).

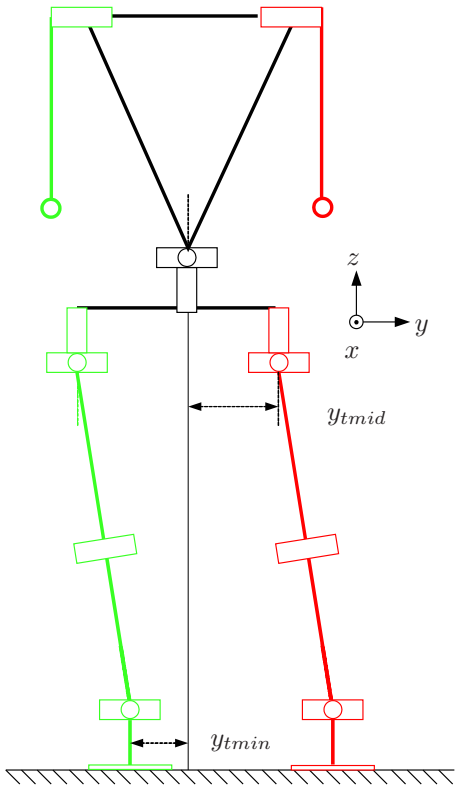


Figure 6.4: Sketch of the **AAU-BOT1** seen from the front.

$$0.4 l_{step} \leq x_{tmax} \leq 2.3 \tag{6.18}$$

The constraints for the torso movement in the y-directions is listed in Equation (6.19) and in Equation (6.20).

$$\begin{array}{lll} y_t(t) & = -y_{tmid} - y_{tmin} & , t = t1 \\ y_t(t) & = -y_{tmid} + y_{tmin} & , t = t2 \text{ and } t3 \text{ and } t4 \end{array} \tag{6.19}$$

where:

$y_t(t)$	is the movement of the torso in the y-direction during T_{step}.
y_{mid}	is the distance from center of the pelvis to one hip.
y_{tmin}	is the length in the y-axis from the torso to the right foot.

$$\begin{array}{ll} \dot{y}_t(t1) & = 0 \\ \dot{y}_t(t2) & = 0 \end{array} \tag{6.20}$$

In order to obtain a good stability, y_{tmin} has been tested in a interval given in Equation (6.21).

$$0.9 y_{mid} \leq y_{tmin} \leq 1.5 y_{mid} \tag{6.21}$$

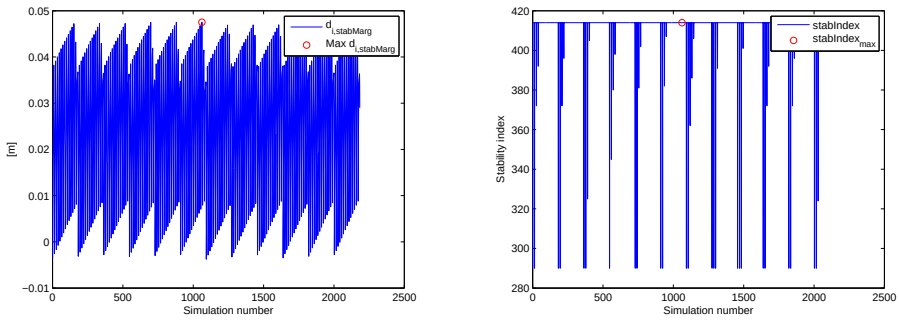
6.3.6 Start and Stop Trajectory

To ensure that the **AAU-BOT1** is not unstable during the start of a walk trajectory and in the end of a trajectory, a special trajectory has to be generated for the static gait and the dynamic gait. The start trajectory moves the robot from initial position, to a path that leads into the chosen trajectory. The trajectory starts very slow, such that it has high stability. When the start trajectory is generated, the stop trajectory is just the inverse of it. The design and implementation of it will not be mentioned further to avoid replications.

6.4 Simulation and Results of Trajectory Generation

This section contains the simulation results of trajectory generation only for the static gait simulations.

In order to find the best trajectory for static gait 2181 different trajectories has been simulated. Each simulation used the Inverse kinematic Model and the kinematic Model to evaluate the GCoM. As mentioned in Section 6.3.1 on page 102 the toe feature has been disabled to increase the Support Area. At each simulation the stability margin and stability index for the entire trajectory is measured. The trajectory for **AAU-BOT1** is simulated at a walking speed of $0.002\frac{m}{s}$, however to reduce the simulation time, the walking speed will be $0.06\frac{m}{s}$ and it will still be static gait. Figure 6.5(a) shows the result of the stability margin. As is can be seen most of the trajectory simulations are stable through the entire simulation. To find the best trajectory out of all simulations, the trajectory with the highest stability margin and the highest is chosen. The best trajectory has stability index 414 and a stability margin at 0.0475m.



(a) Stability margins during all simulations. (b) Stability indexs during all simulations.

Figure 6.5: Stability during all simulations.

The parameters that is used to design the best trajectory can be seen in Table 6.3 on the next page

Table 6.3: Parameters from the best trajectory

Parameters/results	**Values**
Torso x-direction displacement, x_{tmax}	0.13m
Torso y-direction displacement, y_{tmax}	0.182m
Torso max/min angles, θ_{16}	$\pm 12.86°$
Arm max/min angle, θ_{arm}	$\pm 60°$
Simulation number	1062
Max stability margin, $d_{1062,stabMarg}$	0.0475m
Max stability index, $stabIndex_{max}$	414

Each simulation is only simulated for T_{cycle}. The different phases during one the simulation can be seen at Figure 6.6. Note that the used simulation walking speed is $0.002\frac{m}{s}$. Only simulation for 1 step cycle is in the following graphs.

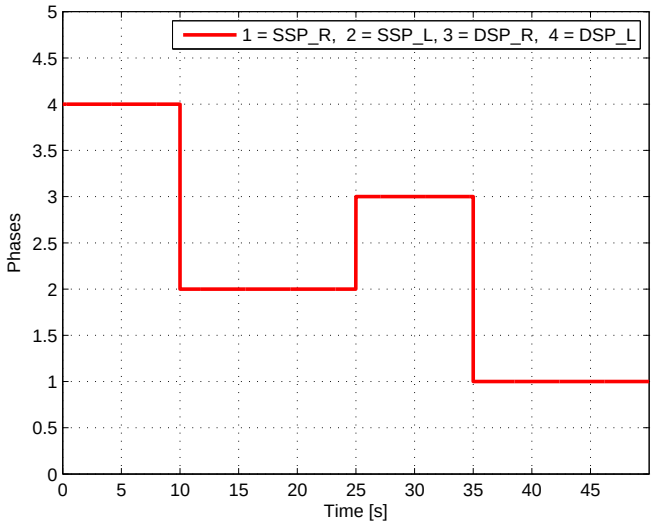
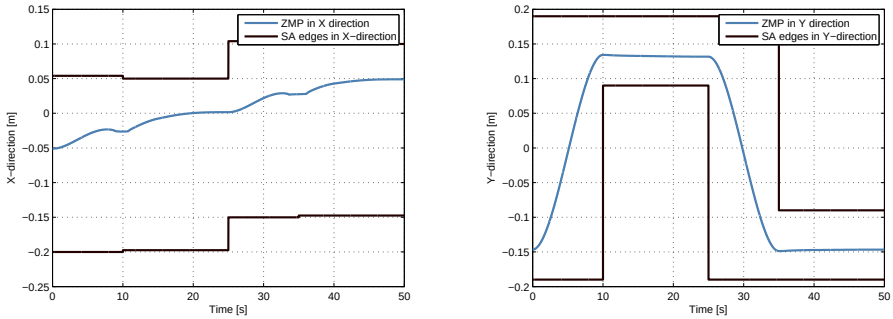


Figure 6.6: The different phases during simulation of the best trajectory.

Figure 6.7(a) on the next page is the resulting ZMP trajectory in the x-direction and Figure 6.7(b) on the following page is the resulting ZMP trajectory in the y-direction. The minimum margin for the ZMP point to the SA edge is 0.0475m. As expected the ZMP is moved from one foot to the other during DSP.



(a) X values of ZMP during T_{cycle} (b) Y values of ZMP during T_{cycle}

Figure 6.7: ZMP during T_{cycle}

Figure 6.8 is a graph of the movement in the x-direction for the ankles and the torso during the best trajectory. Figure 6.9 on the facing page is the movement of the torso and the ankles in the y-direction. As it can be seen at the figure, the torso joint, θ_{16}, has to be move further than the ankles in order to obtain the best trajectory. This can be a problem if the physical constraints of **AAU-BOT1**limit this action shown in Figure 6.8, this information is however not obtained due to time constraints. The green graphs are the right ankle and the red graphs are the left ankle.

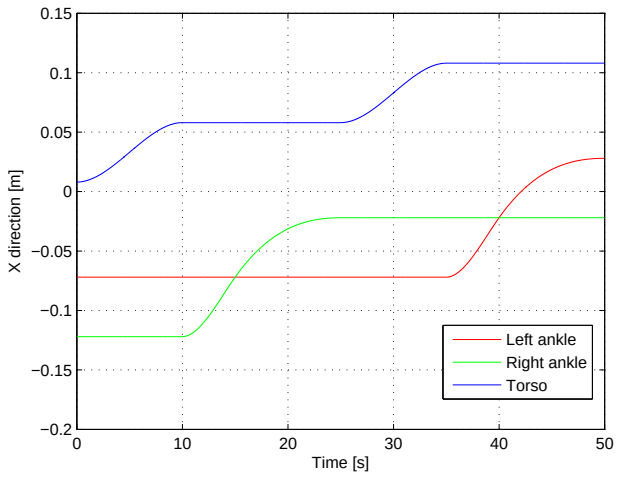


Figure 6.8: Ankles and Torso movement in the x-direction during T_{cycle}.

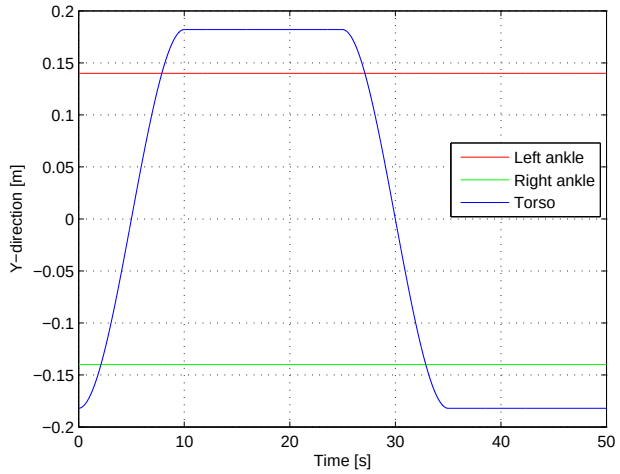


Figure 6.9: Ankles and Torso movement in the y-direction during T_{cycle}.

Figure 6.10 is a graph of the movement of the torso and ankles in the z-direction during the simulation of the best trajectory. As is can be seen the torso moves up during DSP and down in SSP. This is very similarly to what humans do during static gait. The ankle also moves as expected, which means that the right ankle is move first and the left ankle is moved secondly.

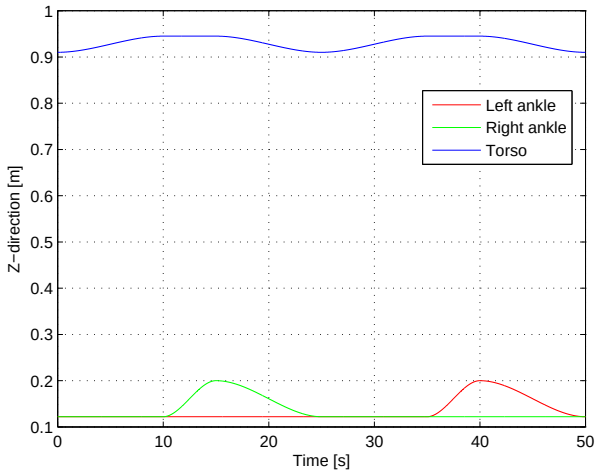


Figure 6.10: Ankles and Torso movement in the z-direction during T_{cycle}.

The resulting maximum ankle joint angles can be seen in Table 6.4 on the next page. It is specially the pitch angle, θ_3 and θ_{13}, that are high. The maximum pitch angles occurs when the ankles are at the maximum height at time 14.5 and 40 sec as shown on Figure 6.10. This is because the inverse kinematic always will try to set the feet level. However in order to create a trajectory that is the most stable for **AAU-BOT1**, further

information about the weight distribution and maximum joint angles constraints on the physical **AAU-BOT1** must be obtained.

Table 6.4: Max ankle roll and pitch angles during simulations

Joint #	Max pos. rotation [degree]	Max neg. rotation [degree]
2	16.52°	-17.97°
3	37.64°	5.18°
12	-5.19°	-38.47°
13	16.52°	-18.11°

Figure 6.11 shows a graph of the rotation of the arms and the torso. During all simulations it was verified that the arms do not affect the stability much. As is can been seen there is a maximum rotation on ±12.86°. Which means a small rotation in the torso does increase the stability.

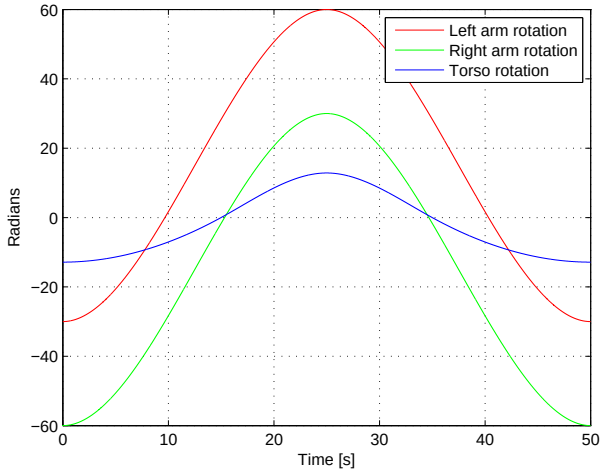


Figure 6.11: Arms and Torso rotations during T_{cycle}.

6.5 Summary of Trajectory Generation

This chapter describes how to create a trajectory for **AAU-BOT1** that has the highest stability. It also describes different approaches that can be used to derive dynamic gait in simulation and static gait trajectories.

Deriving a dynamic model in such extent that it is accurate enough to obtain a correct ZMP, has proved to be hard. This means that only a method to design dynamic gait trajectories has been derived.

It has been possible to obtain from the test, the best static gait trajectory. The trajectory has been tested in simulation and at **AAU-BOT1**. The test of the trajectory on **AAU-BOT1** can be found in Section 8 on page 143. The best found trajectory, i.e. the most stable trajectory, had a high stability margin, which means that the ZMP during the entire simulation is not closer than 0.0475 m to the edge of the Support

Area. Unfortunately it was not possible to measure the energy consumption during the simulation, due to the incomplete dynamic model, so it cannot be guaranteed that the found trajectory is the most energy efficient. The used method to detect the Stability, i.e. by using the stability margin and the stability index is efficient. The method can be used to detect whether the trajectory is stable. However a problem occuredduring the simulation of the best trajectory, since **AAU-BOT1** is designed to be able to walk dynamic gait and not static gait. I.e. it has limited joint angles and that the best trajectory exceeds these limits. However accurate knowledge about the angle limits and the weight distribution on the physical **AAU-BOT1**has to be obtained before these can be used for simulation and a evaluation can be done.

Chapter 7

Control

This chapter describes the development of two different controller strategies which both have the same main objective. The main objective can be split up into two parts, the first part is to control the joint angles such that the trajectories are tracked, this is called a posture controller. The next part is the balance controller that tracks the GCoM trajectories at all time and ensures that the robot is operating within the support area, this is called a ZMP controller. The controller strategies are tested and verified in three stages. The first stages is to test the controllers toward the developed model in Simulink. Hereafter the controller strategy is verified on the virtual robot in Webots. The last verification is done on the actual ***AAU-BOT1****. The controllers are verified to the extent it is possible, as one controller may be dependent of another controller in the control strategy.*

7.1 Controller Structure

By now an overall model for the robot has been derived, and furthermore the inverse kinematics and the trajectories for the robot has been developed. The trajectories is based on the model and throughout the derivation the stability has been in focus. This was done by making sure that GCoM is inside the support area. The inverse kinematic and the trajectories has been used together to derive the preplanned motion patterns for each joint. This will be called the "pre-generated posture trajectory". Along with the posture trajectory a trajectory for the GCoM is derived. It is crucial that the robot keeps its GCoM close to this pre-generated GCoM trajectory or it can cause the robot to tip over the edge of a foot.

Many different control strategies are pursued when browsing through the biped robot literature. The strategy on inexpensive commercial toy robots are often the same, they usually run from a macro and can only do preprogrammed movements, usually little control is combined with these robots.

A more interesting approach can be seen in [Kondak and Hommel, 2003b], where a 5 DoF biped robots is simulated. The angular acceleration in this approach is used as a control reference and by converting this input with an inverse dynamical model the torque in each joint can be calculated. Two controllers are implemented. The first controller is a nonlinear feedback controller that linearize and decouple the nonlinear system where it uses the dynamic model to calculate the joint torques, given the angular acceleration. The other controller is three PD controllers, they are used to control the position of the motors. The input are the positions from the kinematic transformation and the output

are the angular acceleration which is the input to the nonlinear controller. The output from the system is the angular velocity and position, these are used by the PD controllers as feedback.

Another control strategy is seen in [Bachar, 2004], where it is described how ZMP controller is used to retain balance even if exposed to external disturbance. The idea is to utilize the inverse Jacobian to describe the ZMP position based on the generalized coordinates. Thereby it is possible to move ZMP in any desired direction. If this idea is used to ensure that ZMP is within the Support Area then stability is ensured.

7.1.1 Controller Strategies

As stated in the problem formulation 1.4 on page 17 this master's thesis concentrates on obtaining static gait. However since **AAU-BOT1** is a long term project scheduled to finish in year 2010, the possibilities for dynamic gait is also exploited to some extent in this master's thesis as well.

With the above considerations taken into account, it has been chosen to develop two posture controller strategies.

Control Strategy A utilizes the knowledge obtained in the dynamic model, to generate a model based controller, see Figure 7.1. This strategy depends heavily on a correct dynamic model, and is harder to verify due to its high complexity. The reference scaling N_x and N_u ensures a steady state gain of 1 from the reference position to the steady state position. A Kalman filter is implemented to observe all observable states in the system and minimizes noise. The states are then utilized in a feedback control loop together with a posture controller that ensures the system is in the correct position. The reference to the posture controller is the pre-generated posture trajectory from Section 6. The balance is monitored and controlled by the ZMP controller. This controller enables **AAU-BOT1** to stay balanced by using the waist joints 16 and 17 during gait. Control Strategy A is most likely not to be implemented on actual system as a number of key parameters such as the masses, inertia, friction and CoM are not known. Before this can be done the hardware implementation on the **AAU-BOT1** has to be done, the new hardware setup has to be implemented in SolidWorks, which can derive most of these parameters. This means that this control strategy only will be used to control the simulation.

Control Strategy B utilize the build in position control loops in the EPOS amplifiers and the Webots simulation, see Figure 7.2. This is a big advantage as this inner position control loop will have higher sample rate on **AAU-BOT1** (1 ms [Maxon Motors, 2007a]) and will utilize noncausal [Cyberbotics Ltd., 2008] features in Webots. The disadvantage is that the dual axis motors are not supported, and those joints still have to be current controlled. Note that the ZMP controller in this strategy uses custom model for obtaining the torso roll and pitch states. Since this is a relatively small model the derivation from start to end can be observed. This is not feasible for the complete model as this model contains 38 states, 17 inputs and 38 outputs. N_x and N_u is reference scaling that ensures a steady state gain of 1 from the reference to the state.

In the following the two controller strategies are derived and discussed.

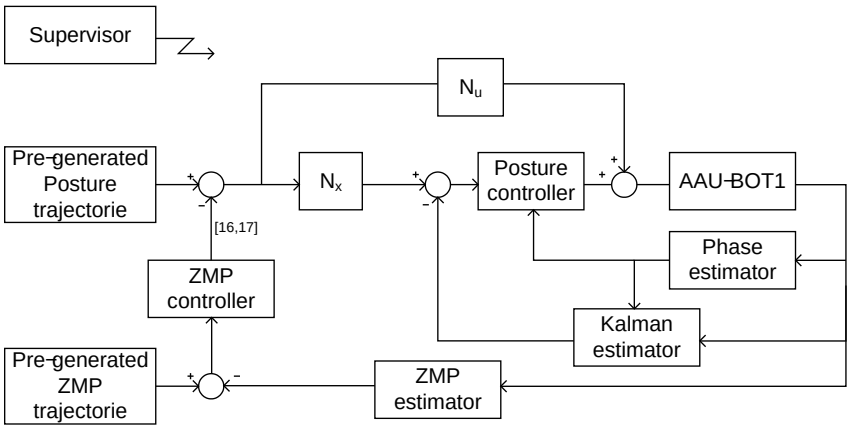


Figure 7.1: Control strategy A.

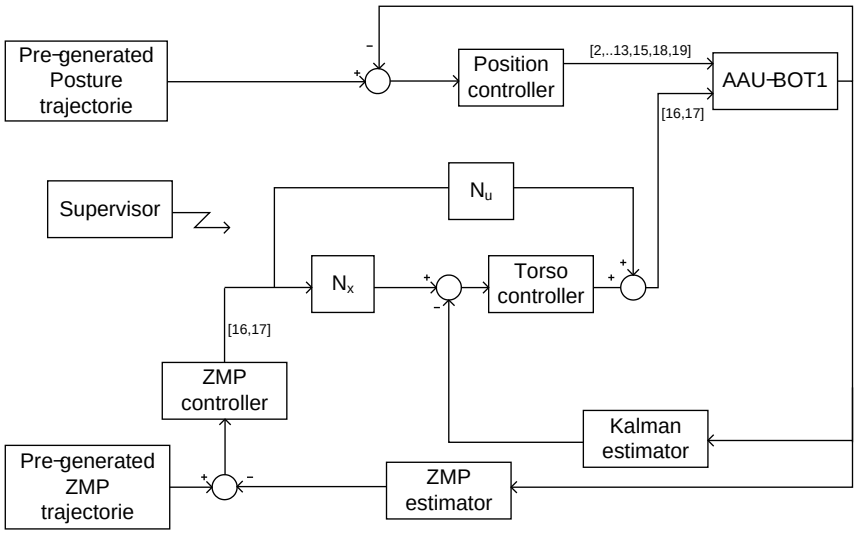


Figure 7.2: Control strategy B.

7.2 Control Strategy A

This section describes the first control strategy, which involves two independent controllers: one ensures that the joints of the robots is in the correct position and the other ensures stable balance.

7.2.1 Posture Control

This controller is necessary in order to follow a trajectory, or just to stand in an upright position. The layout for the posture controller can be seen in Figure 7.3. The

control reference to this control strategy is the angles from the trajectory. The input to the **AAU-BOT1** is torque and the output is the angles of the joints. The feedback control consists of a gain matrix calculated on background of a Linear Quadratic Regulator (LQR) weight performance function and two reference scaling matrices. This feedback controls the model by using the predicted states from the Kalman filter. With the Kalman filter and LQR controller together they form a Linear Quadratic Gaussian (LQG) controller [Dutton et al., 1997]. Before the actual values can be calculated, some requirements have to be satisfied.

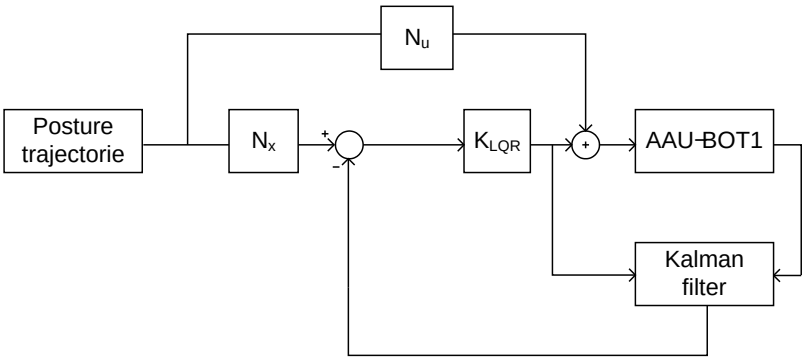


Figure 7.3: The posture controller setup; a LQR controller together with a Kalman filter forms the LQG controller.

Prerequisites

To perform the control strategy some properties have to be complied in order to be sure that the system can be controlled and observed. To test for controllability and observability the state space representation of the robot, derived in Section 5.5.2, is needed. The state space system consist of the A matrix which is (n,n), the B matrix is (n,m), the C is (m,n) and D matrix is (n,m) and can be seen in Equation (5.29). First controllability is checked. One way to determine whether a system is controllable, is to check whether the controllability matrix has full rank. Furthermore the system has to be stabilizable, for a control design to result in something usable. To check whether the system is controllable the sensitivity matrix method can be used as described in [Dutton et al., 1997] and can be seen in Equation (7.1).

$$s(\lambda_i) = [(A - \lambda_i I), B] \qquad i = 1, 2, 3...n \tag{7.1}$$

where:
λ_i is the eigenvalues of the system
A is a 38x38 system matrix
B is a 38x19 input matrix

If the rank of $s(\lambda_i)$ is n, the state is controllable, but if rank(s(λ_i)) $< n$ the state is uncontrollable. In order for a system to be stabilizable all the uncontrollable states must be in the left half plane and all states with poles in right half plane must be controllable.

This means that all the unstable poles can be moved to the left half plane by using control feedback. By applying the method using MATLAB^{TM} yields the following:

$$\text{rank}(s(\lambda_i)) = 38 \qquad i = 1, 2, 3...n \tag{7.2}$$

Equation (7.2) states that all the modes are controllable, hence feedback control can be applied to stabilize the system.

The same procedure as mention above for the controllability can be used to check whether the system is observable. The sensitivity matrix for observability can be seen in Equation (7.3)

$$o(\lambda_i) \quad = \begin{bmatrix} (A - \lambda_i I) \\ C \end{bmatrix} \quad i = 1, 2, 3...n \tag{7.3}$$

where:

λ_i	is the eigenvalues of the system
A	is a 38x38 system matrix
C	is a 38x19 output matrix

If the rank(o(λ_i)) = n the state is observable, but if rank(o(λ_i)) $< n$ the mode is unobservable.

$$\text{rank}(o(\lambda_i)) = 38 \qquad i = 1, 2, 3...n \tag{7.4}$$

With full rank the system can be observed and an observer can be applied to the system.

Linear Quadratic Regulator

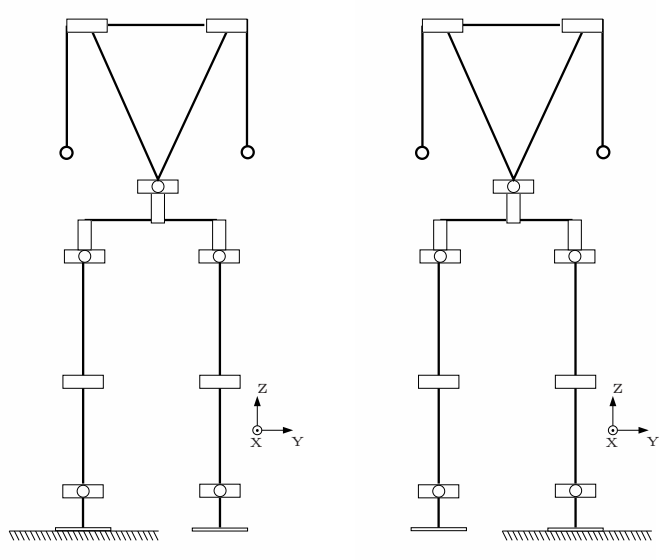
The LQR can be used to design an optimal feedback controller to a linear model. The controller is based on regular state space feedback topology. The input to the system is the error between the reference multiplied by the gain matrix generated by the LQR method. The system used to calculate the LQR gains is the linearized state space representation of the **AAU-BOT1**. The model has been linearized in the working point where the **AAU-BOT1** is standing in an upright position, with all angles equal to zero as seen in Figure 7.4.

As it can be seen in the figure the model is derived for two phases. SSP-R can be seen in Figure 7.4(a) and the SSP-L can be seen in Figure 7.4(b). It looks like the model is in DSP when linearized, but this is not the case. DSP can be obtained by combining the the SSP models as described in Section 5.5 on page 83. The controllers for the two models are done analogously, and by describing the development of the first controller for the SSP-R model will automatically generated parameters used for the second controller. Here the development of the controller for the model in SSP-R will be described.

In Figure 7.5 the pole-zero map of the linearized model in SSP-R can be seen. The Figure shows multiple poles close to zero and the imaginer axis, and a number of poles in the right half plane. The feedback controller must move these poles to the left half plane to stabilize the system.

Since the discrete model is needed a Zero Order Hold (ZOH) is used with a sample time Ts=0.004. The discrete system is on the form as shown in Equation (7.5).

$$X(k+1) = \Phi X(k) + \Gamma U(k) \tag{7.5}$$



(a) Single support phase right. (b) Single support phase left.

Figure 7.4: The working point for the linearized model in both phases.

The feedback control law is defined as in Equation (7.6)

$$U(k) = -L(k)X(k) \tag{7.6}$$

The performance function used in the LQR design is defined in equation (7.7), assuming H is quadratic in $u(k)$ and $x(k)$ [Sørensen, 2007].

$$\begin{aligned} I_i^N &= \sum_{k=i}^{N} H(X(k), U(k)) \\ &= \sum_{k=i}^{N} (X'(k)\boldsymbol{Q}X(k) + U'(k)\boldsymbol{R}U(k)) \end{aligned} \tag{7.7}$$

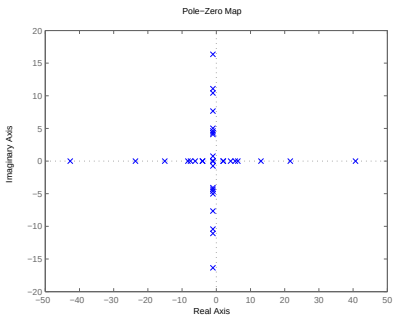


Figure 7.5: Pole-zero map of linear system.

where:
$\boldsymbol{Q}$ is the weight matrix on the state vector (n, n).
$\boldsymbol{R}$ is the weight matrix on the output vector (m, m).

The design process derives control signals that will minimize the performance function from Equation (7.7). The optimal controller is calculated given the set of weights ($\boldsymbol{Q}$ and $\boldsymbol{R}$). This is done by recursively calculating the values of $L(k)$.

In order to choose these proper weights a certain knowledge of the system is needed. The first noticeable thing about this system is the great amount of cross coupling. A change on one state can cause multiple other states to move as well. States that rotate around one axis are most likely to influence other states rotating around the same axis.

The weights should be weighted more on those states that contain the largest process noise and are more influenced by other states. Since this is the model for SSP-R the first joints in the kinematic are more influential and therefore they have to be weighted higher than the joints in the end of this kinematic chain. Two different types of weights have to be chosen. $\boldsymbol{Q}$ in Equation 7.8 contains both weights for the position and velocity. R from Equation (7.9) contains the control signal weights. Generally the position and velocity can be weighted higher if a more accurate system is wanted. If a fast system is needed, the control signal can be weighted higher. For this LQR controller, the weights on the states are higher since an accurate controller is wanted, further more the weights on the states in the beginning of the kinematic chain is weighted higher. The chosen weights for the linear model can be seen in the following two Equations:

$$\begin{aligned}\boldsymbol{Q} =& \text{diag}([330000\ 180000\ 600000\ 18000\ 980000\ 800000\ 400800\ 900000\ 900000 \\ & 1080000\ 880000\ 80000\ 500000\ 1800\ 800000\ 950000\ 180\ 100000 \\ & 100000\ 60000\ 10\ 90050\ 40000\ 500000\ 250000\ 255000\ 20000\ 300000 \\ & 400000\ 5085\ 9040\ 50000\ 600\ 8000\ 100000\ 20005\ 16080\ 16080]) \end{aligned} \tag{7.8}$$

$$\begin{aligned}\boldsymbol{R} =& \text{diag}([35\ 25\ 10\ 30\ 25\ 20\ 20\ 500\ 15\ 2.5 \\ & 80\ 3000\ 3000\ 10\ 100\ 20\ 35\ 325\ 325]) \end{aligned} \tag{7.9}$$

With these weights the LQR method can derive a feedback gain matrix such that the linear model can perform steps on one state and suppress cross coupling to a minimum.

Reference Scaling To use the output from the inverse kinematics, a scaling of the input to the model is needed. This scaling is derived such that the DC-gain from the position of the trajectory to the torque for the model can be considered a one. These scaling parameters are calculated with the MATLAB^{TM} command `refi.m` and takes the model and a H_r matrix that defines which states that can be controlled as input. The command generates two gain matrices N_x and N_u. N_x is the scaling to the states and N_u is the scaling of the input.

Before the developed LQR controller together with the reference scaling can be tested the Kalman predictor must be derived.

Kalman Filter

In order to reduce the noise on the angular velocities from the **AAU-BOT1**, a Kalman filter is implemented, see Figure 7.6. A high precision measurement of the angular velocities are important due to the fact that the LQR controller uses them in the feedback. The Kalman filter design is based on [Grewal and Andrews, 2001]. A Kalman filter

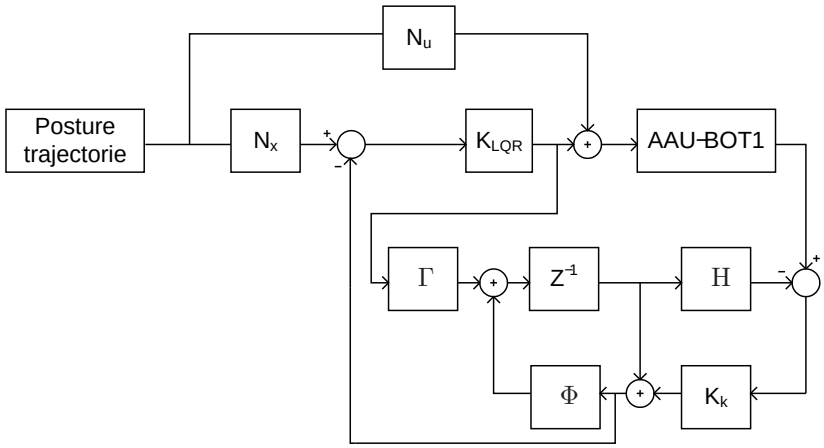


Figure 7.6: The LQG controller.

consists of two steps:

- Predict step:

$$\vec{\hat{x}}_k^- = \boldsymbol{\Phi}\vec{\hat{x}}_{k-1} + \boldsymbol{\Gamma}\vec{u}_{k-1} \tag{7.10}$$

$$\boldsymbol{P}_k^- = \boldsymbol{\Phi}\boldsymbol{P}_{k-1}\boldsymbol{\Phi}^T + \boldsymbol{Q} \tag{7.11}$$

- Update step:

$$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}^{\boldsymbol{T}}(\boldsymbol{H}\boldsymbol{P}_k^-\boldsymbol{H}^{\boldsymbol{T}} + \boldsymbol{R})^{-1} \tag{7.12}$$

$$\vec{\hat{x}}_k = \vec{\hat{x}}_k^- + \boldsymbol{K}_k(z_k - \boldsymbol{H}\vec{\hat{x}}_k^-) \tag{7.13}$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^- \tag{7.14}$$

where:

$\vec{\hat{x}}_k$	is the estimated value of the state at the time k
$\boldsymbol{\Phi}$	is the discrete value of the A matrix
$\boldsymbol{\Gamma}$	is the discrete value of the B matrix
$\boldsymbol{H}$	is the discrete value of the C matrix
$\vec{u}_k$	is the input at time k
$\boldsymbol{P}_k$	is the error covariance matrix at time k
$\boldsymbol{Q}$	is the noise on the states
$\boldsymbol{K}_k$	is the Kalman filter feedback at time k
$\boldsymbol{R}$	is the noise on the measurements
z_k	is the output of **AAU-BOT1** at time k

A caret ($\hat{}$) denotes an estimated value and a minus ($^-$) denotes an internal variable in the Kalman filter. The $\boldsymbol{Q}$ matrix is set to $10^{-5}\cdot I$ and the $\boldsymbol{R}$ matrix is set to $10^{-5}\cdot I$. The Kalman filter is implemented and verified with the rest of the controller in Appendix B.1.

Verification

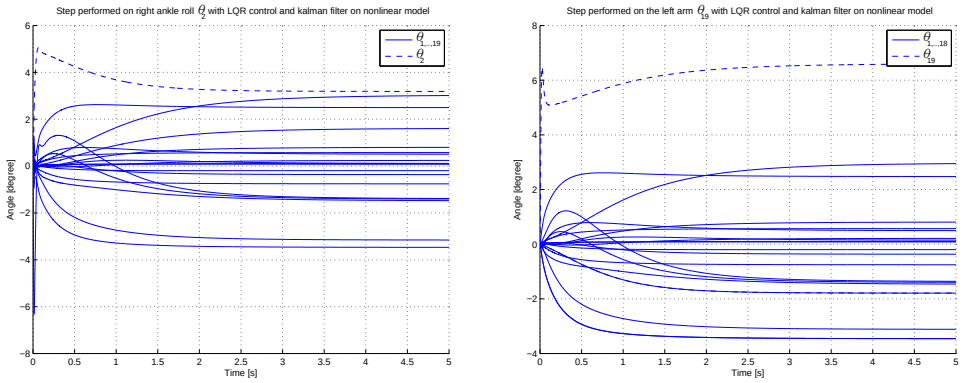
The verification is done in a number of steps and is described thoroughly in Appendix B.1. Here the linearized model was verified with the LQG controller without problems.

Hereafter the nonlinear model was verified with the LQG controller, this did not work initially and the Kalman filter was disabled in order to test the LQR controller. With this attempt the nonlinear model could be stabilized. The Kalman filter was enabled again and the weights on the position states was increased. The weights increased most are those states which are more affected by other states. In Equation (7.15) and (7.16) the new weights are found.

$$\begin{aligned}\boldsymbol{Q} =& 10^{10}\text{diag}(\cdot[830000000000\ 180\ 10000\ 580\ 9200000000\ 100\ 20\ 10\ 70 \\ & 90.8\ 50000\ 100\ 1000\ 1.8\ 100\ 15\ 100\ 9\ 9\ 0.00006 \\ & 0.000000001\ 0.00009\ 0.000004\ 0.0005\ 0.00000025\ 0.00000755 \\ & 0.0002\ 0.00003\ 0.000004\ 0.00000205\ 0.0000091 \\ & 0.00001\ 0.00006\ 0.00008\ 0.0005\ 0.0000002\ 0.00000002\ 0.00000002]) \end{aligned} \tag{7.15}$$

$$\begin{aligned}\boldsymbol{R} =& \quad \text{diag}([35\ 25\ 10\ 30\ 25\ 20\ 20\ 500\ 15\ 2.5 \\ & \quad 80\ 3000\ 3000\ 10\ 100\ 20\ 35\ 325\ 325]) \end{aligned} \tag{7.16}$$

The weights on the position is roughly increased by a factor 10^{10} in order to stabilize the nonlinear model with the LQG controller. These are a very large weight values, and this can only be done because the model assume that the supporting foot is fixed to the ground. These high values can not be used on the actual system nor on the virtual robot, as the supporting foot of the **AAU-BOT1** is not fixed to the ground. In Figure 7.7(a) and (b) two steps are performed on the nonlinear model.



(a) Step on θ_2 (b) Step on θ_{19}

Figure 7.7: Steps performed on the nonlinear model with LQR controller and Kalman filter.

The first step in Figure 7.7(a) shows a step of 5 degrees on θ_2 ankle roll. This shows a significant steady state error, both on the states that should be zero and on the stepped state. The same is valid for the step on θ_{19} the arm, where a steady state error is present on most of the states.

Even though the LQG controller is able to stabilize the nonlinear model, it was not possible to stabilize the Webots representation of **AAU-BOT1**. As stated before the weights used for the nonlinear model was not suitable for the virtual robot nor the actual

system, since they both contain an extra joint. Initially the Webots representation of **AAU-BOT1** was tested by increasing the size and weight of the supporting foot. This was done to imitate the model in SSP as truly as possible. Problems occurred when doing this. The supporting foot can only be so heavy since the solid foot will penetrate through the floor and fall into the virtual eternity. The large supporting solid foot is set to 100 kg this is larger than the rest of the robot 68.5 kg. This setup ensures that the virtual robot does not fall through the floor. Even a large heavy weight foot was not enough to disable the extra joint between the virtual robot and the floor.

Instead of testing the SSP-R model on the virtual **AAU-BOT1** in Webots, the DSP model is implemented and tested instead. This is done as described in Section 5.5. The models both for SSP-R and SSP-L is used to derive two LQG controllers, the output from the two controllers are weighted by a factor that is calculated on basis of how much the **AAU-BOT1** is in one or another phase. For test purpose only the weight factor is chosen to 0.5 this means that both phases are weighted equal when in DSP.

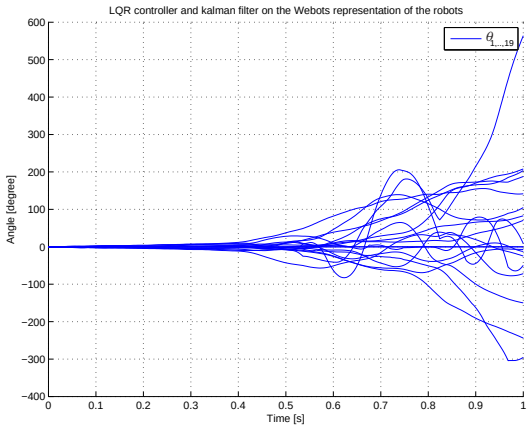


Figure 7.8: Zero step on the Webots representation of the robot.

In Figure 7.8 it can be seen that the controllers try to stabilize the virtual **AAU-BOT1**, and at roughly 0.5 s the states are to far from the working point for the Kalman estimator to give a proper state estimate, and the system becomes unstable.

7.2.2 ZMP Controller

The ZMP controller is proposed such that **AAU-BOT1** can remain balanced during walk. The input to the ZMP controller is the error between the pre-generated ZMP trajectory and the estimated ZMP. As mentioned in the beginning of the chapter the ZMP controller can unfortunately not be derived and verified without the posture controller, and since the posture controller failed to stabilize the Webots model, the ZMP controller is not developed further in Control Strategy A.

7.3 Control Strategy B

This second control strategy is proposed to utilize the build in position controller in the EPOS amplifiers. Also the build in feature of a position controller in Webots is utilized.

7.3.1 Posture Controller

The posture controller in this control strategy consist of 17 classical controllers, one for each actuated joint. The input to the controllers is the error between the pre-generated posture trajectories and the actual position of the robots.

Webots representation of AAU-BOT1

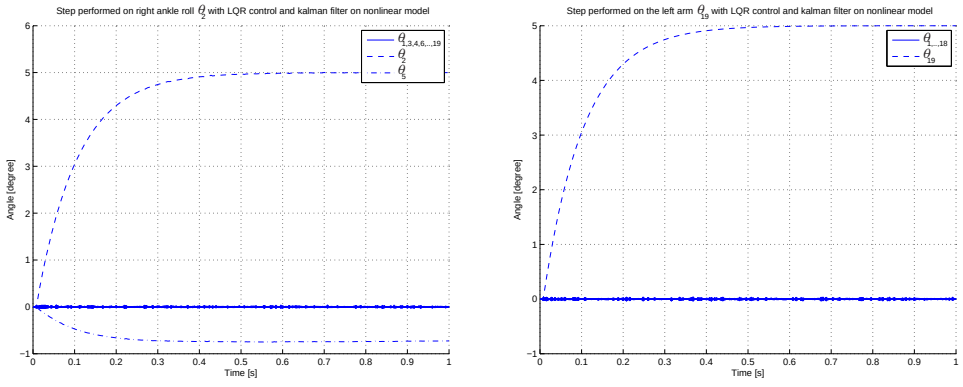
The virtual robot in Webots is implemented with the default controller found in Webots. This controller is not a usual P controller, known from classical control theory.

In Webots position control can be considered in three steps[Cyberbotics Ltd., 2008]. The first step is to determine which angle the controller should be in. This is determined by the proposed trajectories. The second step is performed by the build in proportinal controller that computes the current velocity V_c as shown in Equation (7.17). The third step is carried out by the Open Dynamics Engine (ODE). At every simulation step the P controller calculates the current velocity V_c by the following equation:

$$V_c = P_{gain}(P_t - P_c) \tag{7.17}$$

The current velocity is the control input to the ODE. The way the ODE handles the control input are described in [Smith, 2006]. Here it is stated that the torque is calculated by effectively looking one time step into the future and thereby calculate the needed torque to obtain the desired velocity. This ensures that the joint is brought up to speed in one time step, provided that it does not take more torque than allowed. This feature is computationally expensive, but is robust and stable. As regular control does not look one time step into the future and calculate the necessary torque, this controller in Webots can not be compared directly with a regular classical P controller. It is however chosen to use the build in controller in Webots since the posture controller in Control Strategy A could not stabilize Webots by using torque as input.

A control proportional gain of 10 and a max torque of 150 Nm are used when the two steps are performed on the virtual robot in Webots and can be seen in Figure 7.9.



(a) Step on θ_2 (b) Step on θ_{19}

Figure 7.9: Steps performed on the virtual robot. The joints share the same dynamic due to the Webots controller.

The first steps can be seen in Figure 7.9(a) where a step of 5 degree on ankle roll is performed. Here it is possible to see that 150 Nm is not enough for θ_5 to stay in zero.

The virtual robot is in DSP at all time during this test. The net step is on the left arm and is also of 5 degree. Here is can be seen that it settles after approximately 0.5 seconds. This is a convenient rate, it is important that the system is not to fast as the system is greatly influenced by cross couplings, and if the arm moves to fast can cause the body to moved unwanted. The posture controller for Webots are now configured and performs satisfying.

AAU-BOT1

This section describes the use of the EPOS amplifiers, when using them as part of the posture controller. The information used in the setup comes from [Maxon Motors, 2007a] and [Maxon Motors, 2007b]. The controller has to be versatile such that it can be used in both SSP-R, SSP-L and DSP. This is necessary to avoid changing controller gains during operation.

The general control strategy of the EPOS amplifiers can be seen in Figure 7.10. Profile Position Mode uses two parts of it:

- Position Control loop, generating the Current demand value
- Trajectory Generator generating the Position demand value.

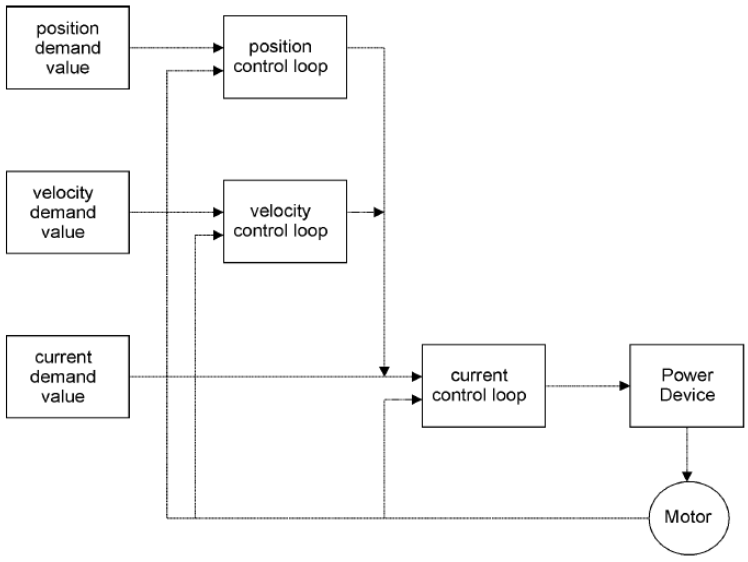


Figure 7.10: Control Strategy of the EPOS amplifiers. The Position demand value and the Velocity demand value are generated by the Trajectory Generator, the Current demand value is taken directly from a command (see Appendix J). Only one demand value is used at a time. [Maxon Motors, 2007a]

Position Control loop: The Position Control loop has the structure seen in Figure 7.11. The control parameters in Figure 7.11 is listed along with their attributes in Table 7.1. In addition to a classical discrete PID regulator, the Position Control loop consists of two Feed Forward constants

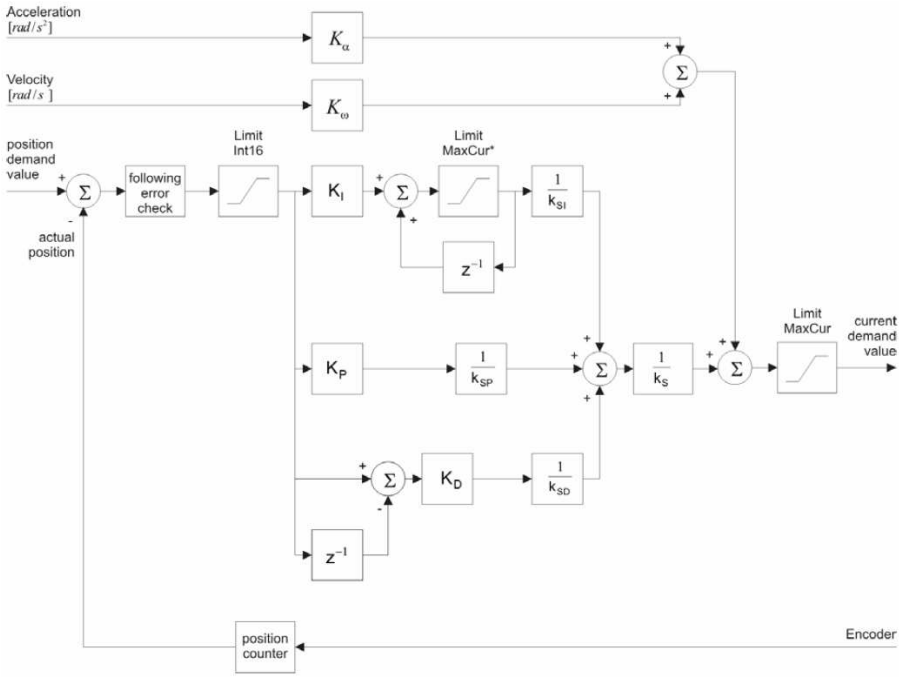


Figure 7.11: Block diagram of the Position Control loop. [Maxon Motors, 2007b]

- K_ω: Velocity Feed Forward Factor, that compensates for viscous friction.
- K_α: Acceleration Feed Forward Factor, that compensates for the moment of inertia and provides additional current in case of a high acceleration.

By adding these two to the transfer function, the DC motor model becomes:

$$\ddot{\theta}J = iK_t - \mu\dot{\theta} - \tau_c - \tau_L + K_\alpha\ddot{\theta}K_t + K_\omega\dot{\theta}K_t, \tag{7.18}$$

which means that if the Feed Forward parameters are chosen using Equation (7.19) and (7.20), the PID regulator is less sensitive to high values of μ and J.

$$K_\alpha = J/K_t \cdot 100 \tag{7.19}$$

$$K_\omega = \frac{30}{\pi}\frac{\Delta I}{\Delta\dot{\theta}} \tag{7.20}$$

As K_P, K_I and K_D are discretisized values of a PID controller, they are converted to continuous values using the EPOS sample time $T_S = 0.001$:

$$K_P/k_{\mathrm{SP}} = 100\frac{\mathrm{mA}}{k_{\mathrm{SP}}k_{\mathrm{s}}\mathrm{qc}}\frac{1}{k_{\mathrm{SP}}} = 100 \cdot \frac{0.001}{4 \cdot 2 \cdot \frac{1}{512\cdot 4}}\frac{1}{4} = 6.4 \tag{7.21}$$

$$K_I/k_{\mathrm{SI}} = 10\frac{\mathrm{mA}}{k_{\mathrm{SI}}k_{\mathrm{s}}T_S\mathrm{qc}}\frac{1}{k_{\mathrm{SI}}} = 10\frac{0.001}{32 \cdot 2 \cdot 0.001 \cdot \frac{1}{512\cdot 4}}\frac{1}{32} = 10 \tag{7.22}$$

$$K_D/k_{\mathrm{SD}} = 200\frac{\mathrm{mA}T_S}{k_{\mathrm{SD}}k_{\mathrm{s}}\mathrm{qc}}\frac{1}{k_{\mathrm{SP}}} = 100\frac{0.001 \cdot 0.001}{1 \cdot 2 \cdot \frac{1}{512\cdot 4}}\frac{1}{1} = 0.2048 \tag{7.23}$$

Table 7.1: Control parameters of Profile Position Mode.

Symbol	Name	Value	Unit
k_{SP}	Input scaling of P-Gain	4	[]
k_{SI}	Input scaling of I-Gain	32	[]
k_{SD}	Input scaling of D-Gain	1	[]
k_e	Encoder pulse no.	512	ticks
k_s	Encoder scaling	2	[]
P	Position		[qc]= $\frac{1}{k_e \cdot 4}$
K_P	Position Regulator P-Gain	100†	$\frac{\mathrm{mA}}{k_{\mathrm{SP}} k_s \mathrm{qc}}$
K_I	Position Regulator I-Gain	10†	$\frac{\mathrm{mA}}{k_{\mathrm{SI}} k_s T_S \mathrm{qc}}$
K_D	Position Regulator D-Gain	200†	$\frac{\mathrm{mA} T_S}{k_{\mathrm{SD}} k_s \mathrm{qc}}$
K_ω	Velocity Feed Forward Factor	1600†	$\frac{\mu\mathrm{A}}{\mathrm{rad}/s}$
K_α	Acceleration Feed Forward Factor	1000†	$\frac{\mu\mathrm{A}}{\mathrm{rad}/s^2}$

†: Default setting of the editable parameters.

The value of K_I is a considered a bit high compared to K_P, however the limits on the acceleration and velocity in the EPOS amplifiers can change the total transfer function of the DC motor to something that has a benefit of a high I-Gain. Implementing the controller on the DC motor model for the arm yields the step response seen in Figure 7.12.

Trajectory generator: To provide the input to the Position Control loop, the reference input is fed into a Trajectory generator, which generates a trajectory based on the following configurable parameters:

- Maximum Velocity
- Maximum Acceleration
- Maximum Deceleration
- Trajectory shape

By submitting the left arm to a 10° step yields the result seen in Figure 7.13. Due to the fact that this step response is stable, has a steady state error of 0% and is relatively fast, it is chosen not to change the settings of the EPOS amplifiers. By submitting all the joints to a step, it is seen that this response is universal for all the joints.

Double Actuated Joints: The intention with double actuated joints is to distribute the torque of a joint evenly amongst two DC motors. To demonstrate the problem, a test has been made on the left knee with both amplifiers in Profile Position mode, see Figure 7.14. To use the double actuated joints, three different control approaches has been considered:

1. The first solution is using a feature in the EPOS amplifiers called "'Master encoder mode"'. This basically works by having one motor as a master motor which receives the reference signal from an outer control loop. The second motor receives its

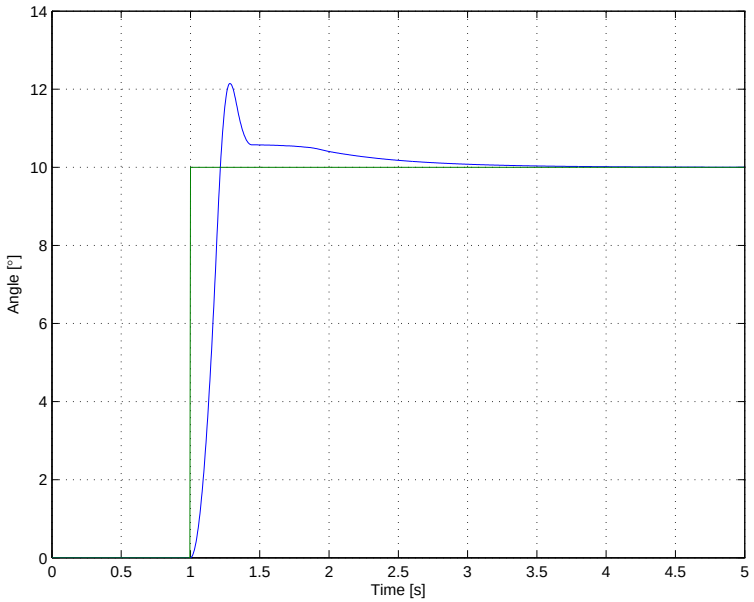


Figure 7.12: Step response of the DC motor model with the implemented Position Control loop. The step response is nonlinear due to the fact that it is submitted to the nonlinear DC motor model derived in Chapter 5.3 and parameter estimated in Appendix A.1, which has a high coulomb friction.

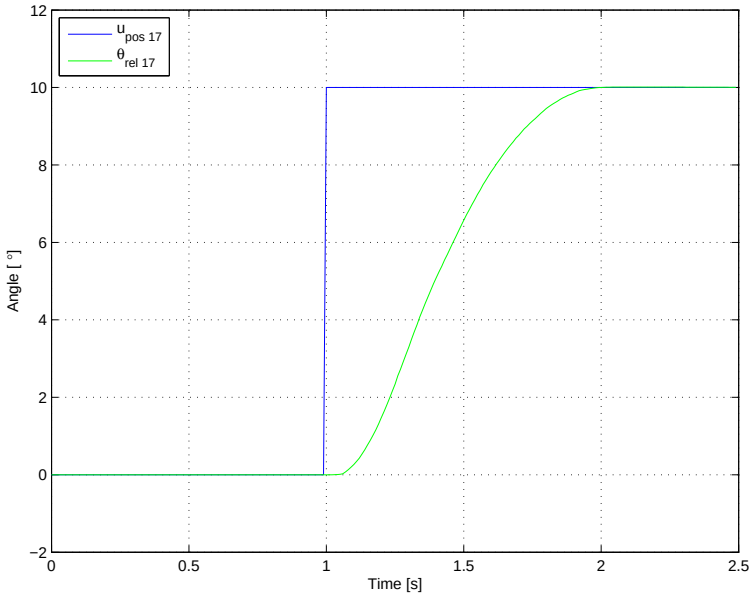


Figure 7.13: Step response of the left arm by submitting it to a step of 10°, with both the trajectory generator and the Position Controller activated.

reference signal from the master motors encoder signal and follows this signal as fast as possible. This feature would be very easy and straight forward to use as it is already incorporated in the EPOS amplifiers. However, the synchronous motors are interconnected by a belt and if even a small bias occur in the encoders or small elasticity is present in the belt then this means that the motors are not running completely synchronous, resulting will be that one motor is doing most the work and the slave motor would just follow the first motors without contributing with torque, before the first motor encounter its torque limit.

2. The second solution is to use the Current Mode controller in the EPOS amplifiers. Due to the fact that the torque is derived directly from the current and angular velocity, the controllers in the EPOS amplifiers should not counteract each other using this procedure.

3. The third solution is to turn off one of the motors, thus making it a single actuated joint.

Due to the fact that the Hardware Supervisor was not implemented at the time of the controller testing, the dual actuated joints has not been verified and designed for control Strategy B. Thus, the double actuated joints are actuated with a single motor (Control approach # 3) to avoid that the amplifiers on the double actuated joints will counteract each other. This yields the step response seen in Figure 7.15.

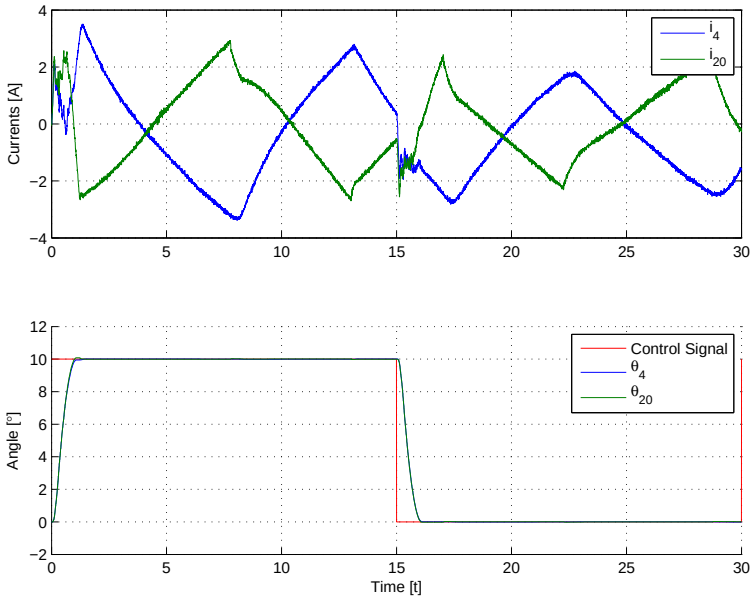


Figure 7.14: Measurement from step input test on the right knee with both motors activated.

7.3.2 Verification of Position Controller for Control Strategy B

Control strategy B is verified in Appendix B.2.3. All the controllers are stable and has a steady state error of 0. By submitting joint #2 and joint # 19 to a step of 3° and

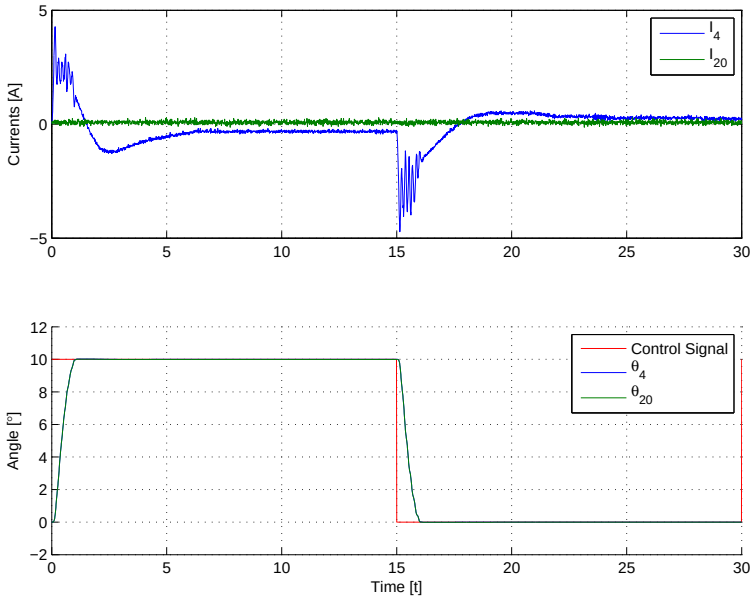
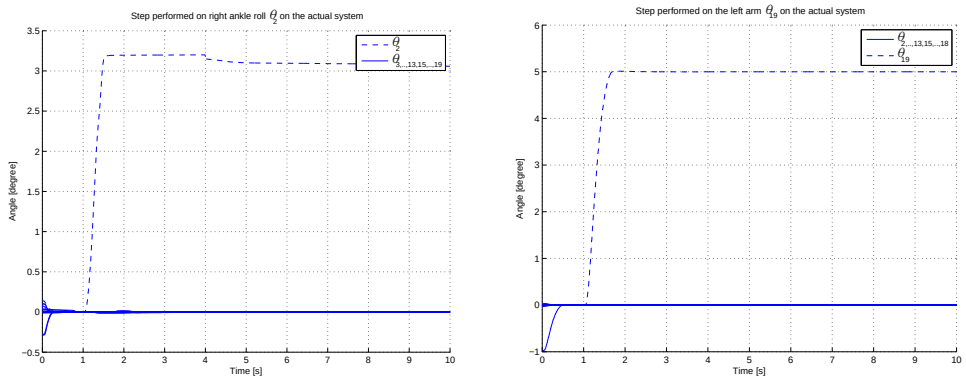


Figure 7.15: Measurement from step input test on the right knee with only one motor activated.

5°, respectively, yields the step response seen in Figure 7.16. Control Strategy B is considered to be working, and will be used in the final test of submitting the trajectories to **AAU-BOT1**.



(a) Step on θ_2. The change at 4 s is due to the fact that **AAU-BOT1** is in DSP.

(b) Step on θ_{19}.

Figure 7.16: Steps performed on the actual system.

7.3.3 Torso Controller

To control the position of the ZMP in real time when executing the trajectories described in Chapter 6, a controller for the upper body (The torso and the arms) is created. The ZMP of the upper body can be moved along both the x and y axis using only joint 16 and 17 (Waist Pitch and Roll, respectively).

Model

The torso control is to be used in Webots, which has torque as input and the angle of the joint as output. The torso controller will be modeled by using parts of the DC motor model seen in Figure 5.2 on page 71. τ_L and τ_F is modeled by using Equation (7.24) and (7.25). The FTS sensors broke before the torso controller was designed, thus the torso control has not been designed for use on the actual **AAU-BOT1**.

$$\tau_L = \sin(\theta) \cdot |b_t| \cdot m \cdot g \tag{7.24}$$

$$\tau_F = -\dot{\theta} \cdot \mu \tag{7.25}$$

where

τ_L	is the load on the DC motor.
θ	is the angle of the joint.
$\lvert b_t \rvert$	is the length of the CoM vector of the torso (b_t).
m	is the mass of the torso.
g	is the gravitational acceleration constant.
τ_F	is the friction of the joint.
$\dot{\theta}$	is the angular velocity of the joint.
μ	is the viscous friction coefficient of the joint.

Equation (7.25) assumes that the coulomb friction and the stiction are negligible. The effect of the arms is also neglected, due to the relatively low weight. Inserting Equation (7.24) and (7.25) into the model yields the following transfer function:

$$\ddot{\theta} = \frac{\tau_M + \sin(\theta) \cdot |b_t| \cdot m \cdot g - \dot{\theta} \cdot \mu}{J} \tag{7.26}$$

From this equation, the states ($\vec{x}$) and the input ($\vec{u}$) can be seen:

$$\vec{x} = \begin{bmatrix} \theta_{16} & \theta_{17} & \dot{\theta}_{16} & \dot{\theta}_{17} \end{bmatrix}^{\mathrm{T}} \quad \vec{u} = \begin{bmatrix} \tau_{16} & \tau_{17} \end{bmatrix}^{\mathrm{T}} \tag{7.27}$$

Using a 1st order Taylor approximation to linearize Equation (7.26) yields the following state space equation:

$$\begin{bmatrix} \dot{\theta}_{16} \\ \dot{\theta}_{17} \\ \ddot{\theta}_{16} \\ \ddot{\theta}_{17} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{|b_t|mg}{\boldsymbol{J}_y} & 0 & \frac{-\mu_{16}}{\boldsymbol{J}_y} & 0 \\ 0 & \frac{|b_t|mg}{\boldsymbol{J}_x} & 0 & \frac{-\mu_{17}}{\boldsymbol{J}_x} \end{bmatrix} \begin{bmatrix} \theta_{16} \\ \theta_{17} \\ \dot{\theta}_{16} \\ \dot{\theta}_{17} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{\boldsymbol{J}_y} & 0 \\ 0 & \frac{1}{\boldsymbol{J}_x} \end{bmatrix} \begin{bmatrix} \tau_{M16} \\ \tau_{M17} \end{bmatrix} \tag{7.28}$$

$$\begin{bmatrix} \theta_{16} \\ \theta_{17} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_{16} \\ \theta_{17} \\ \dot{\theta}_{16} \\ \dot{\theta}_{17} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tau_{M16} \\ \tau_{M17} \end{bmatrix} \tag{7.29}$$

Parameter Estimation

The parameters of the Torso controller are:

- The mass of the torso (m).
- The moment of inertia of the upper body ($\vec{J}$).
- The friction of the joints (μ).

The mass of the torso is found in table C.1 on page 183 to be 21.60 kg. The moment of inertia is calculated by transposing the Inertia tensor to the joint using Equation (7.30) and (7.31) [Serway et al., 2000, p. 304]:

$$\boldsymbol{J}_x = \boldsymbol{J}_{Torso,x} + m \cdot |\vec{b}_t|^2 \tag{7.30}$$

$$\boldsymbol{J}_y = \boldsymbol{J}_{Torso,y} + m \cdot |\vec{b}_t|^2 \tag{7.31}$$

The friction (μ) is set to 1 [$\frac{\text{N m s}}{\text{rad}}$], as this is the current setting in Webots. To enable the controller to be used with Webots, it is discretisized with the MATLAB$^{\text{TM}}$ `c2d` command.

Linear Quadratic Gaussian Controller Design

By examining the controlabillity matrix and the observability matrix, the system is determined to be both controllable and observable. To control the Torso, a Linear Quadratic Gaussian (LQG) controller is designed. The process noise $\boldsymbol{Q}_K$, observation noise $\boldsymbol{R}_K$, process cost $\boldsymbol{Q}_C$ and input cost $\boldsymbol{R}_C$ matrices can be seen in Equation (7.32).

$$\boldsymbol{Q}_K = 10^{-5} \cdot \boldsymbol{I}^{4\times4} \tag{7.32}$$

$$\boldsymbol{R}_K = 10^{-4} \cdot \boldsymbol{I}^{2\times2} \tag{7.33}$$

$$\boldsymbol{Q}_C = \text{diag}(\begin{bmatrix} 500 & 500 & 2000 & 2000 \end{bmatrix}) \tag{7.34}$$

$$\boldsymbol{R}_C = 500 \cdot \boldsymbol{I}^{2\times2} \tag{7.35}$$

Initially $\boldsymbol{Q}_C$ and $\boldsymbol{R}$ where set to $200 \cdot \boldsymbol{I}^{4\times4}$ and $100\boldsymbol{I}^{2\times2}$, respectively. However by hand tuning to this value, a better performance was achieved when implemented in Webots, with regards to rise time and stability. As the controller is to follow a reference, two reference gain matrices ($\boldsymbol{N}_x$ and $\boldsymbol{N}_u$) are calculated by using the MATLAB$^{\text{TM}}$ function `refi.m` [Franklin et al., 1997, p. 313]. In addition to the system and controller, `refi.m` requires a matrix ($\boldsymbol{H}_r$) that determines which states are to be controlled. As the desired controllable states are the angles, $\boldsymbol{H}_r$ is:

$$\boldsymbol{H}_r = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{7.36}$$

ZMP controller

To determine the input for the Torso control, a outer loop is created, which can be seen in Figure 7.17. As the Torso control does not take the effects of acceleration into account when achieving balance, the outer loop is designed to be slow and have zero steady state error. This is done by using a PI controller with a P gain set to 1 and a I gain set to 2.

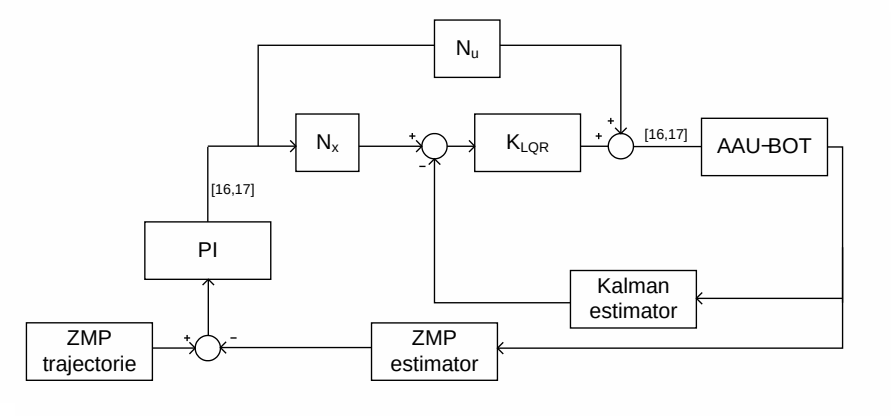


Figure 7.17: Design of Outer loop for Torso ZMP Control.

Due to the fact that the ZMP coincides with the GCoM when using static gait, the maximum correction of ZMP by moving the torso can be found using Equation (2.1) from page 29:

$$x_{\text{GCoM}} = \frac{\sum_{n=1}^{N_{\text{Links}}} \vec{x}_n \cdot m_n}{\sum_{n=1}^{N_{\text{Links}}} m_n} \tag{7.37}$$

$$\Delta x_{\text{GCoM}} = \frac{\Delta x_{\text{Torso}} \cdot m_{Torso}}{\sum_{n=1}^{N_{\text{Links}}} m_n} \tag{7.38}$$

$$\max \Delta x_{\text{GCoM}} = \frac{\sin \max \theta_{16,\max} \cdot |b_t| \cdot m_t}{\sum_{n=1}^{N_{\text{Links}}} m_n} \tag{7.39}$$

$$= \frac{\sin(13) \cdot \sqrt{0.019^2 + 0^2 + (-0.341)^2} \cdot 21.6}{70} = 0.024 \text{ [m]} \tag{7.40}$$

$$\max \Delta y_{\text{GCoM}} = \frac{\sin(56) \cdot \sqrt{0.019^2 + 0^2 + (-0.341)^2} \cdot 21.6}{70} = 0.087 \text{ [m]} \tag{7.41}$$

Verification of Torso Controller and ZMP Controller

The description of the verification of the torso controller can be found in Appendix B.2.2, it is tested by implementing it on the Webots simulation, as seen in Figure 7.2. The torso controller is found to have a fairly large steady state error, however this does not matter as the ZMP Controller has an Integral Gain.

7.4 Observers

The purpose of the observers is to extract measurements of the current stance phase and the position of ZMP, for usage in the controllers.

7.4.1 Phase Observer

The Phases are observed using the Force Torque Sensors (FTS), the orientation of the feet (O_{llp} and O_{rlp}) and the angle of the toes (θ_1 and θ_{14}).

$$\vec{Q} = \begin{cases} q_1 & \text{if} \quad F_{zR} < \epsilon_1 \wedge F_{zL} > \epsilon_2 \wedge O_{llp} < \epsilon_4 \wedge \theta_{14} < \epsilon_6 \\ q_2 & \text{if} \quad F_{zR} > \epsilon_1 \wedge F_{zL} < \epsilon_2 \wedge O_{rlp} < \epsilon_4 \wedge \theta_1 < \epsilon_5 \\ q_3 & \text{if} \quad F_{zR} < \epsilon_1 \wedge F_{zL} > \epsilon_2 \wedge O_{llp} < \epsilon_4 \wedge \theta_{14} > \epsilon_6 \\ q_4 & \text{if} \quad F_{zR} > \epsilon_1 \wedge F_{zL} < \epsilon_2 \wedge O_{rlp} < \epsilon_4 \wedge \theta_1 > \epsilon_5 \\ q_5 & \text{if} \quad F_{zR} > \epsilon_1 \wedge F_{zL} > \epsilon_2 \wedge O_{rlp} < \epsilon_3 \wedge O_{llp} < \epsilon_4 \wedge \theta_1 < \epsilon_5 \wedge \theta_{14} < \epsilon_6 \wedge x_r < x_l \\ q_6 & \text{if} \quad F_{zR} > \epsilon_1 \wedge F_{zL} > \epsilon_2 \wedge O_{rlp} < \epsilon_3 \wedge O_{llp} < \epsilon_4 \wedge \theta_1 < \epsilon_5 \wedge \theta_{14} < \epsilon_6 \wedge x_r > x_l \\ q_7 & \text{if} \quad F_{zR} > \epsilon_1 \wedge F_{zL} > \epsilon_2 \wedge O_{rlp} < \epsilon_3 \wedge O_{llp} < \epsilon_4 \wedge \theta_1 > \epsilon_5 \wedge x_r < x_l \\ q_8 & \text{if} \quad F_{zR} > \epsilon_1 \wedge F_{zL} > \epsilon_2 \wedge O_{rlp} < \epsilon_3 \wedge O_{llp} < \epsilon_4 \wedge \theta_{14} > \epsilon_6 \wedge x_r > x_l \\ q_9 & \text{if} \quad F_{zR} > \epsilon_1 \wedge F_{zL} > \epsilon_2 \wedge O_{llp} > \epsilon_4 \wedge \theta_1 > \epsilon_5 \wedge x_r < x_l \\ q_{10} & \text{if} \quad F_{zR} > \epsilon_1 \wedge F_{zL} > \epsilon_2 \wedge O_{rlp} > \epsilon_3 \wedge \theta_{14} > \epsilon_6 \wedge x_r > x_l \end{cases} \tag{7.42}$$

where:

ϵ_n	is threshold n, determined experimentally with **AAU-BOT1**
F_{zR}	is the force excerted in the z-direction on the right foot.
F_{zL}	is the force excerted in the z-direction on the left foot.
O_{llp}	is the pitch of the left foot.
O_{rlp}	is the pitch of the right foot.
θ_1	is the angle of the right toe.
θ_{14}	is the angle of the left toe.

The Phase Observer has not been implemented on the Webots simulation. The Webots simulation does not contain a force torque sensor, so to estimate the stance phase, the Phase Estimator designed in Section 5.6 is used. This is also the reason it is the phase estimator that is featured in Figure 7.1 and 7.2. One of the FTS amplifier on **AAU-BOT1** broke before the Phase Observer was made, so it has never been tested or used.

7.4.2 ZMP Estimator

The purpose of the ZMP estimator is to observe the position of the ZMP, for use in the ZMP Controller. The ZMP is measured using the FTS amplifiers and and the positions of the feet. The inputs and outputs of the ZMP estimator can be seen in Figure 7.18. The measurements obtained in the FTS for both feet are:

- The force in the x-direction: F_x

- The force in the y-direction: F_y
- The force in the z-direction: F_z
- The moment around the x-axis: M_x
- The moment around the y-axis: M_y
- The moment around the z-axis: M_z

To estimate the ZMP, the observations of Figure 7.19 are used in Equation (7.43). The ZMP estimator is to only function correctly with the feet flat on the ground, as this is the primary state of **AAU-BOT1**, and the additional complexity of adding support for heel-strike does not add significant value, as the angle of the feet relative to the ground should always be small. The x-value of the ZMP for SSP is calculated using Equation (7.44).

$$l_x = \frac{M_x}{F_z} \tag{7.43}$$

$$x_{\text{ZMP}} = P_0 + P_{FTS,x} + l_x \tag{7.44}$$

The same procedure is done for the y_{ZMP}. In the Double Support phase, the ZMP is

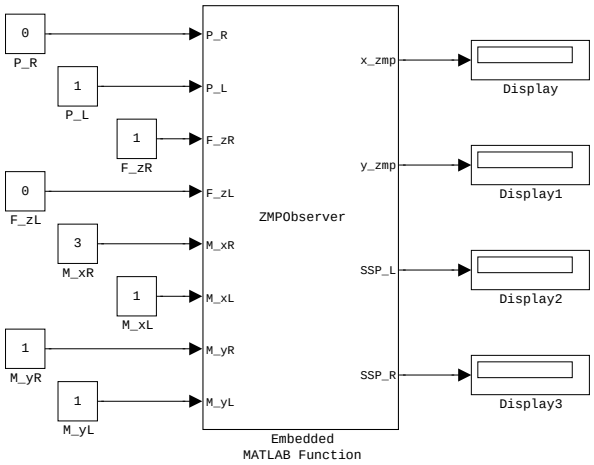


Figure 7.18: ZMP estimator, rendered as a Simulink block.

found by combining the ZMP for the left and the right foot, using Equation (7.45):

$$P_{\text{ZMP}} = \frac{F_{z,L} \cdot x_{\text{ZMP}} L + F_{z,R} \cdot x_{\text{ZMP}} R}{F_{z,L} + F_{z,R}} \tag{7.45}$$

The ZMP estimator was not verified due to the lack of force torque sensors in Webots and the FTS amplifiers in **AAU-BOT1** broke down before it was implemented. The ZMP is estimated using Equation (2.7) and (2.8) and measured data from Webots.

7.5 Supervisor

To ensure that **AAU-BOT1** does not damage itself, a supervisor is added. The Supervisor has three layers:

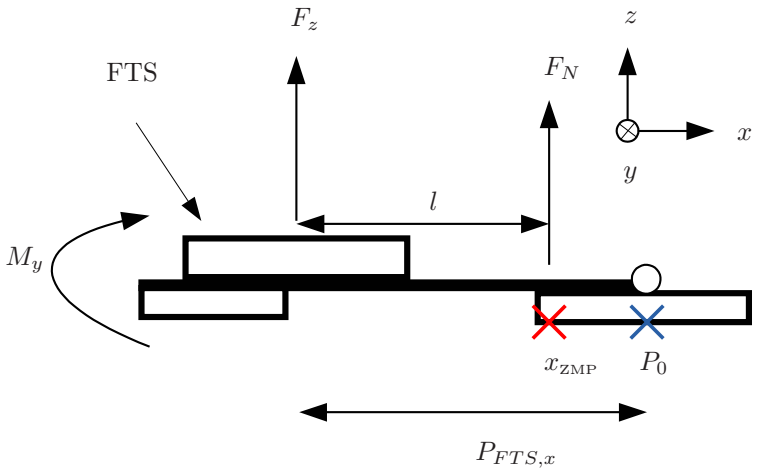


Figure 7.19: Free body diagram of foot. l is the distance between the ankle and the ZMP in the x-direction, F_N is the normal force excerted by the ground.

- Sofware layer within Simulink
  The Position commands sent to the Actuator Sink are limited to be only within the limits defined in Table 5.3 on page 76.
- Software layer in the EPOS Amplifiers
  The EPOS Amplifiers detects whether the angle of the motor is within 2000 ticks of the reference given by the Trajectorie generator.
- Hardware layer
  A circuit is designed to monitor the angles of the joints with potentiometers. If the angle is outside its permitted range, the circuit turns off the EPOS amplifier.

These three in combination are sufficient to ensure that **AAU-BOT1** does not damage itself using the joint that the motors are attached to. However, this does not ensure that the joints does not touch each other further down the kinematic chain. Additionally, a supervisor for when the controllers are in other control modes than Profile Position Mode is not designed.

7.6 Summary of Control

In this chapter, multiple controllers for two different controller strategies has been proposed. The ZMP controller, phase observer and the ZMP estimator are not implemented on the actual system due to the fact that one of the FTS amplifiers broke. The ZMP controller was however implemented with the virtual robot in Webots as the ZMP could be estimated from other measurements.

Control Strategy A: It was possible to stabilize the nonlinear model with the proposed LQG controller. This showed that the weights on the states had to be significantly larger on the joints in the beginning of the kinematic chain, in order to stabilize the nonlinear model. This makes sense as they are most influenced by the other states. Furthermore it was observed that the steady state errors on the model were

large. It is clear that integral action is needed to minimize these steady state errors if further development on this control strategy has to be done.

Control Strategy B: In this section, a control strategy for posture control of **AAU-BOT1** was developed. The main strategy of this was to utilize the build-in controllers in Webots and in the EPOS amplifiers, putting a PID controller on each of the joints except joint # 16 and 17, which were controlled with a LQG controller. This controller was capable of controlling both Webots and **AAU-BOT1**.

ZMP Controller: To balance the virtual robot in Webots and **AAU-BOT1**, a ZMP controller was designed. Its main feature is to be very slow, to keep the acceleration of the torso to a minimum, as this otherwise would move the ZMP in the other direction than intended.

Phase Observer: To observe which phase **AAU-BOT1** is in, the phase observer was designed. The phases are determine from the force torque sensors.

ZMP estimator: To enable the ZMP controller to work on **AAU-BOT1** the ZMP estimator was created.

Supervisor: To prevent **AAU-BOT1** in destroying itself, a number of limitations are implemented. During tests these have shown to be efficient to prevent damages.

Chapter 8

System Test

*This chapter deals with the test of the complete system. The test will be carried out in two steps. The first step is to test the complete system on the virtual **AAU-BOT1** in Webots. After this the complete system will be tested on the actual **AAU-BOT1**, only the posture controller is tested here as it is not possible to obtain a ZMP measurement.*

8.1 Introduction to Complete Test

In the previous chapters a complete system for the **AAU-BOT1** has been designed and developed. This includes models such that trajectories for static gait and controllers could be developed. Furthermore instrumentation of the actual robot together with a software platform was completed. All this is needed to conduct this final test.

The test is split up into two parts. This is chosen since both the Webots representation of the **AAU-BOT1** and the actual **AAU-BOT1** are tested. The first part is the virtual robot in Webots which is tested with Control Strategy B, described in Section 7.3 on page 128. The virtual **AAU-BOT1** is tested with both the posture controller and the ZMP-controller.

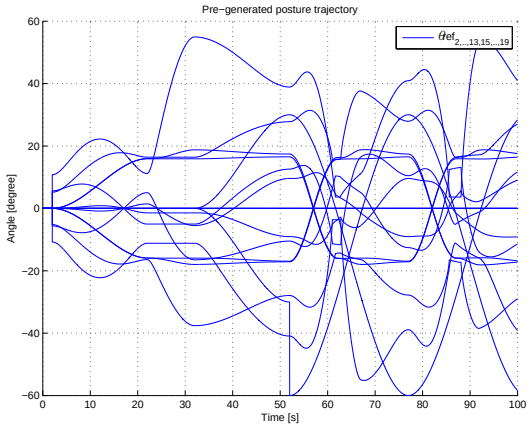
Hereafter the actual **AAU-BOT1** is tested. The actual **AAU-BOT1** is also tested with Control strategy B, described in Section 7.3 on page 128. As mentioned in the Section 3.6.5 on page 42 it has not been possible to extract ZMP measurements from the actual robots due to malfunctioning hardware, hence it has not been possible to test the ZMP controller on the actual **AAU-BOT1**.

The trajectories for the posture controller is 100 s long and contains a startup phase followed by two walking steps, the trajectories for the posture controller can be seen in Figure 8.1(a) and corresponding support phases can be seen in Figure 8.1(b).

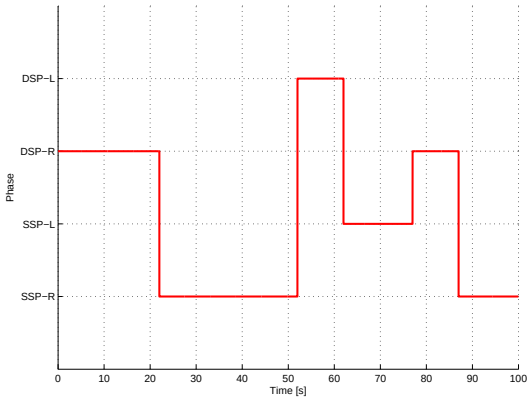
The startup phase includes a settling sequence of 2 s in the beginning, this is included to ensure that the robot is settled in an upright position. After this the actual start phase begins by taking half a step followed by two regular steps.

8.2 Virtual Robot in Webots

The trajectories from Figure 8.1(a) are now tested together with the two controllers from Control Strategy B. The first controller is the posture controller and the second is the ZMP controller which maintains the robot in a stable state. In Figure 8.2(a) the measured joint angles can be seen and in Figure 8.2(b) the error between the input trajectories and



(a) Input trajectory.



(b) Estimated support phase of input trajectory.

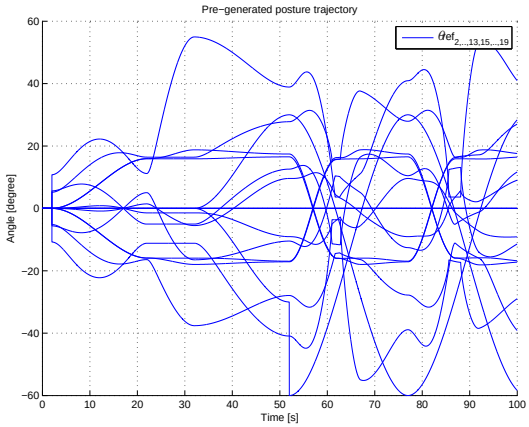
Figure 8.1: The trajectories and the phase of the input trajectory.

the measured joint angles can be seen. The trajectories are made such that the robots is moving smoothly and relatively slow such that static gait is possible and the posture controller is able to track the trajectories.

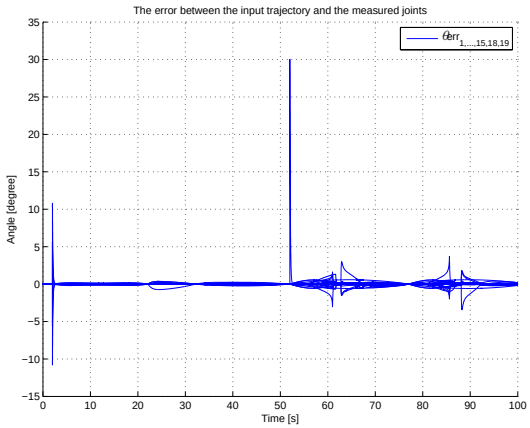
Figure 8.2(b) shows the errors between the trajectories and the measured joint angles. The largest spike is seen at time 52 s, this is an error in the trajectory for the right arm when switching phase. The errors seen at time 2 s, 62-63 s and 85-87 s are all caused by a offset mismatch between the trajectory generator and the inverse kinematics. The mentioned errors are relatively simple to correct but unfortunately they where discovered late in the project period and is therefore not corrected. Besides the spikes the remaining errors are relatively small and roughly not larger than 1 degree which is considered satisfying.

The performance of the ZMP controller can be seen in Figure 8.3, that includes the input ZMP reference and the measured ZMP for both the x- and y-direction. Furthermore the error between the reference and output for both directions can be seen.

In Figure 8.3(b) and (d) the error for the ZMP controller in both directions can be seen. Two spikes is present both in the x- and y-direction and can be seen at time 52 s and 78 s. Both spikes are connected with the phase transition. The ZMP controller in



(a) Measured joint angles.



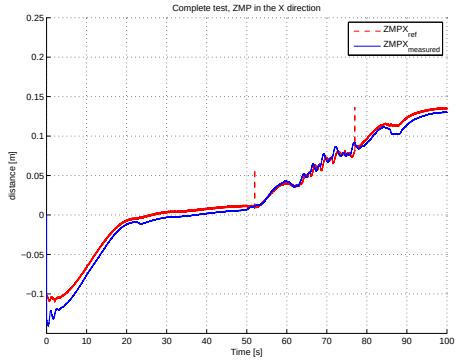
(b) The error between the measured angle and the trajectory.

Figure 8.2: Static gait trajectory on the virtual **AAU-BOT1** in Webots.

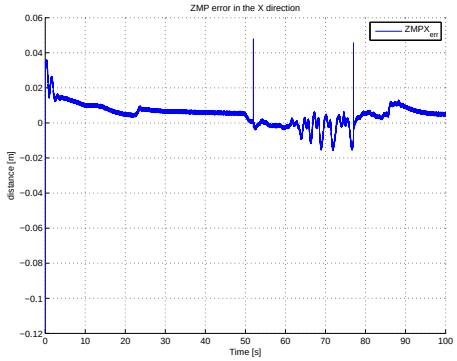
the x-direction has a mean following error of 0.5 cm. In the y-direction the mean error is 0.07 cm. The errors are considered small since the controller is developed to be slow and steady but not necessarily accurate. The ZMP controller ripples at time 60-80 s. The ripples occurs due to the movement of the upper body. By moving the upper body the lower body will be influenced by the same torque just in the opposite direction. This is unavoidable, but to suppress this phenomenon the controller can be slowed down such that it is steady, when tracking the ZMP reference. If it is slowed down too much it will not be fast enough to prevent the robot from falling. The controller has to handtuned, if it is used for e.g. dynamical walk.

8.3 Actual AAU-BOT1

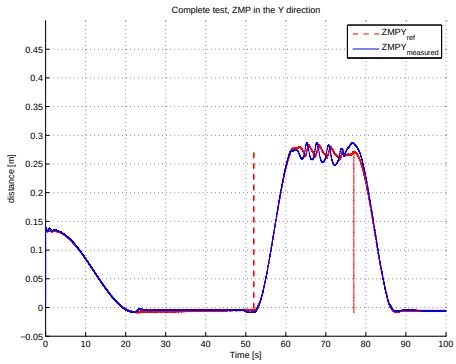
The static gait trajectory shown in Figure 8.1 is now tested on the actual **AAU-BOT1**. Figure 8.4(a) shows the measured angles and Figure 8.4(b) shows the errors between the



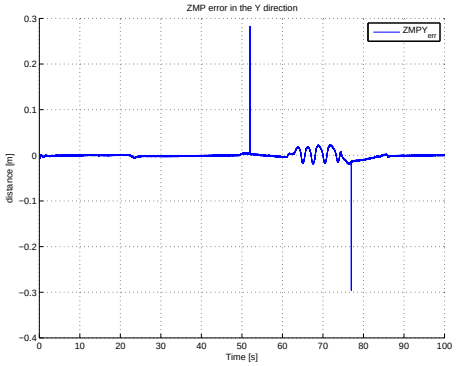
(a) ZMP reference and measured ZMP in the x-direction.



(b) Error between the measured and the reference in the x-direction.



(c) ZMP reference and measured ZMP in the y-direction.



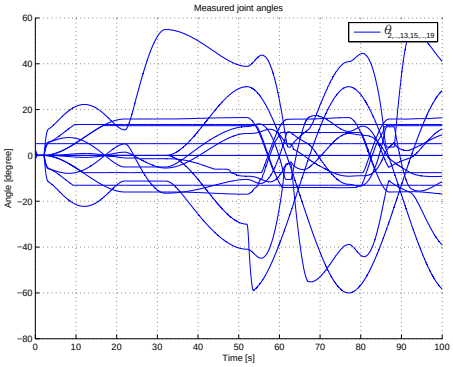
(d) Error between the measured and the reference in y-direction.

Figure 8.3: The ZMP trajectory and the measured ZMP on Webots

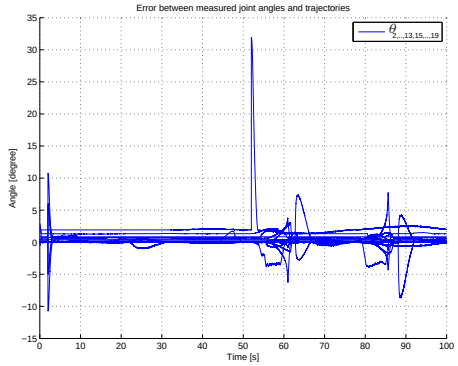
input trajectories and the measured joints. Figure 8.4(b) show similar errors as noticed when testing the virtual **AAU-BOT1**. The large spikes to time 2 s, 52 s, 60-62 s, and 88-90 s are caused, since the dynamics of the **AAU-BOT1** can not follow some of the joint trajectories fast enough. Although the trajectories for the feet and torso in cartesian space are made soft and continuously the trajectories are truncated by the inverse kinematics and this makes the trajectories discontinuously in some points. This truncation can be avoided by ensuring that the mismatch between the inverse kinematics and the cartesian space trajectories.

Another interesting observation is seen in the time frames 55-60 s and 80-85 s, here two upside down trapezoidal signals can be seen. This is the two knee joints which are limited by the current. The reason why there is not enough energy, is because only one DC motor is connected to each of the 6 joints that are supposed to be double actuated.

Joint limitations of the physical system are an important factor, this was observed on the actual system while performing the complete test. Ankle roll and ankle pitch were initially limited by physical constraints. After altering the mounting of the potentiometer on ankle roll this solved the problem for this joint. Ankle pitch are still limited to -10 degrees which is sufficient for dynamic gait, but is not as suitable for static gait.



(a) Input trajectory to the actual system.



(b) Trajectory on the actual system.

Figure 8.4: Static gait trajectory on actual system.

8.4 Summary of System Test

The trajectories together with Control Strategy B enabled the virtual robot to take steps and thereby move forward with static gait. The strategy proposes a ZMP controller which maintains stability while walking. The ZMP controller worked as intended in this complete test on the virtual **AAU-BOT1**.

The performance of the posture controller on the actual system are acceptable, but the physical constraints of the system complicates the static gait, as static gait requires larger joint angle fluctuations than dynamic walk. The performance of the ZMP controller are yet unknown as the **AAU-BOT1** is still not able to measure the ZMP due to a faulty FTS amplifier.

The total performance of the actual **AAU-BOT1** is still unknown, but it is assumed that it is possible to obtain static gait if the actual physical constraints of **AAU-BOT1** are included in the trajectory generation and proper ZMP measurements are obtained. In order to do that, further tests has to be conducted, since the trajectory has only been tried once.

Chapter 9

Epilogue

In this chapter the epilogue to this master's thesis is given. It contains a discussion of some of the main problems and possibilities discovered during the project. The discussion is divided into the following key areas: instrumentation, software architecture, modeling, trajectory generation and control. During the discussion the results obtained throughout this master's thesis are compared to the entire AAU-BOT1 project starting with the mechanics group, who was the first project group working on the ***AAU-BOT1*** *research project. After this a conclusion to the entire master's thesis is given and ideas for future work is presented.*

9.1 Discussion

This section will mainly bring up areas of the thesis that has been problematic to some extent and discuss them.

9.1.1 Instrumentation Strategy

A complete solution to the instrumentation strategy is described and proposed. The complete solution is implemented as intended. The following hardware parts has been dimensioned and applied to **AAU-BOT1**.

EPOS Amplifiers: Initially the mechanics group who previously worked on **AAU-BOT1** bought analog power amplifiers for the DC motors. It has not been possible to develop nor find a suitable on-board solution to sample all 23 analog amplifiers. It was thus decided to discard the analog amplifiers and buy new digital amplifiers of the type EPOS 70/10 from Maxon Motors. A great advantage of these amplifiers is that they can communicate via the CAN-Open protocol and thereby transmit sampled data via a shared bus. This gives an amplifier system that suppress noise well. The downside is the limited bandwidth of the CAN network, but this is solved by implementing 5 CAN networks serving the 23 EPOS amplifiers. Furthermore it have features as current limitation, position limitation and it shuts down if the implemented controller becomes unstable.

FTS and FTS Amplifiers: **AAU-BOT1** was delivered with two custom build FTS's, each containing 6 strain gauge full bridges. The instrumentation strategy proposed a solution where the analog signals from the full bridges have to be amplified,

filtered, sampled and transmitted via a digital bus. The development of such a device with these features was initiated, but this development was stopped since a similar device with the right specifications was discovered and bought. FTS amplifiers with build in RS485 were bought for the project. The amplifiers worked well when sampling the 12 full bridges. The FTS was calibrated and showed promising results, but yet still not satisfying results as an offset of 50 N was present, if this offset is removed only a RMS error of 2.4% is present. This large offset is assumed to be caused by an insufficient calibration test rig. An accident with a faulty EPOS caused one of the amplifiers to malfunction. This resulted in a system where only one FTS is operational and it has not been possible to test the total throughput.

IMU: An IMU has been bought, however it has not been implemented, as focus is placed elsewhere in the master's thesis. Even though the IMU is not implemented, it is still taken into consideration when designing the software architecture.

On-board Computer: An ordinary, but small laptop computer was bought and used as an on-board computer. It is an IBM with a 2.0 GHz Core 2 Duo processor. The hard drive of the on-board computer has been replaced by a Flash HDD, since this disk has no moving parts it is less prone to faults when submitted to a shaking environment during operation. The new Flash HDD is unfortunately very slow and should be replaced by a faster one, if the on-Board computer is used as a development tool. It does not have an effect on the system when running the real time target, since the program is loaded into memory and does not access the Flash hard drive during runtime.

9.1.2 Software

The software architecture is based on a Linux platform developed by Xubuntu which is the smallest and most optimized Ubuntu distribution. It runs with Kernel 2.6 and it was possible to implement the needed drivers for the CAN network. The software is divided into two main parts. The first part utilized a real time target for MATLAB™. The second part contains a multi threaded software architecture. The multi threaded software handles data between the actuators, sensors and updates the shared memory.

By using this approach, the controller which is running real time do not have to wait for the communication to take place. The threaded parts can run as fast as possible and update the shared memory with data from the sensors. Furthermore it updates the actuators with data from shared memory.

The software architecture has both advantages and disadvantages. The advantages are that it has a simple structure, and the layout has been used and proved useful for other projects. Furthermore there is no need for real time drivers for all the peripheral hardware.

The main disadvantage of this software architecture is that, if the system has insufficient processing power to run both the real time target and the multi threaded software, the real time target will uses all the processing power and leaving the multi threaded software unprocessed. This means the actuators and sensors are not updated and this can lead to a faulty system. To ensure that this is not the case, test has been performed to verify that the CAN busses have the necessary bandwidth. The test showed that a loss in packages of 13.6% when running 250 Hz. The packages loss is random distributed. By

lowering the update frequency to 200Hz the loss of packages are lowered to 1.9 %. A loss of 13.6% is not considered acceptable, this can be optimized by several means. One solution could be to set the actuator server and the sensor servers to the same priority as the real time target. Another solution is to use a custom build optimized Linux distribution, such that unnecessary modules are not compiled into the kernel. This will leave more processing power to the multi threaded software. If the software architecture is used in its current state, it should be considered to lower the update frequency to 200 Hz to lower the packages loss. It should also be noted that the throughput test the EPOS's are in position mode with all measurements of the EPOS amplifiers on. Using the EPOS's in current mode require less packages and reduces the load on the bus. Since the safety system has not been completed in such extend that it is justifiable to use current mode, this has not been tested.

When visualizing the model two methods have been developed. The first method was to use the MATLAB^{TM} plot function to representation the kinematics of the robot. As the dynamics of the **AAU-BOT1** does not include the extra degree of freedom that exist between robot and the ground, it is chosen to use a three dimensional robot simulator. This robot simulator gives a more realistic picture of how **AAU-BOT1** behave. It has been difficult to construct the virtual **AAU-BOT1** in Webots such that it behaves like the developed dynamical model. It has not been determined whether this is caused by a faulty dynamical model or a faulty configuration of Webots.

9.1.3 Model

The model is divided into multiple parts, and these are discussed separately:

DC motor model: The DC motor model is derived from physical and electrical models, and one joint i parameter estimated (the left arm). Using the estimated parameters the model is verified with a mean squared error of 12.1%. Due to the high interconnectivity of the measured states, this is regarded as a fairly good result. The parameter estimation is done for one joint only to show the procedure of finding the parameters for the DC motors. Unfortunately due to time constraints in this thesis, the rest of the joints are not parameter estimated, but it is a necessary task as the friction is large and has to be estimated along with the other parameters in the DC motor model. Furthermore a method to implementation of the double actuated joint has been developed. However this method has not been verified.

Kinematic model: The kinematic model determines the position, velocity and accelleration of all links' CoM, based on the angles of the joints of **AAU-BOT1**. It was made by using transformation matrices and the mechanical data of **AAU-BOT1**. The kinematic model is verified and works satisfactory.

Inverse Kinematic model: The inverse kinematic model determines the angles of the joints of **AAU-BOT1** from positions of the links' center of mass. By deciding both the orientation and position of the pelvis joint, and the foot that is not on the ground, a unique solution is found. The inverse kinematic model is verified and works satisfactory.

Dynamic model: The dynamic model was derived by using Lagrangian dynamics, combined with the Jacobian matrix and the kinematic model. Due to the immense size

of the model, this was done by using Maple. Problems occurred during development, the matrices which had to be computed caused Maple to fail due to insufficient amount of available RAM. To overcome this problem the matrices where exported to Simulink which was capable of linearizing it. The nonlinear dynamical model was also implemented in Simulink for test purposes. The dynamical model has not been verified toward the actual system, as the parameters of the actual system are still unknown and still subject to changes.

Foot Model: The foot model was derived in such extend that it is possible to determine the forces and the torques that the floor impose on the foot. The foot model also include the hybrid states that **AAU-BOT1** is in. However since the left FTS was not implemented due to a defect amplifier, it has not been possible to verify the foot model.

Phase Observer: The phase observer was developed based on the orientation of the feet (O_{llp} and O_{rlp}) and the angle of the toes (θ_1 and θ_{14}). However since the phases are observed by the FTS, it has not been possible to verify the phase observer, because one of the FTS amplifiers broke. It was neither possible to verify the phase observer in Webots since it is not possible to use a 6 axis FTS in Webots.

9.1.4 Trajectory Generation

A method for offline trajectories generation was developed and implemented. The developed method can be extended adn combined with adaptive trajectory adjustments, to ensure a more stable or energy efficient walk.

In order to develop trajectories for dynamic gait and static gait, human movement data through walking was studied, to obtain a natural movement of **AAU-BOT1**. It ended up with a method to generate trajectories based on high stability and energy efficiency. However due to an inaccurate dynamic model it has not been possible to simulate dynamic gait nor to get the most energy efficient trajectories. However the dynamic gait trajectory has been verified via visual test in Simulink, which means the presented method work as intended.

A static gait trajectory has been possible to obtain, via the inverse kinematic model and the kinematic model. In order to get a large support area, the toe feature was disabled and a minimum stability margin of 4.75 cm from all GCoM points to the edge of the support area were accomplished. However it was not possible to find the most energy efficient trajectory, due to the dynamic model. When knowledge about weight distribution and joint angle limits are obtained from the physical **AAU-BOT1**, these have to be added to the model and it will require a new evaluation of the trajectories, in order to find the most stable and energy efficient trajectories.

9.1.5 Controller

To control **AAU-BOT1**, three different controllers were developed:

Control Strategy A: Control Strategy A utilize the dynamic model to create a Linear Quadratic Gaussian (LQG) controller, utilizing the knowledge about the joints interference on each other to make a more stable controller. After tweaking the controller it was discovered that the model was not an accurate description of the

implementation in Webots. Even though the controller was able to stabilize the nonlinear model it was not able to stabilize the virtual **AAU-BOT1** in Webots.

Control Strategy B: Control Strategy B utilize the controllers in the amplifiers and Webots, having a separate PID controller for each joint. In addition, joint # 16 and 17, was controlled by a LQG controller. This controller was capable of controlling both Webots and **AAU-BOT1**, thus enabling the use of the generated trajectories. Unfortunately this approach did not allow control of the double actuated joints. This lead to removing one of the motors on the double actuated joints, meaning that they did not have sufficient power to be controlled.

ZMP controller: To ensure that **AAU-BOT1** is capable of maintaining its balance, a ZMP controller was implemented. The ZMP controller utilizes that the Torso can move in both the x and y direction by using joint #16 and 17. The resulting controller was capable of moving the ZMP, thus making Control Strategy B capable of walking through its first step in Webots. Due to the fact that one of the FTS amplifiers broke, it has not been possible to test this controller on **AAU-BOT1**.

9.1.6 Status of the AAU-BOT1 Project

The **AAU-BOT1** project is the first humanoid robot project where a humanoid robot of this size has been entirely designed by students. The overall plan of the **AAU-BOT1** research project is that [Pedersen et al., 2007] design the mechanical part of **AAU-BOT1** in 2006/2007, in September 2007 it should be handed over to the Control department. **AAU-BOT1** should be in the Control department until year 2010, where it is handed over to the department of Health Science and Technology . When it is handed over to the department of Health Science and Technology it should be able to perform human-like gait. The scope for this project in 2007/2008 was to establish a platform where control of **AAU-BOT1** can be implemented and develop the control part in such extent that it is possible to walk static gait with the physical **AAU-BOT1** and walk dynamic gait in simulation.

Development and installation of hardware and networks on **AAU-BOT1** was delayed since **AAU-BOT1** was handed over partially assembled the 30^{th} of November, i.e. some of the belts that drive the joints were still missing. The delay was a setback, since some of the safety circuits, that justifies operation with **AAU-BOT1** was first mounted the 14^{th} of May. This resulted in short time periode to perform test with the physical **AAU-BOT1**, due to the limited project time. Even though the hardware has been challenging and time consuming, the groups effort has resulted in a flexible platform. The platform gives the possibilities to control by using current mode, velocity mode or position mode. **AAU-BOT1** has been developed in such extend that it follows a static gait trajectory when it is hanging in the air. However due to a problem with double actuated joints, i.e. only one motor can be used in the double actuated joint, **AAU-BOT1**cannot walk on the floor. This problem has to be solved by further groups on the **AAU-BOT1** project. Since **AAU-BOT1** followed the trajectory for the first time the 1^{st} of June, it has not been possible to make any adjustment to optimize the performance, due to limited time of this thesis.

9.2 Conclusion

Humanoid robots are one of the hardest problems in the area of control, due to the complex mechanical motion of humans. The aim of controlling humanoid robots vary, some are made to show purposes, others are made to relieve humans by carrying out their work, but common for all humanoid robots are that they must be able to move around in human environment, e.g. walk on steps or walk in terrain. Another area where humanoid robots are expected to be used is in the area of health care, where the robots can be used to help humans who are physically disabled, to rehabilitate their physically movement. The third humanoid robot at Aalborg University, **AAU-BOT1**, is made for that purpose.

AAU-BOT1 is the first humanoid robot in Denmark of its size. It has 19 degrees of freedom, where 17 of these are actuated, weighs 68 kg and it is 180 cm tall. It is also the first humanoid robot of its size, that has been entirely developed by students. The main focus for this thesis is to develop and implement an instrumentation strategy and network design for **AAU-BOT1**, modeling it, and develop control strategies, such that it is possible to obtain static gait with **AAU-BOT1** and enable it to obtain dynamic gait in simulation.

One of the key points of the master's thesis was to find a solution to the instrumentation. The solution require implementation of a system with high performance, a high number of sensors and actuators and a requirement of low weight. The result ended up replacing the chosen amplifiers for the DC motors, with CAN enabled DC motor amplifiers (EPOS), and implementing Force Torque Sensor (FTS) amplifiers that have a RS485 interface. Both types of amplifiers were implemented with success. By implementing the EPOS amplifiers it was possible to control the motors with high precision and get readings of relative position, absolute position, velocity and current. Furthermore the EPOS amplifiers enables use of current control, velocity control and position control, which can be used directly to control **AAU-BOT1**. The FTS amplifier has excellent performance comparing with other available amplifiers on the market, i.e. high resolution and high sample rate, which is necessary to determine the stability during walk of **AAU-BOT1**. Unfortunately one of the EPOS short circuited, which caused one of FTS amplifier to break due to a common power supply. Even though it has been possible to obtain a maximal RMS error of 2.4% on the remaining FTS, the stability cannot be calculated without a new FTS amplifier.

To enable the on-board computer to retrieve data from the FTS amplifiers and to send and receive data from the 23 EPOS amplifiers, drivers for each sensor and actuator were developed. The result of this is an interface and a software architecture that is module based and simple to use, which will be easy to maintain for future groups working on **AAU-BOT1**. It has not been possible to test the total real time performance of the entire software since one of the FTS amplifiers were disabled, however by measuring the throughput it was found that 86% of the frames were send and received at a sampling rate of 250 Hz on the 5 CAN busses.

In order to design a controller, several models have been developed, describing the different parts of **AAU-BOT1**. Firstly, the DC motor model was developed to give an accurate view of the actuators. To verify it, the motor on the left arm is parameter estimated, yielding that the model has a mean squared error of 12.1%. Secondly, The kinematic model and inverse model were developed to determine the position and angles

of **AAU-BOT1** by using transformation matrices and mechanical data of **AAU-BOT1**. Lastly, the dynamic model was developed by applying the Lagrange method on the data from **AAU-BOT1** in each single stance phase and including the kinematic model and the DC motor model. This resulted in a hybrid state space model with 38 states, 17 inputs and 34 outputs. Unfortunately it has not been possible to verify the dynamic model, due to its immense size combined with the relative short project period.

To be able to walk dynamically and statically, trajectories were developed using an off-line trajectory generator. Based on stability, energy consumption and human parameters a method was developed to generate the best trajectory for static and dynamic gait. However since the dynamic model was not complete it was not possible obtain a ZMP that was accurate in an extent that could be used for trajectory generation. A static gait trajectory was developed, using the inverse kinematic model. This resulted in a maximum stability margin of 4.75 cm to the edge of the support area.

Two control strategies have been proposed. The first control strategy uses an LQG controller as a posture controller. The controller stabilized the nonlinear model, but is not found feasible to use on the actual robot, before e.g feedback linearization has been implemented and a more accurate model has been obtained.

The second control strategy utilizes PID controllers on all the joints except the ones controlling the waist pitch and roll, which uses a LQG controller. Furthermore the strategy proposes a ZMP controller which maintains stability while walking. The ZMP controller worked as intended and The second control strategy has been tested successfully up against the virtual robot in the Webots environment.

The performance of the second control strategy on the actual system are yet unknown as **AAU-BOT1** is still not able to measure the ZMP due to a faulty FTS amplifier. The preliminary results show that the actual **AAU-BOT1** can track the trajectories, but cannot maintain balance without the ZMP controller.

To summarize, a platform for **AAU-BOT1** has been developed. It consists of a complete instrumentation strategy, containing both EPOS power amplifiers and FTS amplifiers together with the belonging network strategy. Furthermore a software architecture was implemented on the on-board computer, enabling communication between the controllers and the transducers. A model has been developed and used to derive trajectories which are stable and suitable for the robot. It is therefore concluded that this master's thesis gives a solution to the implementation, modeling and control of the **AAU-BOT1** and thereby ending this second step of the **AAU-BOT1** project successfully.

9.3 Future Work

During this master's thesis it has been possible to build a platform that is able to control **AAU-BOT1** in simulation and on the physical robot. The platform has been build such that it is possible to implement a more sophisticated control of **AAU-BOT1**, in order to obtain dynamic gait on the physical robot. This section describes what the authors regard as the next step in the **AAU-BOT1** project.

The IMU has not been fully implemented yet, i.e. driver software need to be devel-

oped, such that is it possible to obtain a precise rotational orientation. The IMU has been purchased, but has not been implemented. The S-function and the shared memory server are however prepared for its inclusion.

A new FTS amplifier has been ordered, such that it is possible to replace it with the faulty one. It is recommended that all amplifier gets their firmware updated since the software protocol contains a design bug from the manufacturer. This bug results in a time consuming survey of all received bytes in order to retrieve data from the amplifiers. When the new FTS amplifier is received a new calibration is needed of both FTS because the designed test rig has to be redesigned.

Currently, potentiometers are utilized to obtain the absolute position of each joint, however these seems to drift over time. This means that the system has to be recalibrated often. It is recommended that the potentiometers are replaced by hall effect sensors to prevent this.

To increase the control of the flow of information on the CAN bus, the actuator S-functions and the sensor S-function is recommended to be combined into one S-function. This will also make the software easier to maintain.

AAU-BOT1 is still not fully equipped with the required safety measures. This includes emergency stop devices and protection against short circuit. The FTS amplifiers requires a noise decoupled power supply to operate properly. It is recommended to separate the EPOS's in groups, such that each group has its own protection.

To obtain an accurate dynamic model knowledge about friction in each joint and the precise weight distribution must be obtained. The friction can be obtained by performing the proposed parameter estimation method for all joints. The weight distribution can be obtained by updating the SolidWorks model and extracting the parameters. Furthermore all joint movement constraints from the physical system must be implemented in the inverse kinematics. This should be done since the physical constraints are shifted with the implementation of the potentiometers. With these constraints the trajectories can be improved for the actual robot. Furthermore, the heel impact is neglected and the toe off phase is not fully implemented in the models of this master's thesis and should be among the next subjects examined in the **AAU-BOT1** project.

Bibliography

Palle Andersen and Tom S. Pedersen. Modeldannelse. Lecture note about modelling of systems (in danish), used in the 6th semester at AAU, `http://www.control.aau.dk/~pa/kurser/PR6model/modelnote.pdf`, february 2007.

Takemasa Arakawa and Toshio Fukuda. Natural motion trajectory generation of biped locomotion robot using genetic algorithm through energy optimization. *IEEE 1996*, pages 1495–1500, 1996. doi: None.

ASLSOFT. *ASL "standard" software (active).* Autonomous System Lab (ASL), which is part of the Swiss Federal Institute of Technology in Zürich, 408th edition, 2008. `https://lsa1pc12.ethz.ch/`.

Yariv Bachar. Developing controllers for biped humanoid locomotion. Technical report, The University of Edinburgh, 2004.

Thomas Bak and Roozbeh Izadi-Zamanabadi. Lecture notes - Hybrid systems at Aalborg University. Lecture note for the Hybrid systems course at AAU, IAS, 2004.

Ozkan Bebek and Kemalettin Erbatur. Adaptive fuzzy system for tuning biped robot gate parameter. *Fira Robot World Congress 2003*, 3, Oct. 2003. doi: None.

Morten Bisgaard. Instrumentation and Data network at Autonomous Systems. Lecture in instrumentation, wirering and finding the right computer system for Autonomous Systems., November 2007a.

Morten Bisgaard. *Modeling, Estimation and Control of Helicopter Slung Load System.* PhD thesis, Aalborg University, 2007b.

Youngjin Choi, Bum-Jae You, and Sang-Rok Oh. On the stability of indirect zmp controller for biped robot systems. *Proceedings of 2004 IEEORSl International Conference on intelligent Robots and Systems*, pages 1966–1971, Sep. 2004. doi: None.

Jens Christensen, Jesper Lundgaard Nielsen, Mads Sølver Svendsen, Mikael Svenstrup, Kasper Winther, and Peter Falkesgaard Ørts. Modelling and control of a biped robot. Technical report, Aalborg University, 2006.

Jens Christensen, Jesper Lundgaard Nielsen, Mads Sølver Svendsen, and Peter Falkesgaard Ørts. Development, Modeling And Control of A Humanoid Robot. Master's thesis, Aalborg University, 2007.

John J. Craig. *Introduction to Robotics: Mechanics and Control.* Pearson Prentice Hall, 3rd edition, 2005. ISBN: 0-13-123629-6.

Cyberbotics Ltd. *Webots Reference Manual*, 5.9.0 edition, April 2008.

Jens Dalsgaard. Using CAN in control systems. Meeting about using CAN on **AAU-BOT1** and finding the right number of CAN network when using a sampling rate at 1000 Hz., November 2007.

D. Djoudi, C. Chevallereau, and Y. Aoustin. Optimal reference motions for walking of a biped robot. *Proceedings of the 2005 IEEE Conference on Robotics and Automation*, pages 2002–2007, April 2005. doi: None.

L. Dozio and P. Mantegazza. Real time distributed control systems using rtai. *Object-Oriented Real-Time Distributed Computing, 2003. Sixth IEEE International Symposium on*, 1:11–18, May 2003. doi: 10.1109/ISORC.2003.1199229.

Ken Dutton, Steve Thompson, and Bill Barraclough. *The art of control engineering*, 1997.

R.G.J. Flay and I.J. Vuletich. Development of a wind tunnel test facility for yacht aerodynamic studies. *Journal of Wind Engineering and Industrial Aerodynamics*, 58: 231–258, July 1995. doi: None.

Gene F. Franklin, J. David Powell, and Michael Workman. *Digital Control of Dynamic Systems*. Addison-Wesley, 3rd edition, 1997.

Emilio Frazzoli. Talk about online trajectory generation on MIT's Darpa car. Study trip to Massachusetts Institute of Technology, where Associate Professor (of Aeronautics and Astronautics) Emilio Frazzoli demonstrated how the generated trajectories during Darpa challenges., Februar 2008.

Adolfo Garcia. *SINGLE-SUPPLY AMPLIFIERS*, 2000.

Andrew A. Goldenberg, B. Benhabib, and Robert G. Fenton. A complete generalized solution to the iverse kinematics of robots. *IEEE Journal of Robotics and Automation*, RA-1(1):14–20, March 1985.

Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering, Theory and Practice Using MATLAB*. Wiley-Interscience Publication, 2nd edition, 2001. doi: 10.1002/0471266388.ref, ISBN: 9780471392545.

Jørgen Haffgaard. *Lecture Notes in Data Aqusition at Engineer Colleges Aarhus*, 2005.

Jan Helbo. Homepage for Jan Helbo. web, 2008. url: `http://www.control.aau.dk/~jan/`.

Qiang Huang, Kenji Kaneko, Kazuhito Yokoi, Shuuji Kajita, Tetsuo Kotoku, Noriho Koyachi, Hirohiko Arai, Nobuaki Imamura, Kiyoshi Komoriya, and Kazuo Tanie. Balance control of a biped robot combining off-line pattern with real-time modification. *Proceedings of the 2000 IEEE International Conference on Robotics and Automation San Francisco*, pages 3346–3352, April 2000. doi: None.

Qiang Huang, Kazuhito Yokoi, Shuuji Kajita, Kenji Kaneko, Hirohiko Arai, Noriho Koyachi, and Kazuo Tanie. Planning walking patterns for a biped robot. *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, 17(3):874–879, June 2001. doi: None.

Shuuji Kajita, Fumoi Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kazuhito Yokoi, and Hirohisa Hirukuwa. Biped walking pattern generation by a simple three-dimensional inverted pendulum model. *Advanced Robotics, VSP and Robotics Society of Japan*, 17 (2):131–147, May 2003. doi: None.

Charles Kitchin, Lew Counts, and Moshe Gerstenhaber. *Reducing RFI Rectification Errors in In-Amp Circuits*, 2003.

Morten Knudsen. *Experimental modelling of dynamic systems*. Afd. for Proceskontrol, AAU, 0.2 edition, January 2004.

Konstantin Kondak and Gunter Hommel. Control and online computation of stable movement for biped robots. *Intl. Conference on Intelligent Robots and Systems. Las Vegas, Nevaaa*, pages 874–879, Oct. 2003a. doi: None.

Konstantin Kondak and Günter Hommel. Control and online computatopm pof stable movement for biped rogbots. *Intelligent Robots and Systems, 2003 IEEE/RSJ International Conference on*, pages 874–879, Oct. 2003b. doi:.

David C. Lay. *Linear Algebra*. Greg Tobin, 3rd edition, 2003.

LOLITech. *CanFestival*, 2007. `http://www.canfestival.org/`.

Lorenz Messtechnik. *RS485 USB Protokoll und Datenblocks*, 2001.

Klaus Löffler, Michael Gienger, and Friedrich Pfeiffer. Sensors and control concept of a biped robot. *Industrial Electronics, IEEE Transactions on*, 51(5):972–980, Oct. 2004a. ISSN 0278-0046. doi: 10.1109/TIE.2004.834948.

Klaus Löffler, Michael Gienger, Friedrich Pfeiffer, Fellow, IEEE, and Heinz Ulbrich. Sensors and control concept of a biped robot. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 51(5):972–980, Oct. 2004b. doi: None.

Maxon Motors. *EPOS Positioning Controller Firmware Specification*. maxon motor ag, Brünigstrasse 220 P.O. Box 263 CH-6072 Sachseln, May 2007a. Maxon document #798675-01.

Maxon Motors. *EPOS Application Note: Position Regulation with Feed Forward*. maxon motor ag, Brünigstrasse 220 P.O. Box 263 CH-6072 Sachseln, 06 edition, May 2007b.

Maxon Motors. *RE ø40mm, Graphite Brushes, 150 Watt*, 2007c.

Maxon Motors. *RE ø30mm, Graphite Brushes, 60 Watt*, 2007d.

Maxon Motors. *RE ø35mm, Graphite Brushes, 90 Watt*, 2007e.

Tad McGeer. Passive walking with knees. *1990 IEEE*, pages 1640–1645, 1990. doi: None.

Alan S. Morris. *Measurement & Instrumentation Principles*. Elsevier Butterworth Heineman, 3rd edition, 2005. doi: 10.1088/0957-0233/12/10/702, ISBN: 0-7506-5081-8.

Masaki Ogino1, Koh Hosoda2, and Minoru Asada3. Learning energy efficient walking based on ballistics. *IEEE 1996*, 17(3):1495–1500, June 2001. doi: None.

Yu Ogura, Kazushi Shimomura, Hideki Kondo, Akitoshi Morishima, Tatsu Okubo, Shimpei Momoki, Hun ok Lim, and Atsuo Takanishi. Human-like walking with knee stretched, heel-contact and toe-off motion by a humanoid robot. *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3976–3981, Oct. 2006. doi: 10.1109/IROS.2006.281834.

Jong H. Park and Kyoung D. Kim. Biped robot walking using gravity-compensated inverted pendulum mode and computed torque control. *International Conference on Robotics and Automation in Leuven, Belgium*, May 1998. doi: None.

Mikkel Melters Pedersen, Allan Agerbo Nielsen, and Lars Fuglsang Christiansen. Design of Biped Robot AAUBOT1. Master's thesis, Aalborg University, 2007.

Peter Rydesäter. *TCP UDP IP toolbox for* MATLAB™, 2003.

Jean-Claude Samin. *Mechanics of Multibody Systems*, preliminary edition, January 2005.

Raymond Serway, Robert Beichner, and John Jewett. *Physics for Scientists and Engineers*. 5th edition, 2000.

Russell Smith. *Open Dynamics Engine, V0.5 User Guide*, February 2006.

Ole Sørensen. *Optimal Control*, preliminary edition, February 2007.

Jakob Stoustrup. Data systems at Autonomous Systems. Meeting about finding the right computer system for Autonomous Systems., November 2007.

Masaki Takahashi, Terumasa Narukawa, Ken Miyakawa, and Kazuo Yoshida. Combined control of cpg and torso attitude control for biped locomotion. *Intelligent Robots and Systems, 2005*, 2005. doi: None.

Russ Tedrake. Biped robots energy consumption during human locomotion. Study trip to Massachusetts Institute of Technology, where the master thesis group visit CSAIL department and followed Professor Russ Tedrake one day, February 2008.

Dimitri van Heesch. *Doxygen*, May 2008. `www.doxygen.org`, Doxygen is a tool for automatic creation of documentation of software.

Christopher L. Vaughan, Brian L. Davis, and Jeremy C. O'Connor. *DYNAMICS OF HUMAN GAIT*. Kiboho Publishers, 2nd edition, 1992.

Richard Voyles, James Morrow, and Pradeep Khosla. The shape from motion approach to rapid and precise force/torque sensor calibration. *Journal of Dynamic Systems, Measurement and Control*, 119(2):229–235, June 1997.

M. Vukobratović, B. Borovac, and V. Potkonjak. Contribution to the synthesis of biped gait. In *Proceedings of the IFAC Symposium on Technical and Biological Problem and Control*, 1969.

M. Vukobratović, B. Borovac, and V. Potkonjak. Towards a unified understanding of basic notions and terms in humanoid robotics. *Robotica*, 25(1):87–101, 2007. ISSN 0263-5747. doi: 10.1017/S0263574706003031.

Wolfram MathWorld. Cubic spline, 2004. URL `http://mathworld.wolfram.com/CubicSpline.html`. The web's most extensive mathematics resource.

WolframMathWorld. Point to line equaltion, 2004. URL http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html. The web's most extensive mathematics resource.

Dirk Wollherr. *Design and Control Aspects of Humanoid Walking Robots*. PhD thesis, Technical University Munich, 2005.

Dirk Wollherr, Martin Buss, Michael Hardt, and Oskar von Stryk. Research and development towards an autonomous biped walking robot. *International Conference on Advanced Intelligent Mechatronics*, pages 968–973, 2003. doi: None.

Xiuping Mu; Qiong Wu. A complete dynamic model of five-link bipedal walking. *American Control Conference, 2003. Proceedings of the 2003*, 6:4926–4931, June 2003. ISSN 0743-1619. doi: 10.1109/ACC.2003.1242503.

Xubuntu. http://www.xubuntu.org, 2008. URL http://www.xubuntu.org. Xubuntu is an entirely open source operating system, which uses the Xfce desktop environment on top of a Ubuntu GNU/Linux core.

Appendix A

Verification of Models

In this appendix, the models that are designed in Chapter 5 will be verified. Firstly, the DC Motor model is parameter estimated using the MATLAB^{TM} toolbox SENSTOOLS. Then the kinematic model is verified visually and using Webots, and lastly the inverse kinematic model is verified using the kinematic model.

A.1 Verification and Parameter Estimation of DC Motor Model

To verify the DC Motor model seen in Section 5.3, the parameters of the model are parameter estimated. Due to the fact that the amplifiers allow direct control of the current (i), the model is simplified to the following:

$$\ddot{\theta}J = \underbrace{K_t i}_{\tau_M} \underbrace{-\mu\dot{\theta}}_{\tau_\mu} - \tau_c \underbrace{-\frac{\sin(\frac{\theta}{G})mlg}{G}}_{\tau_L} \tag{A.1}$$

where:

J	is the inertia of the arm
K_t	is the motor constant
μ	is the viscous friction coefficient
m	is the mass of the arm
l	is the distance from the joint to the center of mass of the arm
g	is the gravity constant
G	is the gear ratio
θ	is the angle of the rotor of the DC motor
τ_M	is the torque exerted by the DC motor
τ_μ	is the torque exerted by the viscous friction
τ_c	is the torque exerted by the coulomb friction
τ_L	is the torque exerted by the load

The model is discretizised and implemented nonlinearly in SENSTOOLS [Knudsen, 2004], which is a parameter estimation toolbox for MATLAB^{TM}.

A.1.1 Method

The DC motor model is tested by subjecting the left arm of **AAU-BOT1** to an escalating pulse train current command (u_i, see Figure A.2). The measured current i is used as input to the nonlinear model seen in Equation (A.1). The results from the test is implemented

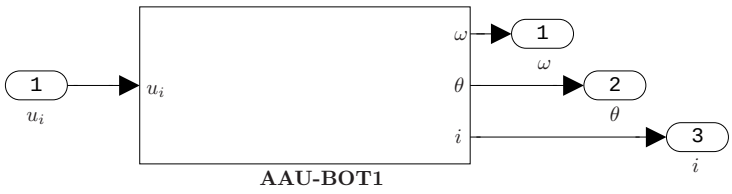


Figure A.1: Test setup for the parameter estimation.

in SENSTOOLS, with the following known parameters:

- The gearing G for the arm is found in Table 5.1 to be 111
- The motor constant K_t is $53.8 \cdot 10^{-3}[\frac{\text{Nm}}{\text{A}}]$

The parameters that are to be estimated has the following starting values:

- Inertia of the arm (from Table C.2 on page 185) :
  $J = \frac{0.0419}{111} = 377.48 \cdot 10^{-6} \frac{\text{kg}}{\text{cm}^2}$
- Coulomb friction is estimated by examining how much torque is exerted at the point where the motor starts moving:
  $\tau_c \approx i_{\text{motor starts}} \cdot K_t \approx 0.75 \cdot 0.0538 = 40.3 \cdot 10^{-3}$
- The load constant $Tl = |b_{la}| \cdot m \cdot \frac{g}{G}$ is estimated to
  $Tl = |b_{la}| \cdot m \cdot \frac{g}{G} = 0.3503 \cdot 0.9 \cdot \frac{9.82}{111} = 0.0279$
- The starting value of the viscous friction μ is chosen to
  $\mu = 0.0002$

A.1.2 Result

The results can be seen in Figure A.2 and A.3. The measured angle is the angle of the motor. By using SENSTOOLS, the parameters are found to be the following, yielding a mean squared error of 12.1% (see Figure A.3 on the next page):

- $J = 12.15 \cdot 10^{-6} \frac{\text{kg}}{\text{cm}^2}$
- $\tau_c = 48.76 \cdot 10^{-3}\text{Nm}$
- $Tl = 21.40 \cdot 10^{-3} \frac{\text{Nm}}{\text{m}}$
- $\mu = 52.42 \cdot 10^{-6} \frac{\text{Nm s}}{\text{rad}}$

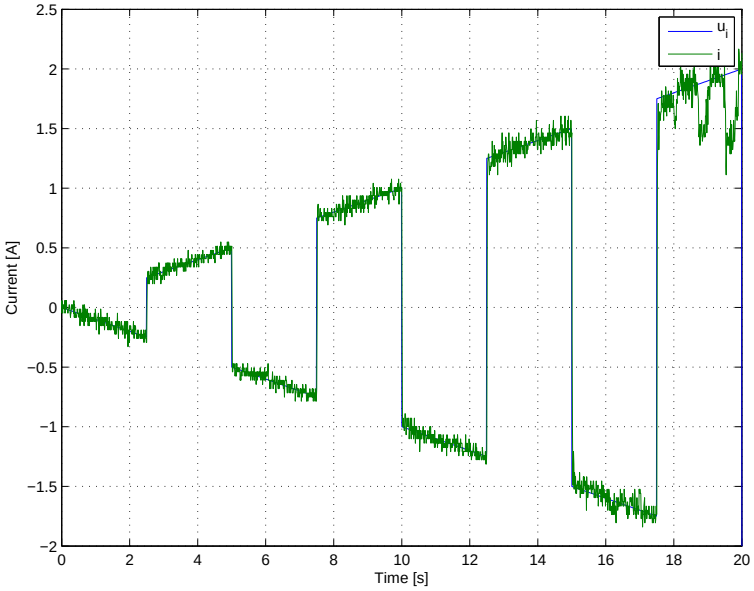


Figure A.2: Input for parameter estimation.

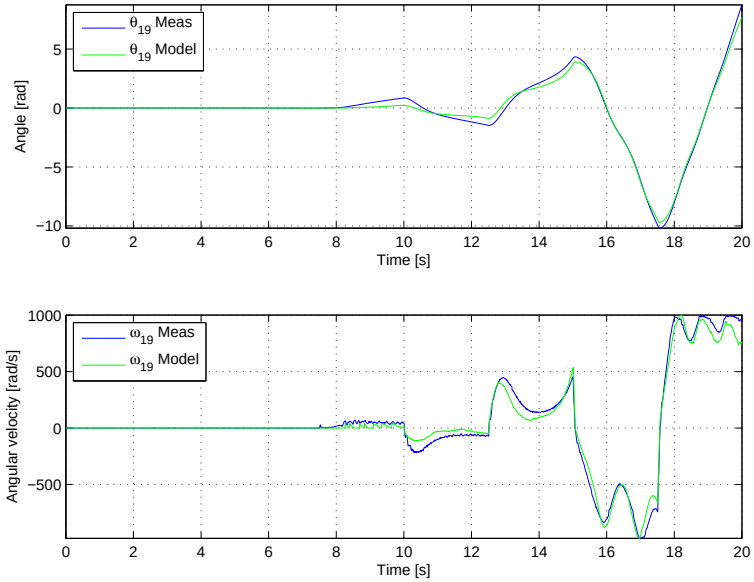


Figure A.3: Output of parameter estimation test. The model has a squared mean error of 12.1%.

A.1.3 Discussion and Conclusion

By parameter estimating the model, a mean squared error of 12.1% is achieved. This error is considered low due to the high value of the gearing, the high degree coulomb friction and potential offset of the angle. Another thing to consider is the high degree of mutual cross coupling of the measurements. According to [Knudsen, 2004], a normed squared error of only 5-8% should be obtainable. The found parameters are perceived as correct, and will be used in the model. For the same reasons the DC Motor Model is considered verified as being correct.

A.2 Verification of Kinematic Model

The kinematics are used to derive the positions of individual links and their CoM's from the joint angles. This is done by a transformation from joint space to Cartesian space. To do this transformation different kinematic chains has been developed as described in Section 5.4. To verify the kinematics the different kinematic chains are verified towards the Webots representation of the **AAU-BOT1**. Webots is a robots simulation tool which contains its own 3D virtual world and is described in Section 4.8. The **AAU-BOT1** has been constructed in this virtual environment such that the physical proportion of the robots match the parameters used for the kinematic model.

A.2.1 Method

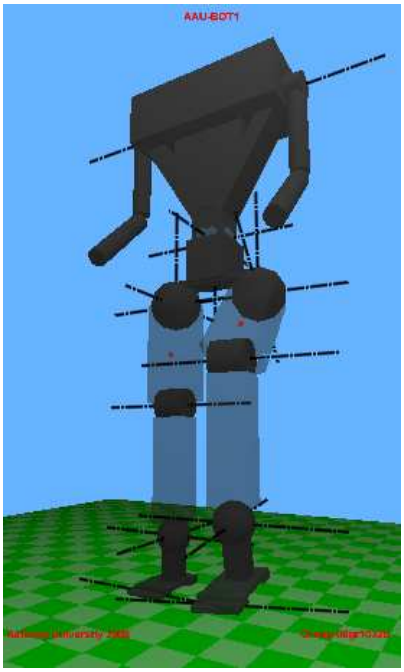
The outputs from the kinematic chains are described in two phases, SSP-R and SSP-L. The two phases will be verified for some given joint angles as input. This is done by utilizing the developed Webots representation of **AAU-BOT1** together with the programmed functions used to interface Webots from Simulink. The following commands are used: `set_theta_joints(angles)` which sets the joint angles of the **AAU-BOT1** in Webots from Simulink. When the joints of the **AAU-BOT1** are set to the correct angle the commands `get_Pj_webots()` are used to retrieve the absolute Cartesian position of all the links where after they are transmitted back to Simulink.

The retrieved positions of the links from Webots are compared to the link positions generated from the kinematic model in Simulink.

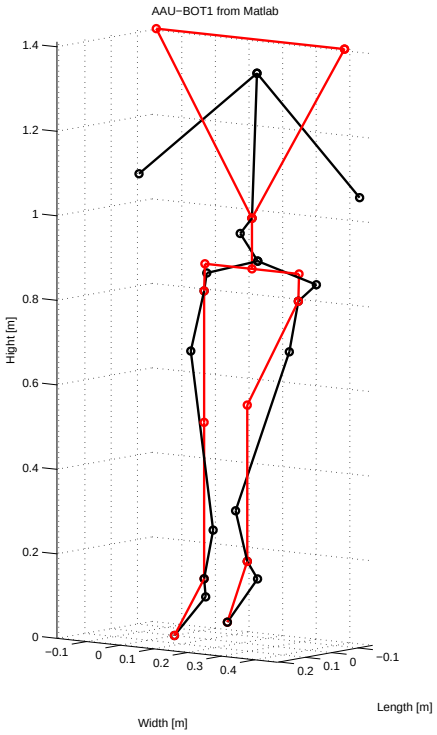
As this test is performed to verify the kinematic, the dynamics in the Webots world has to be disabled. Unfortunately there is no direct option to disable the physics engine such that the Webots model can be considered ideal. To accommodate this problem the gravity is set to a small value and the supporting toe is set to a relatively high value compared to the weight of the rest of the robot. This value can not be set to high since this will cause the robot in the Webots world to penetrate and break through the virtual floor, the weight is set to 100 kg since this is greater than 68.47 kg which is the weight of **AAU-BOT1** used for the simulation. Lastly the supporting toe of the robot in the Webots world is made wide and long this will ensure a stable system and this is needed to compare the kinematic in Simulink with Webots.

A.2.2 Results

The verification is done in two steps. First in SSP-R and thereafter in SSP-L. The joint angles used to test with are listed in Table A.1: With these test angles as input to the Webots model and the kinematic model the models are actuated and the visual result for SSP-R can be seen in Figure A.4



(a) Webot representation of AAU-BOT1 in SSP-R.



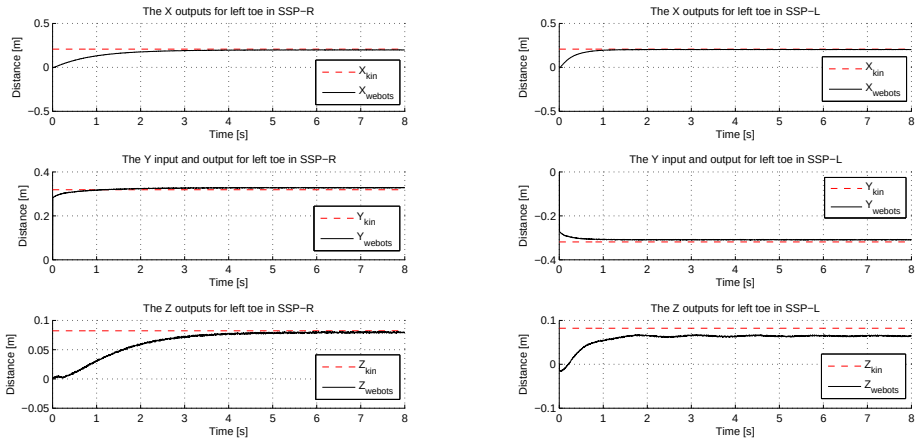
(b) Matlab plot of AAU-BOT1 in SSP-R. The red dots represent the position of joints and the black dots are the CoM.

Figure A.4: Visual result by applying the test angles for SSP-R.

Table A.1: Test angles for the two phases.

SSP-R	SSP-L
$\theta_1 \cdots \theta_7$=0°	θ_1=0°
θ_8=5°	θ_2=5°
θ_9=-45°	θ_3=-5°
θ_{10}=0°	θ_4=-45°
θ_{11}=45°	θ_5=0°
θ_{12}=5°	θ_6=45°
θ_{13}=5°	θ_7=5°
$\theta_{14} \cdots \theta_{19}$=0°	$\theta_8 \cdots \theta_{19}$=0°

From visual inspection the two models seem similar and to verify the similarity the position of the last joint in the chain is compared to each other. The last joint is the left toe when in SSP-R and the right toe when in SSP-L. The result of this comparison can be seen in Figure A.5-(a) and in Figure A.5-(b). Here the coordinates for feet in both phases are listed and the red dashed line is the output from the kinematics in Simulink and the black solid line is the output from the Webots model.



(a) Right phase.

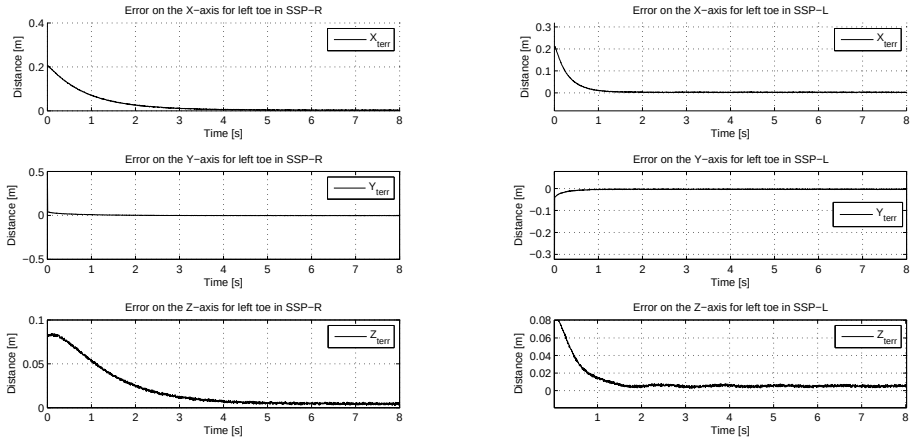
(b) Left phase.

Figure A.5: Graph showing the output position for toe joint for kinematic model and for Webot model.

To show the difference between the two models the error can be seen in Figure A.6.

From both figures it is obvious that the Webots model contains dynamics and the kinematic contains none. This can be seen since the Webots model propagates towards the kinematic model.

The largest error can be seen in the z-coordinate both for SSP-R and SSP-L. When the Webots model is settled the error is roughly 0.004 m. This is not necessarily an error in the kinematic model, but more likely a steady state error in the position controller used to control the servo motor controller in Webots. The proportional gain in Webots is by default set to 10 and this value is also suitable for this verification even though it is changeable.



(a) Right phase.

(b) Left phase.

Figure A.6: Graph showing the output error between the kinematic model and the Webot model.

A.2.3 Discussion and Conclusion

The error between the toe joints from the Webots model and the kinematic model are at first relatively large, but when the dynamics in Webots is propagated the error comes close to zero in both phases. The small deviation of 0.004 m are considered to be caused by the dynamics in Webots and not by an faulty kinematics. The kinematic is now considered successfully verified towards the virtual world in Webots. This does not verify the actual **AAU-BOT1** as the hardware mounting results in new parameters and may lead to inaccuracy between the model and the system.

A.3 Verification of Inverse Kinematic Model

During this appendix the inverse kinematic is verified. The inputs to the inverse kinematic model are the $\vec{P}_{t_r}, \vec{O}_{t_r}, \vec{P}_{t_l}, \vec{O}_{t_l}$ The first input is the position of torso joint 17 calculated from right foot, consult Figure 5.9 to see the geometry. The following input are the orientation of torso given by the right leg and the last two inputs are the position and orientation of torso given by the left leg. The output of the inverse kinematic are the angles in joint space. This means that the kinematic model gives a coordinate transformation from the generalized coordinates of the end of each limb and torso, to joint space. It has been chosen to verify the inverse kinematic model toward the kinematic model, as the kinematic model do the opposite transformation, from joint space to generalized coordinates. This means that the input to the inverse kinematic model and the output from the kinematic model can be compared and should be equal, under normal circumstances. The test set up can be seen in Figure A.7 and is setup the same way in simulink.

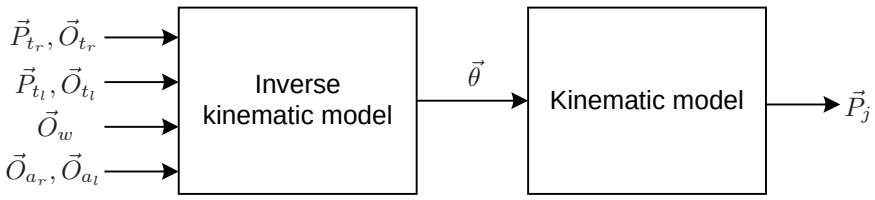
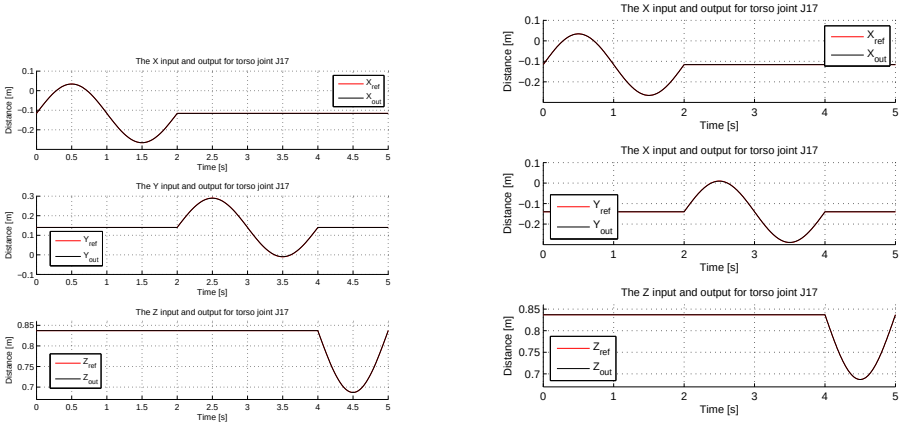


Figure A.7: Block diagram of input and output for the test setup.

A.3.1 Translatory Displacement of Torso

The first test is developed to verify the translatory motion of the torso joint J_{17} by actuating $\vec{P}_{t_r}, \vec{P}_{t_l}$ which is the position of torso given for the right and left phases. In Figure A.8(a) x,y and z components of the trajectory for $\vec{P}_{t_r}$ can be seen and for left phase$\vec{P}_{t_l}$ can be seen in Figure A.8(b).



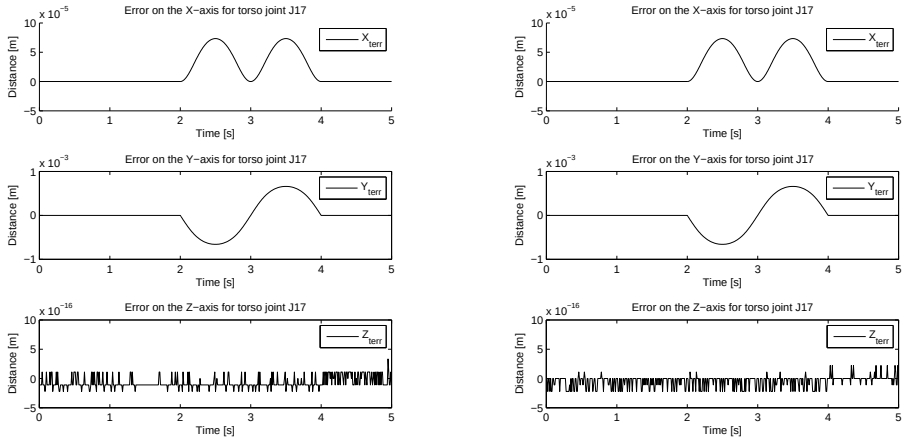
(a) Right phase.

(b) Left phase.

Figure A.8: Graph showing the input for the inverse kinematic model and output for the inverse kinematic model.

As is can be seen in Figure A.8 the torso is first moved translatory in a sine wave motion in x-direction and next in the y-direction and lastly in the z-direction. Since the **AAU-BOT1** is standing upright from start, only the negative part of the sinewave is used. The span of the test is 5 sec. The output from the kinematic model is also shown in the Figure A.8 but it is hard to see the difference between the input and output so in order to see the difference the input is subtracted from the output and the error can be viewed. The error between the reference-input and the output can be seen for the right phase in Figure A.9(a) and for the left phase in Figure A.9(b).

In Figure A.9 is can be seen that the errors for the right and left phase are the same, this is because the left a right phase make use of substantially the same inverse kinematic model, the only difference is the negative signs applied in the left side model so that is



(a) Right phase.

(b) Left phase.

Figure A.9: Graph showing the error between the input to inverse kinematics and output from the kinematic model.

it behaves the opposite way of the right phase.

Furthermore the errors between the input to the inverse kinematic model and the output from the kinematic model are small. The max error on the x-direction is in scope of 10^{-5} and the y-direction is roughly 0.6 mm and the z-direction is in the scope of 10^{-16}. The errors in the x, y and z-directions are substantially small and can be regarded as zero. Since the backlash and precision on joint angle measurement on the physical system, will give larger error than the one seen in this test. This is because it is chosen to calibrate the absolute position with an analog potentiometer which is potentially noisy and it will be hard to even detect the errors shown in this test.

A.3.2 Torso Rotation

This test is conducted to verify the rotation of the torso caused by the hips. In order to test this a test trajectory has be proposed. This trajectory includes rotation around the x-,y- and z-axis as seen in Figure A.10 on the following page. Here the red are rotation around x-axis and the green are rotation around y-axis and the last rotation is around the z-axis.

The test is for rotations in the hip, but the verification is done from the torso as a rotation in the hip will rotate the entire torso. So when the hip is given a rotation it will result in a similar rotation on the generalized coordinate for the torso P_{t_r},P_{t_l}. This extra rotation is applied to the torso coordinate and it is used as the reference input to this test and the output is from the kinematic model. The reference input and output for the right phase can be seen in Figure A.11(a) and the left phase can be seen in Figure A.11(b).

The difference between the input and output are hard to distinguish between since they are very small. In order to see the performance better, the signals are subtracted and the errors is displayed. The errors for the right phase is seen in Figure A.12(a) and the left phase can be seen in Figure A.12(b)

The errors shown in Figure A.12 are equal, which is because the inverse kinematic

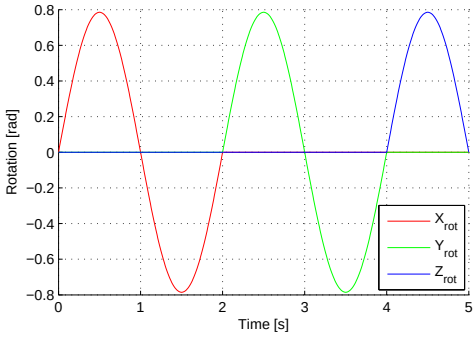
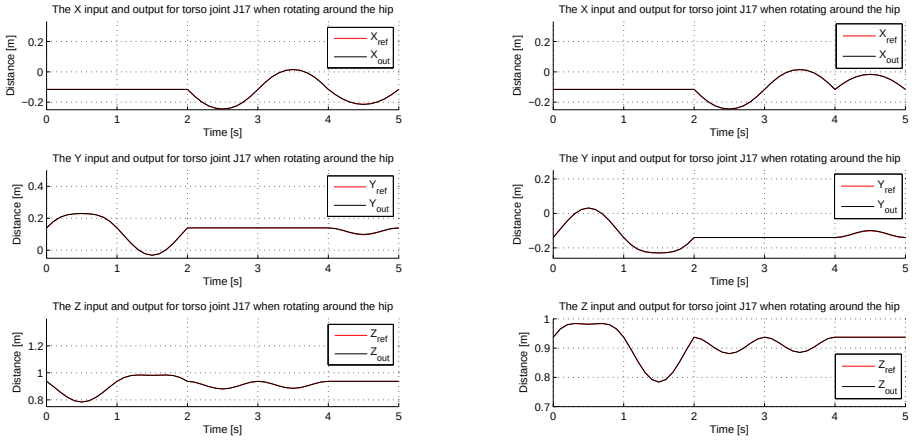


Figure A.10: Graph showing the test rotation around the three axis.

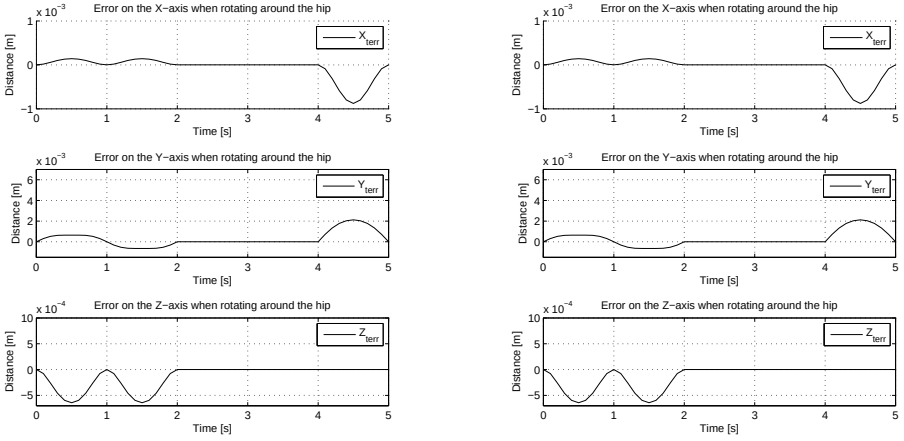


(a) Right phase.

(b) Left phase.

Figure A.11: Graphs showing the input for the inverse kinematic model and output for the inverse kinematic model when rotating around the hip.

model for the left and right phase only differs with negative signs. The errors on the X, Y and Z component are small but a little larger than in the first test. The largest error is 2 mm and can be seen on the Y component and is considered very small on this size system. The largest errors on the X and Z component are roughly -0.8 mm and -0.06 mm which are considered small. It is obvious that these small errors are caused by the rotation in pelvis as the errors have a smooth wave behavior just as the test trajectory. This also means that it is possible to reduce the errors if higher precision is needed.



(a) Right phase.

(b) Left phase.

Figure A.12: Graph showing the error between the input to inverse kinematics and output from the kinematic model when rotating around the hip.

A.3.3 Discussion and Conclusion

Two tests have be conducted. The first test was carried out to verify and test the performance of the inverse kinematic model when performing a translatory motion with the torso. The next test was made to verify and test the performance when rotating the torso with the hip. During the two tests small errors were found. This concludes that the performance of the inverse kinematic is very high and a success. The errors are so small that they will not be noticeable on the physical system as they will be hidden in other noise from e.g. the potentiometers and errors due to backlash.

Appendix B

Verification and Implementation of Controllers

In this appendix, the controllers that are designed in Chapter 7 will be verified. This concerns the controllers from both Control Strategy A and Control Strategy B. Both controller strategies are based on the same main idea. To each control strategy, two controllers with different objectives are proposed. The first controller enables the robot to track the walking trajectories, this is also called a posture controller. The second controller proposed is a balance controller. That works by moving the torso such that the robot is in balance.

B.1 Verification of Control Strategy A

This section describes the implementation and verification of the first control strategy and which involves two controllers. The first controller is for the posture control and the next is the balance controller.

B.1.1 Posture Controller Verification

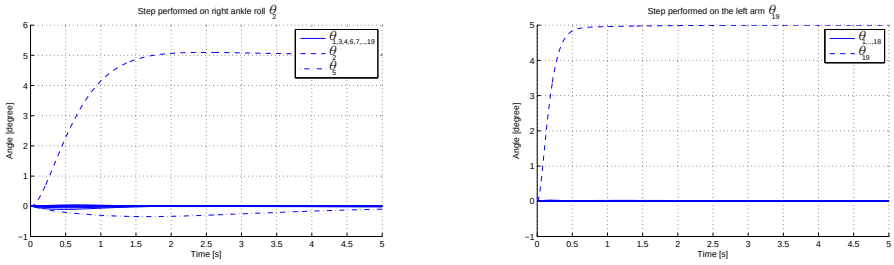
The posture controller consists of a LQG controller. It is verified after a bottom up method. This means that the controller is verified on the linear model first and thereafter on the nonlinear model. Lastly on the Webots representation of the **AAU-BOT1**.

In Section 7.2.1 on page 121 the initial LQG weights on the state are proposed, these weights are for the linear model and are listed below in Equation (B.1) and (B.2)

$$\begin{aligned} Q = & diag[330000\ 180000\ 600000\ 18000\ 980000\ 800000\ 400800\ 900000\ 900000 \\ & 1080000\ 880000\ 80000\ 500000\ 1800\ 800000\ 950000\ 180\ 100000 \\ & 100000\ 60000\ 10\ 90050\ 40000\ 500000\ 250000\ 255000\ 20000\ 300000 \\ & 400000\ 05085\ 9040\ 50000\ 600\ 8000\ 100000\ 20005\ 16080\ 16080] \end{aligned} \tag{B.1}$$

$$\begin{aligned} R = \quad & diag[35\ 25\ 10\ 30\ 25\ 20\ 20\ 500\ 15\ 2.5 \\ & 80\ 3000\ 3000\ 10\ 100\ 20\ 35\ 325\ 325] \end{aligned} \tag{B.2}$$

The Kalman filter is also implemented with the linear model, this will not cause any loss in performance as the Kalman filter is developed from the linear model. In order to see the performance of the controller with the specified weights and the Kalman filter, two steps are applied. It is chosen to step θ_2 which is ankle roll and θ_{19} the left arm with 5°. These two joints are chosen as the first one is heavily influenced by other states and contains relatively large process noise. The two steps on the linear model can be seen in Figure B.1a and B.1b.



(a) Step on θ_2.

(b) Step on θ_{19}.

Figure B.1: Steps performed on the linear model with LQR controller and kalman filter.

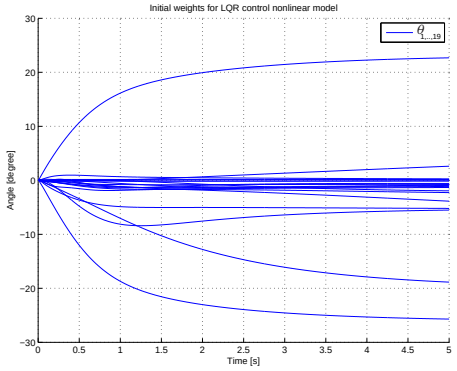
In Figure B.1a a step on ankle roll is performed, where it is possible to see another state θ_5 that is especially influenced by this step. This is because they rotate around the same axis. This is the general picture of the cross coupling situation on the robot. The states that rotates around one axis are most likely to influence other states rotating around the same axis. The LQG controller is able to stabilize the linear model.

The controller can now by tested on the nonlinear model. Initially the controller was not able to stabilize the nonlinear model, therefore the verification is split up into two parts. First with the LQR controller and thereafter adding the kalman filter. In Figure B.2 on the next page(a) the LQR controller with the weights used with the linear model can be seen. The initial weights are not sufficient as multiple states is settling in the area of 20 degrees. To accommodate this problem the weight on the position is roughly raised by a factor 10^6. Because this is the model for SSP-R the first joints are influenced more and therefore they have to be weighted higher than the joints in the end of this kinematic chain. The new gains can be seen in Equation (B.3) and (B.4).

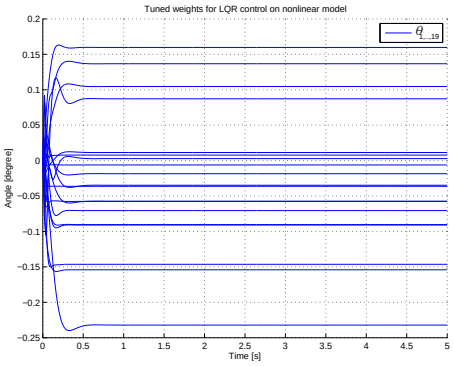
$$\begin{aligned} diag(Q) =& 10^6[830000\ 7800\ 900\ 1800\ 5200\ 800\ 901\ 10\ 70 \\ & 508\ 10\ 100\ 100\ 1.8\ 100\ 950\ 900\ 10\ 10 \\ & 0.06\ 0.000001\ 0.09\ 0.004\ 0.05\ 0.025\ 0.0255\ 0.002\ 0.03\ 0.04 \\ & 0.0205\ 0.00091\ 0.005\ 0.000006\ 0.8\ 0.001\ 0.002\ 0.0002\ 0.0002] \end{aligned} \tag{B.3}$$

$$\begin{aligned} diag(R) = \quad & [35\ 25\ 10\ 30\ 25\ 20\ 20\ 500\ 15\ 2.5 \\ & 80\ 3000\ 3000\ 10\ 100\ 20\ 35\ 325\ 325] \end{aligned} \tag{B.4}$$

In Figure B.2 on the facing page(b) the tuned weights to the nonlinear model are tested. Here it can be seen that the steady state error is in some cases improved by a factor 100. The largest steady state error is now 0.16 and -0.25 degree. The performance



(a) Initial weights on the states, zero input.



(b) Handtuned weights for the states, zero input.

Figure B.2: LQR controller on the nonlinear model.

is accepted and the Kalman filter can now be added to see whether this can stabilize the nonlinear model.

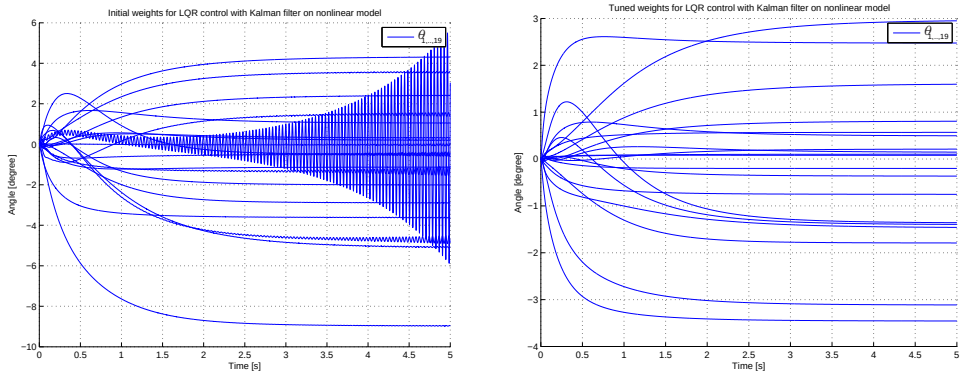
The Kalman filter is now added to the verification. The purpose of the Kalman filter is to estimate the angular velocity of the model and later on the real system. Since the Kalman filter is based on the linearized model it can only to some extent predict the angular velocities. This means that if the nonlinearities are to high or the process noise has a great factor this result in a faulty prediction of the angular velocities.

In Figure B.3 on the next page(a) the tuned weights for the nonlinear model with LQR controller are tested on the nonlinear model both with Kalman filter and LQR controller. Here it can be seen that one state is oscillating, to minimize this oscillating state the weight on the angular velocity is raised a factor 10. Furthermore the weight on the position is raised roughly a factor 10^4 in order to reduce the steady state error. In order to reduce the steady state error further an integral regulator has to be included in this design and thereby forcing the states towards zero. The tunes gains for the LQR controller with the Kalman filter can be seen in Equation (B.5) and (B.6).

$$\begin{aligned} diag(Q) =& 10^{10} \cdot [830000000000\ 180\ 10000\ 580\ 9200000000\ 100\ 20\ 10\ 70 \\ & 90.8\ 50000\ 100\ 1000\ 1.8\ 100\ 15\ 100\ 9\ 9 \\ & 0.00006\ 0.000000001\ 0.00009\ 0.000004\ 0.0005\ 0.00000025 \\ & 0.00000755\ 0.0002\ 0.00003\ 0.000004\ 0.00000205\ 0.0000091 \\ & 0.00001\ 0.00006\ 0.00008\ 0.0005\ 0.0000002\ 0.00000002\ 0.00000002] \end{aligned} \tag{B.5}$$

$$\begin{aligned} diag(R) =& \quad [35\ 25\ 10\ 30\ 25\ 20\ 20\ 500\ 15\ 2.5 \\ & 80\ 3000\ 3000\ 10\ 100\ 20\ 35\ 325\ 325] \end{aligned} \tag{B.6}$$

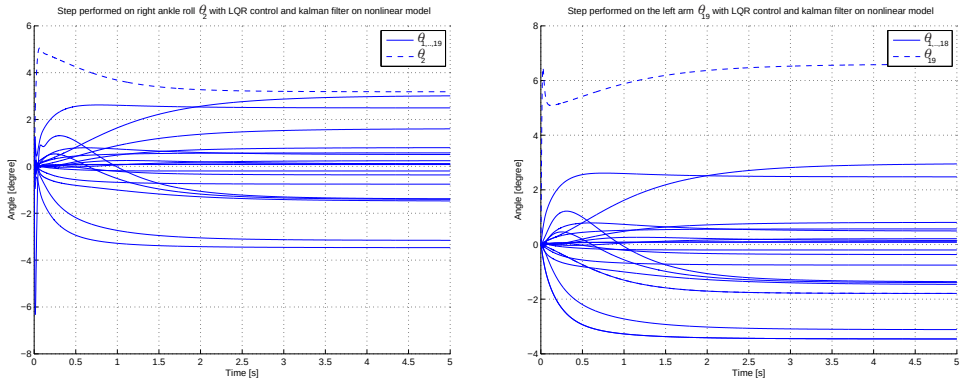
In Figure B.3 on the following page(b) the tuned weight for the Kalman filter and LQR controller can be seen. The oscillation is suppressed and the steady state errors are slightly improved, where the largest error was -9 degrees and is now -3.5 degrees. Now the nonlinear model together with the LQR controller and the Kalman filter is tested



(a) Initial weights on the states, zero input. (b) Altered weights for the states, zero input.

Figure B.3: LQR controller and kalman filter on the nonlinear model.

with two different steps. Both steps have the magnitude of 5 degrees and the first is applied to right ankle roll θ_2 which are heavily influenced by other states. The next is applied to the left arm θ_{19}. The two steps can be seen in Figure B.4(a) and (b).



(a) Step on θ_2. (b) Step on θ_{19}.

Figure B.4: Steps performed on the nonlinear model with LQG controller.

In Figure B.4(a) the controller is not able to settle θ_2 in 5 degrees and has a steady state error of roughly 2 degree. In Figure B.4(b) a step of 5 degrees on the arm is performed here the arm swing up to 5 degrees, but propagates to roughly 7 degrees this presumably happens together with the propagation of torso pitch.

Now the LQR controller together with the Kalman filter can stabilize the non linear model, and the next step is to verify the controller and Kalman filter toward the Webots representation of the **AAU-BOT1**. The verification is done with a zero input and the result can be seen in Figure B.5 on the facing page.

Unfortunately it has not been possible to stabilize the Webots representation of the **AAU-BOT1**. This is presumably because the model is not accurate enough toward the virtual robot in Webots. All the inertia, mass and proportions parameters of the virtual robot has been double checked with the model parameters. It is presumed that it is the friction on the virtual robot in Webots that is not modeled thoroughly. It is

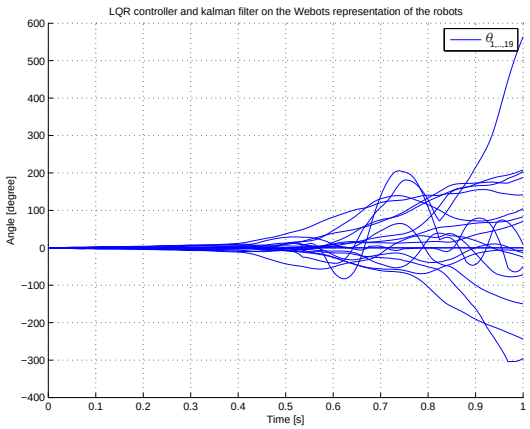


Figure B.5: Zero step on the Webots representation of the robot.

still unknown how the virtual world deals with viscous friction and coulomb friction in motors and joints.

Another important factor is that the model is derived for a robot where the first link, the toe link, is pinned to the ground. This is not entirely right as there exist an unactuated joint between the robot and the floor. This unactuated joint is present in Webots model, but not in the dynamical model for the robot. This means that the first 3-4 joints in the kinematic chain cannot be moved as aggressively as the LQR controller dictates. If the intended LQR controller weights are used it results in a system that moves the lower leg aggressively such that it is the lower leg that is moved and not the upper robot as intended. When this happens the unactuated joint between the robot and the ground comes into play. This results in a unstable system as seen in Figure B.5.

B.1.2 Verification of ZMP Controller

The ZMP controller is verified together with the posture controller. This is a requirement since the ZMP controller is the outer loop in the cascade control strategy. Since the posture controller failed to stabilize the Webots representation of the **AAU-BOT1** it is not possible to verify the ZMP controller in Control Strategy A. However in control strategy B a similar controller is developed and verified, so the interested reader can consult Appendix B.2 where Control Strategy B is verified.

B.1.3 Discussion and Conclusion

It was possible to stabilize the nonlinear model with the proposed LQG controller. This showed that the weights on the states have to be larger on the first joints in the kinematic chain in order to stabilize the nonlinear model. Furthermore it was observed that steady state error was large and it is clear that integral action is needed to minimize these steady state before is can be used on the actual system.

B.2 Verification of Control Strategy B

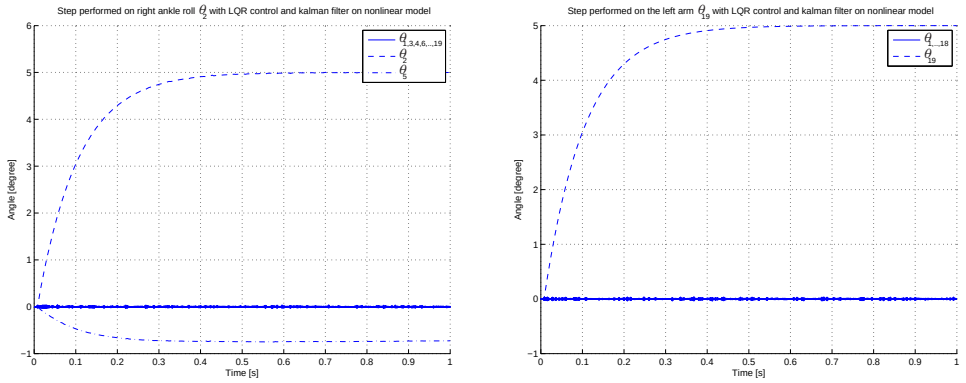
This section describes the implementation and verification of the second control strategy which involves two independent controllers. The first controller is for the posture and the next is a balance controller. The verification is conducted on both the Webots representation of the **AAU-BOT1** and on the actual system.

B.2.1 Webots

Posture Controller Verification

As described in Section 7.3 The posture controller consist of a position controller for the actual **AAU-BOT1** and a P controller for the Webots representation of the robots.

In Figure B.6(a) and B.6(b) two step on the virtual robot in Webots are conducted.



(a) Step on θ_2 (b) Step on θ_{19}

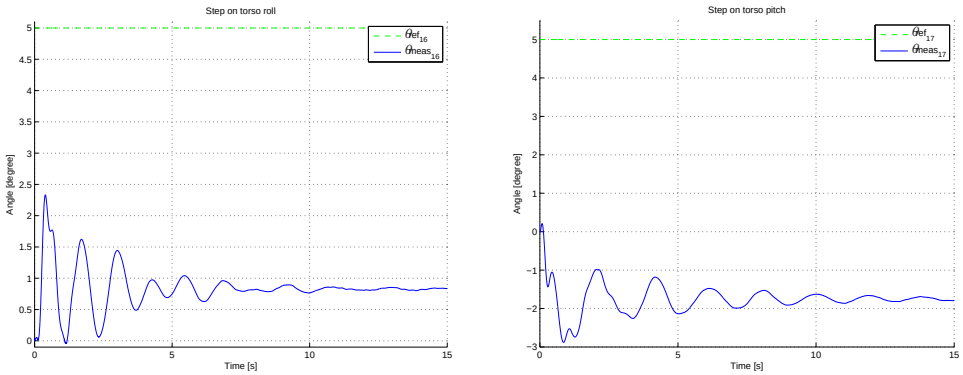
Figure B.6: Steps performed on the virtual robot.

The first steps can be seen in Figure B.6(a) where a step of 5 degree on ankle roll is performed. Here it is possible to see that 150 Nm is not enough for θ_5 to stay in zero. The virtual robot is in DSP at all time during this test. The net step is on the left arm and is also of 5 degree. Here is can be seen that it settles after approximately 0.5 seconds. This is a convenient rate, it is important that the system is not to fast as the system is greatly influenced by cross couplings, and if the arm moves to fast can cause the body to moved unwanted.

B.2.2 ZMP Controller Verification

The torso controller consist of a LQG controller, the Kalman filter in this strategy is proposed to estimate the angular velocity of the torso roll and pitch movements. These are tested on the Webots representation of the **AAU-BOT1**. In Figure B.7 on the facing page two steps of 5 degrees can be seen. In Figure B.7 on the next page(a) it can be seen that the step settles in roughly 1 degree which indicates a large steady state error and reveal that the linear model of the upper torso does not fit the Webots representation of the **AAU-BOT1** very well. Figure B.7 on the facing page(b) shows a even larger steady state error.

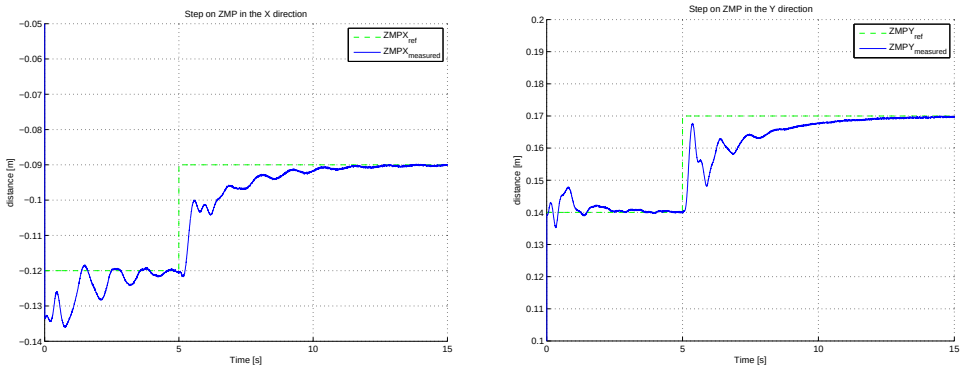
The large steady state error from the test for the LQR controller and the Kalman filter is reduced with the integral action in the PI controller for the ZMP controller. In



(a) Step on θ_{16} (b) Step on θ_{17}

Figure B.7: Steps performed on the webots representation of the robot.

Figure B.8 two steps on the ZMP controller can be seen. In Figure B.8(a) step on the x-direction can be seen and in Figure B.8(b) a step in the y-direction can be seen.



(a) Step on ZMP in x-direction (b) Step on ZMP in y-direction

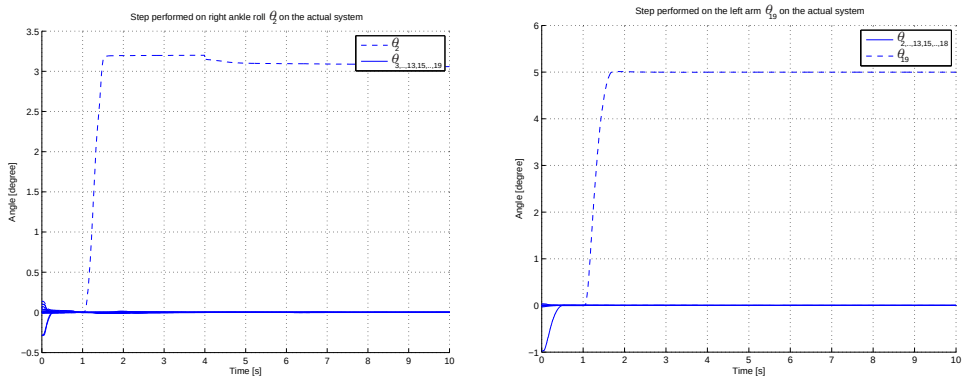
Figure B.8: Steps performed on the webots representation of the robot.

The large steady state error is gone and the system settles with an oscillation that is slowly suppressed. This can be explained by the dynamics in the Webots. The robots can rock back and forward by a small factor and that causes some of the oscillation.

B.2.3 Actual AAU-BOT1

In order to test the actual system a step has been performed at the ankle joint θ_2. In Figure B.9 on the next page-(a) the step is shown. As it can be seen there is a large offset error, due to that the joint is only actuated by one motor and it should be double actuated. In the time period 1.5-4 sec the angle is higher, this is because the EPOS allow the current to be more than the nominal current for 2.5 sec., before it limits the current to nominal current again. Since motor cannot produce enough torque to obtain a step on 5 degree it has a large steady state error. In Figure B.9 on the following page-(b) a step on the joint in the arm θ_{19} is performed. As shown at the graph the arm has no problem with a step on 5 degrees. On both graphs there is a small spike in the beginning

of the test, this is because **AAU-BOT1** has to obtain 0 degrees on all joints, before it perform the step after 1 sec.



(a) Step on θ_2.

(b) Step on θ_{19}.

Figure B.9: Steps performed on the actual system.

B.2.4 Discussion and Conclusion

Now the posture controller and the ZMP controller is verified. The ZMP controller is able to track a ZMP reference, but it is slow due to when a step is performed on the ZMP the acceleration of the torso affects the ZMP in the opposite direction. It is important that this controller is calm and does not have any fast movements, as this can cause the robot to tip and thereby become unstable. At the real system is was clearly that **AAU-BOT1** has problems with obtaining the wanted joint angle in the joint where it have been double actuated joints. Further groups at the **AAU-BOT1** project must enable this feature in order to obtain a steady state error at zero.

Appendix C

Mechanical Data

In this appendix, the mechanical data of **AAU-BOT1**, extracted from[Pedersen et al., 2007, app. N] will be listed in table C.1 to C.4.

Table C.1: Motor/joint properties.

Motors/joints	**mrotor**	**Jgear + Jrotor**	**Jbelt (est).**	**ubelt**	**ugear**	**utot**
Waist yaw	m60	J60 + J17	0.1080e-4	48/16	100	300
Waist pitch	m150	J150 + J17	0.3171e-4	56/18	120	373
Waist roll	m90	J90 + J14	0.2911e-4	62/18	100	344
Shoulder pitch	m60	J60 + J14	0.0051e-4	20/18	100	111
Hip yaw	m60	J60 + J17	0.0510e-4	40/16	100	250
Hip roll	2 m150	2 J150 + J20	0.5576e-4	72/30	120	288
Hip pitch	m150	J150 + J20	0.2937e-4	60/28	160	342
Knee pitch	2 m150	2 J150 + J17	0.2111e-4	32/24	100	133
Ankle pitch	2 m150	2 J150 + J20	0.0421e-4	38/18	100	211
Ankle roll	m90	J90 + J14	0.0789e-4	44/22	100	200

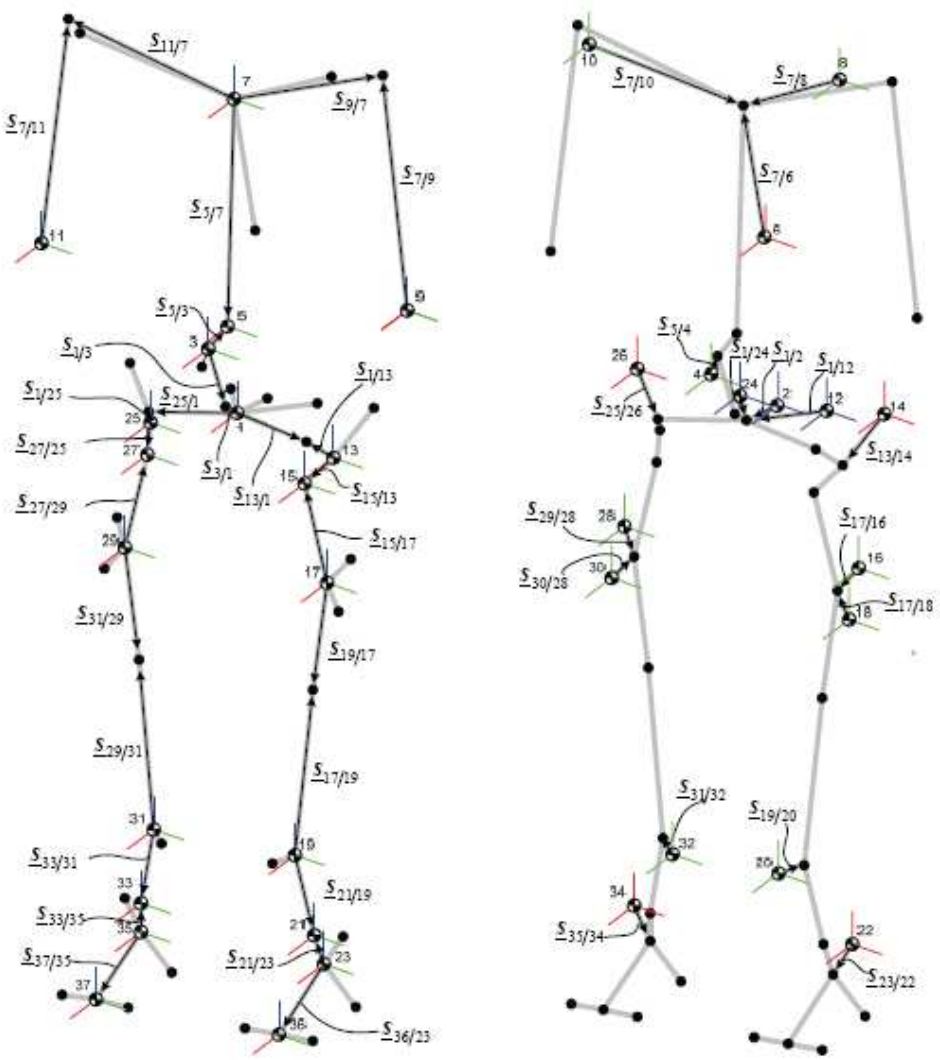


Figure C.1: Local vectors from the individual body's CoM to the joints connecting the adjacent bodies. Regular bodies (left) and motor bodies (right). The colors of the rotor coordinate systems, defines the axis of rotation (red, green and blue for roll, pitch and yaw, respectively). The size of the vectors are found in table C.3.

Table C.2: Overview of 37 bodys that **AAU-BOT1** consists of [Pedersen et al., 2007, App. N].

#	Name	Mass [kg]	Inertia [x y z] [kg·m^2]
1	Pelvis	4.90	[5.47e-02 1.29e-02 6.57e-02]
2	Waist yaw motor	0.28	[3.34e-05 3.34e-05 3.34e-05]
3	Waist pitch joint	2.27	[9.88e-03 6.78e-03 5.87e-03]
4	Waist pitch motor	0.48	[6.48e-05 6.48e-05 6.48e-05]
5	Waist cross	0.75	[3.49e-04 9.56e-04 1.21e-03]
6	Waist roll motor	0.34	[4.50e-05 4.50e-05 4.50e-05]
7	Torso	21.60	[8.26e-01 5.86e-01 5.02e-01]
8	Left shoulder motor	0.28	[1.30e-05 1.30e-05 1.30e-05]
9	Left arm	0.90	[3.07e-02 4.19e-02 1.25e-02]
10	Right shoulder motor	0.28	[1.30e-05 1.30e-05 1.30e-05]
11	Right arm	0.90	[3.07e-02 4.19e-02 1.25e-02]
12	Left hip yaw motor	0.28	[2.78e-05 2.78e-05 2.78e-05]
13	Left hip roll joint	3.07	[1.61e-02 1.27e-02 9.32e-03]
14	Left hip roll motor	0.96	[1.24e-04 1.24e-04 1.24e-04]
15	Left hip cross joint	0.64	[4.70e-04 3.24e-04 6.44e-04]
16	Left hip pitch motor	0.48	[8.36e-05 8.36e-05 8.36e-05]
17	Left thigh	6.66	[9.90e-02 8.94e-02 2.40e-02]
18	Left knee motor	0.96	[6.80e-05 6.80e-05 6.80e-05]
19	Left shin	4.97	[7.26e-02 6.86e-02 1.30e-02]
20	Left ankle pitch motor	0.96	[7.22e-05 7.22e-05 7.22e-05]
21	Left ankle cross joint	0.47	[1.81e-04 3.58e-04 4.34e-04]
22	Left ankle roll motor	0.34	[2.38e-05 2.38e-05 2.38e-05]
23	Left foot	2.75	[7.89e-03 1.67e-02 1.45e-02]
24	Right hip yaw motor	0.28	[2.78e-05 2.78e-05 2.78e-05]
25	Right roll hip joint	3.07	[1.61e-02 1.27e-02 9.32e-03]
26	Right hip roll motor	0.96	[1.24e-04 1.24e-04 1.24e-04]
27	Right hip cross joint	0.64	[4.70e-04 3.24e-04 6.44e-04]
28	Right hip pitch motor	0.48	[8.36e-05 8.36e-05 8.36e-05]
29	Right thigh	6.66	[9.90e-02 8.94e-02 2.40e-02]
30	Right knee motor	0.96	[6.80e-05 6.80e-05 6.80e-05]
31	Right shin	4.97	[7.26e-02 6.86e-02 1.30e-02]
32	Right ankle pitch motor	0.96	[7.22e-05 7.22e-05 7.22e-05]
33	Right ankle cross joint	0.47	[1.81e-04 3.58e-04 4.34e-04]
34	Right ankle roll motor	0.34	[2.38e-05 2.38e-05 2.38e-05]
35	Right foot	2.75	[7.89e-03 1.67e-02 1.45e-02]
36	Left toe	0.01	[1.39e-04 4.26e-04 3.12e-04]
37	Right toe	0.01	[1.39e-04 4.26e-04 3.12e-04]

Table C.3: Overview of distance vectors that **AAU-BOT1** consists of [Pedersen et al., 2007, App. N].

Vector	Size [mm]	Vector	Size [mm]
$s_{1/2}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{1/3}$	$[\ 4\ \ 38\ \ -80\ ]^T$
$s_{1/12}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{1/13}$	$[\ 38\ \ -24\ \ 28\ ]^T$
$s_{1/24}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{1/25}$	$[\ 38\ \ 24\ \ 28\ ]^T$
$s_{2/1}$	$[\ -80\ \ 0\ \ 4\ ]^T$	$s_{3/1}$	$[\ 22\ \ 0\ \ 14\ ]^T$
$s_{3/4}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{3/5}$	$[\ 0\ \ 0\ \ 0\ ]^T$
$s_{4/3}$	$[\ -16\ \ -22\ \ -32\ ]^T$	$s_{5/3}$	$[\ 4\ \ 38\ \ 40\ ]^T$
$s_{5/7}$	$[\ 19\ \ 0\ \ -341\ ]^T$	$s_{6/7}$	$[\ -54\ \ 0\ \ -211\ ]^T$
$s_{7/5}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{7/6}$	$[\ 0\ \ 0\ \ 0\ ]^T$
$s_{7/8}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{7/9}$	$[\ -4\ \ -48\ \ 347\ ]^T$
$s_{7/10}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{7/11}$	$[\ -4\ \ 48\ \ 347\ ]^T$
$s_{8/7}$	$[\ 75\ \ 222\ \ 84\ ]^T$	$s_{9/7}$	$[\ 25\ \ 280\ \ 84\ ]^T$
$s_{10/7}$	$[\ 75\ \ -222\ \ 84\ ]^T$	$s_{11/7}$	$[\ 25\ \ -280\ \ 84\ ]^T$
$s_{12/1}$	$[\ -96\ \ 77\ \ 4\ ]^T$	$s_{13/1}$	$[\ 22\ \ 140\ \ -16\ ]^T$
$s_{13/14}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{13/15}$	$[\ 0\ \ 0\ \ 0\ ]^T$
$s_{14/13}$	$[\ -24\ \ 54\ \ 77\ ]^T$	$s_{15/13}$	$[\ 41\ \ -24\ \ -36\ ]^T$
$s_{15/17}$	$[\ 2\ \ -41\ \ 146\ ]^T$	$s_{16/17}$	$[\ -19\ \ 27\ \ 37\ ]^T$
$s_{17/15}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{17/16}$	$[\ 0\ \ 0\ \ 0\ ]^T$
$s_{17/18}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{17/19}$	$[\ -4\ \ 30\ \ 252\ ]^T$
$s_{18/17}$	$[\ 13\ \ 27\ \ -34\ ]^T$	$s_{19/17}$	$[\ 2\ \ -41\ \ -165\ ]^T$
$s_{19/20}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{19/21}$	$[\ 0\ \ 0\ \ 0\ ]^T$
$s_{20/19}$	$[\ 21\ \ -25\ \ -13\ ]^T$	$s_{21/19}$	$[\ -4\ \ 30\ \ -118\ ]^T$
$s_{21/23}$	$[\ 19\ \ -10\ \ 46\ ]^T$	$s_{22/23}$	$[\ 0\ \ 30\ \ 46\ ]^T$
$s_{23/21}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{23/22}$	$[\ 0\ \ 0\ \ 0\ ]^T$
$s_{23/36}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{24/1}$	$[\ -96\ \ -77\ \ 4\ ]^T$
$s_{25/1}$	$[\ 22\ \ -140\ \ -16\ ]^T$	$s_{25/26}$	$[\ 0\ \ 0\ \ 0\ ]^T$
$s_{25/27}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{26/25}$	$[\ -24\ \ -54\ \ 77\ ]^T$
$s_{27/25}$	$[\ 41\ \ 24\ \ -36\ ]^T$	$s_{27/29}$	$[\ 2\ \ 41\ \ 146\ ]^T$
$s_{28/29}$	$[\ -19\ \ -27\ \ 37\ ]^T$	$s_{29/27}$	$[\ 0\ \ 0\ \ 0\ ]^T$
$s_{29/28}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{29/30}$	$[\ 0\ \ 0\ \ 0\ ]^T$
$s_{29/31}$	$[\ -4\ \ -30\ \ 252\ ]^T$	$s_{30/29}$	$[\ 13\ \ -27\ \ -34\ ]^T$
$s_{31/29}$	$[\ 2\ \ 41\ \ -165\ ]^T$	$s_{31/32}$	$[\ 0\ \ 0\ \ 0\ ]^T$
$s_{31/33}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{32/31}$	$[\ 21\ \ 25\ \ -13\ ]^T$
$s_{33/31}$	$[\ -4\ \ -30\ \ -118\ ]^T$	$s_{33/35}$	$[\ 19\ \ 10\ \ 46\ ]^T$
$s_{34/35}$	$[\ 0\ \ -30\ \ 46\ ]^T$	$s_{35/33}$	$[\ 0\ \ 0\ \ 0\ ]^T$
$s_{35/34}$	$[\ 0\ \ 0\ \ 0\ ]^T$	$s_{35/37}$	$[\ 0\ \ 0\ \ 0\ ]^T$
$s_{36/23}$	$[\ 132\ \ 10\ \ -76\ ]^T$	$s_{37/35}$	$[\ 132\ \ 10\ \ -76\ ]^T$

Table C.4: Overview of Rotation Joints (RJs) **AAU-BOT1** consists of. [Pedersen et al., 2007, App. N]

Joint	**Joint name**	**Vector**	**Drive**
$RJ_{1/2}$	Waist yaw motor	$[\ 0\ \ 0\ \ 1\ ]^T$	Gear Output
$RJ_{1/3}$	Waist yaw joint	$[\ 0\ \ 0\ \ 1\ ]^T$	Motor Joint
$RJ_{3/4}$	Waist pitch motor	$[\ 0\ \ 1\ \ 0\ ]^T$	Gear Output
$RJ_{3/5}$	Waist pitch joint	$[\ 0\ \ 1\ \ 0\ ]^T$	Motor Joint
$RJ_{7/6}$	Waist roll motor	$[\ 1\ \ 0\ \ 0\ ]^T$	Gear Output
$RJ_{5/7}$	Waist roll joint	$[\ 1\ \ 0\ \ 0\ ]^T$	Gear Input
$RJ_{7/8}$	Left shoulder motor	$[\ 0\ \ 1\ \ 0\ ]^T$	Gear Output
$RJ_{7/9}$	Left shoulder joint	$[\ 0\ \ 1\ \ 0\ ]^T$	Motor Joint
$RJ_{7/10}$	Right shoulder motor	$[\ 0\ \ 1\ \ 0\ ]^T$	Gear Output
$RJ_{7/11}$	Right shoulder joint	$[\ 0\ \ 1\ \ 0\ ]^T$	Motor Joint
$RJ_{1/12}$	Left hip yaw motor	$[\ 0\ \ 0\ \ 1\ ]^T$	Gear Output
$RJ_{1/13}$	Left hip yaw joint	$[\ 0\ \ 0\ \ 1\ ]^T$	Motor Joint
$RJ_{13/14}$	Left hip roll motor	$[\ 1\ \ 0\ \ 0\ ]^T$	Gear Output
$RJ_{13/15}$	Left hip roll joint	$[\ 1\ \ 0\ \ 0\ ]^T$	Motor Joint
$RJ_{17/16}$	Left hip pitch motor	$[\ 0\ \ 1\ \ 0\ ]^T$	Gear Output
$RJ_{15/17}$	Left hip pitch joint	$[\ 0\ \ 1\ \ 0\ ]^T$	Gear Input
$RJ_{17/18}$	Left knee motor	$[\ 0\ \ 1\ \ 0\ ]^T$	Gear Output
$RJ_{17/19}$	Left knee joint	$[\ 0\ \ 1\ \ 0\ ]^T$	Motor Joint
$RJ_{19/20}$	Left ankle pitch motor	$[\ 0\ \ 1\ \ 0\ ]^T$	Gear Output
$RJ_{19/21}$	Left ankle pitch joint	$[\ 0\ \ 1\ \ 0\ ]^T$	Motor Joint
$RJ_{23/22}$	Left ankle roll motor	$[\ 1\ \ 0\ \ 0\ ]^T$	Gear Output
$RJ_{21/23}$	Left ankle roll joint	$[\ 1\ \ 0\ \ 0\ ]^T$	Gear Input
$RJ_{1/24}$	Right hip yaw motor	$[\ 0\ \ 0\ \ 1\ ]^T$	Gear Output
$RJ_{1/25}$	Right hip yaw joint	$[\ 0\ \ 0\ \ 1\ ]^T$	Motor Joint
$RJ_{25/26}$	Right hip roll motor	$[\ 1\ \ 0\ \ 0\ ]^T$	Gear Output
$RJ_{25/27}$	Right hip roll joint	$[\ 1\ \ 0\ \ 0\ ]^T$	Motor Joint
$RJ_{29/28}$	Right hip pitch motor	$[\ 0\ \ 1\ \ 0\ ]^T$	Gear Output
$RJ_{27/29}$	Right hip pitch joint	$[\ 0\ \ 1\ \ 0\ ]^T$	Gear Input
$RJ_{29/30}$	Right knee motor	$[\ 0\ \ 1\ \ 0\ ]^T$	Gear Output
$RJ_{29/31}$	Right knee joint	$[\ 0\ \ 1\ \ 0\ ]^T$	Motor Joint
$RJ_{31/32}$	Right ankle pitch motor	$[\ 0\ \ 1\ \ 0\ ]^T$	Gear Output
$RJ_{31/33}$	Right ankle pitch joint	$[\ 0\ \ 1\ \ 0\ ]^T$	Motor Joint
$RJ_{35/34}$	Right ankle roll motor	$[\ 1\ \ 0\ \ 0\ ]^T$	Gear Output
$RJ_{33/35}$	Right ankle roll joint	$[\ 1\ \ 0\ \ 0\ ]^T$	Gear Input
$RJ_{23/36}$	Left toe	$[\ 0\ \ 1\ \ 0\ ]^T$	Unactuated
$RJ_{35/37}$	Right toe	$[\ 0\ \ 1\ \ 0\ ]^T$	Unactuated

Appendix D

Foot Model

In this appendix the foot model is desrived. The purpose of the foot model is to determine the forces and the torques that exerted on the foot.

The forces and moments depends on which hybrid state the robot is in. E.g. if the **AAU-BOT1** is in SSP-L, the right foot will not be affected by the passive springs or forces caused by the floor. Figure D.1 is a simplified sketch of the foot model. Further explanation will be given in the following sections.

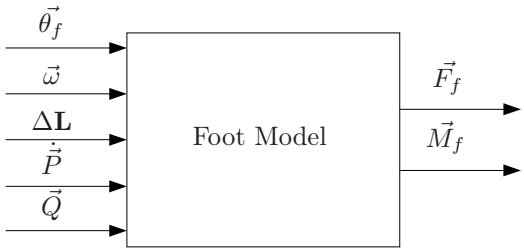


Figure D.1: Foot model input and output relations.

Where:

$\vec{\theta_f}$	is the CoM rotation of the foot.
$\vec{\omega}$	is the CoM rotational velocity of the foot.
$\Delta\mathbf{L}$	is the penetration depth of the spring in the heel [m].
$\dot{\vec{P}_f}$	is the x,y,z CoM velocities of the foot.
Q	is the phase model that the robot is in.
$\vec{F_f}$	is the forces that are acting on the CoM of the foot.
$\vec{M_f}$	is the moments that are acting on the CoM of the foot.

D.1 Foot Design Overview

Very few of the observed existing robots incorporates toes, since they are unnecessary when walking flat-footed. The objective of this project is to make the **AAU-BOT1** walk dynamically and it is therefore essential that the toe is implemented on **AAU-BOT1**. The feet have been designed as a spring actuated system, i.e. a spring is attached to the toe and springs have been mounted in the heel. Figure D.2 shows a sketch of the developed foot.

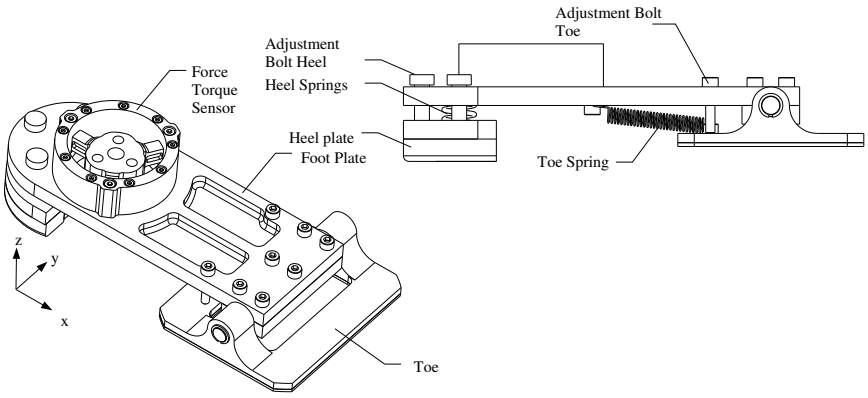


Figure D.2: Sketch of the designed foot with springs in heel and toe.

D.2 Forces and Torques on the Foot

The force and the moment caused by springs in the heel and the toe can be decomposed in sets of rotational and translational 3D-vectors. Initially the **AAU-BOT1** is in DSP, i.e. both feet have contact with the floor, and the translational spring reference position is recorded as the initial position of the feet. The reference angles for the rotational springs are set to zero during the initial phase.

The coefficients have been found by [Pedersen et al., 2007] via laboratory experiments. The experiments showed that the rotational and the translational spring and dampening coefficients are non-linear, and they are therefore scaled to ensure a smooth walking. The coefficients have to be calculated for each foot, the general coefficients are described below and the penetration depth is sketched at Figure D.3:

$$k_t = \frac{m_{\text{Tot}} g}{\mathbf{L}_{\max}} = \frac{68.47_{kg} \cdot 9.82_{\frac{m}{s^2}}}{0.001_m} \tag{D.1}$$

$$c_t = 7000_{\frac{Ns}{m}} \tag{D.2}$$

$$k_r = 5000_{\frac{Nm}{rad}} \tag{D.3}$$

$$c_r = 35000_{\frac{Nms}{rad}} \tag{D.4}$$

$$\vec{k_t} = k_t[0.5 \quad 0.2 \quad 1.0]^T \tag{D.5}$$

$$\vec{c_t} = c_t[0.5 \quad 0.2 \quad 1.0]^T \frac{\Delta \mathbf{L}_f}{\mathbf{L}_{\max}} \tag{D.6}$$

$$\vec{k_r} = k_r[1.0 \quad 1.0 \quad 0.5]^T \tag{D.7}$$

$$\vec{c_r} = c_r[1.0 \quad 1.0 \quad 0.5]^T \frac{\Delta \mathbf{L}_f}{\mathbf{L}_{\max}} \tag{D.8}$$

Where:

$\Delta\mathbf{L}_f$ is the penetration depth of the spring in the heel [m].
$\mathbf{L}_{\max}$ is the maximum penetration depth of the spring in the heel [m].
m_{Tot} is the **AAU-BOT1**'s total weight [kg].
$\vec{c_t}$ is the translational dampening coefficients $[\frac{\text{N s}}{\text{m}}]$.
$\vec{k_t}$ is the translational spring coefficients $[\frac{\text{N}}{\text{m}}]$.
$\vec{c_r}$ is the rotational dampening coefficients $[\frac{\text{Nm s}}{\text{rad}}]$.
$\vec{k_r}$ is the rotational spring coefficients $[\frac{\text{Nm}}{\text{rad}}]$.

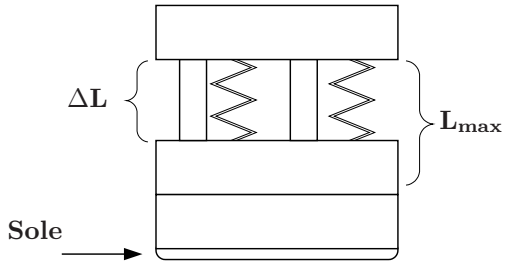


Figure D.3: Sketch of the heel

The resulting forces $\vec{F}$ and moments $\vec{M}$ acting on the CoM of the foot are calculated as the sum of the spring and damper forces and moments as in Equations (D.9), (D.10), (D.11) and (D.12) where (f, L) denotes the left foot's link index and (f, R) denotes the right foot's link index.

$$\vec{F}_{f,L} = \begin{cases} \vec{k}_t^{\mathrm{T}} \Delta\mathbf{L}_{f,L} - \vec{c}_t^{\mathrm{T}} \dot{\vec{P}}_{f,L} & \text{if} \quad Q = \{q_1, q_5, q_6, q_8, q_{10}\} \\ 0 & \text{else} \end{cases} \tag{D.9}$$

$$\vec{F}_{f,R} = \begin{cases} \vec{k}_t^{\mathrm{T}} \Delta\mathbf{L}_{f,R} - \vec{c}_t^{\mathrm{T}} \dot{\vec{P}}_{f,R} & \text{if} \quad Q = \{q_2, q_5, q_6, q_7, q_9\} \\ 0 & \text{else} \end{cases} \tag{D.10}$$

$$\vec{M}_{f,L} = \begin{cases} \vec{k}_r^{\mathrm{T}} \Delta\boldsymbol{\Theta}_{f,L} - \vec{c}_r^{\mathrm{T}} \boldsymbol{\omega}_{f,L} & \text{if} \quad Q = \{q_1, q_3, q_5, q_7, q_9\} \\ 0 & \text{else} \end{cases} \tag{D.11}$$

$$\vec{M}_{f,R} = \begin{cases} \vec{k}_r^{\mathrm{T}} \Delta\boldsymbol{\Theta}_{f,R} - \vec{c}_r^{\mathrm{T}} \boldsymbol{\omega}_{f,R} & \text{if} \quad Q = \{q_2, q_3, q_5, q_8, q_{10}\} \\ 0 & \text{else} \end{cases} \tag{D.12}$$

In Section 5.4 the true rotational and the translational spring and dampening coefficients are to be fully determined when the final weight has been found and motor dynamic have been determined since to high coefficients will result in saturation of the motor during dynamic gait.

D.3 Constraints

Since there are some physical limitations the forces and moments are defined in restricted areas, e.g. F_z can not be less than zero, since it otherwise will mean that the foot is pulled down by the floor. The forces and moments are calculated positive on the foot and negative on the floor. The vertical force is defined as:

$$F_z \geq 0 \tag{D.13}$$

Furthermore the two transverse moments, M_x and M_y, are also limited because of the foot's design. Figure D.4 is a sketch of the limits that are applied to the horizontal moments. Equation (D.14) and (D.15) describes the applied limits mathematical.

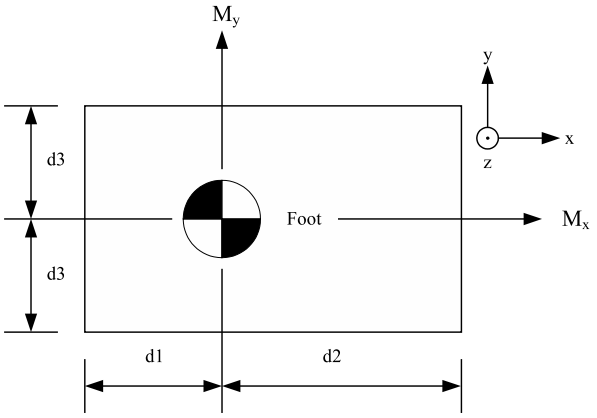


Figure D.4: Sketch of the foot with limits.

$$-d_2 F_z \leq M_y \leq d_1 F_z \tag{D.14}$$
$$-d_3 F_z \leq M_x \leq d_3 F_z \tag{D.15}$$

If the transverse moments are higher than the limits, they are simply set to the maximum value. The traction forces are also limited, which makes it possible to slip the floor. The maximum forces must not exceed the following values, otherwise they are scaled down. Equation (D.16) describe the relation to traction forces and the friction force.

$$\sqrt{F_x^2 + F_y^2} \leq \mu F_z \tag{D.16}$$

Group [Pedersen et al., 2007] has determined the friction coefficient (μ) to be 0.5 via simulation. Based on the result of the simulations all traction forces are ignored if the foot moves upwards, such that only the vertical spring forces acts on the foot.

Appendix E

Motivating Example

This chapter will go through the calculations in Appendix 5.4 and 5.5 on a relatively simple (3 DoF, 2 dimensional) biped robot named Strider. The distances and revolution joints are defined in Figure E.1. Only the left phases are calculated, as the right phases are calculated using the same algorithm from the right side. It is assumed that Strider has the zero moment point within its support area at all times, so that Strider does not rotate around origo (PJ_0).

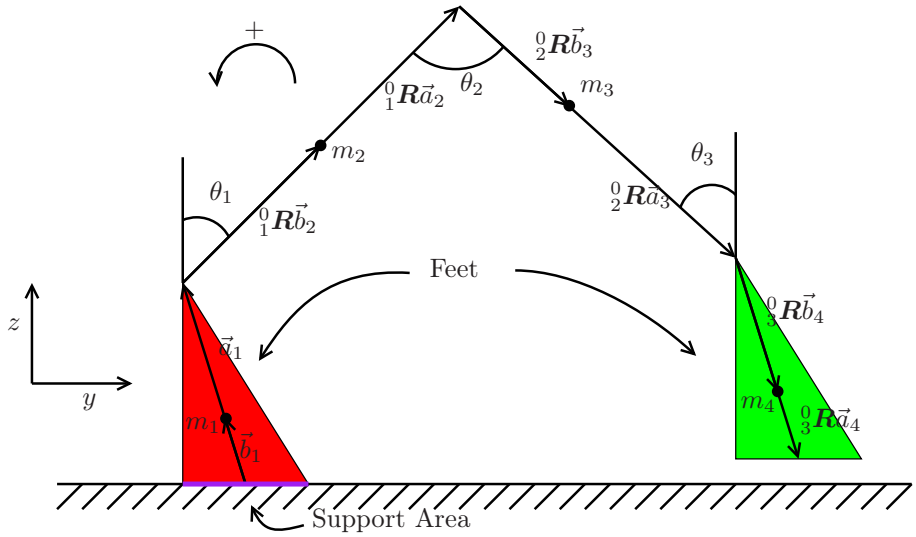


Figure E.1: Definitions of the joint angles and link parameters for the simple 3 DoF imaginary biped robot (Strider) in SSP-L. Strider is only used to demonstrate the methods used for creating the kinematic and dynamic model for **AAU-BOT1**.

E.1 Kinematic Model

The purpose of the kinematic model is to determine the position, velocity and acceleration of the individual links, see Figure E.2. The global positions of the CoM of the individual

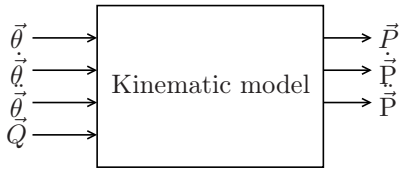


Figure E.2: Block diagram of kinematic model with joint angles as input and generalized coordinates as output. Q determines which state the model is in, in this example the model is in SSP-L.

links of Strider can be calculated using Equation (E.1) [Craig, 2005, Chap. 2]:

$$\vec{P}_j = \left(\prod_{i=1}^{j-1} \begin{bmatrix} {}^{i-1}_{i}\boldsymbol{R} & \vec{a}_i \\ \mathbf{0} & 1 \end{bmatrix} \right) \begin{pmatrix} \vec{b}_j \\ 1 \end{pmatrix} \tag{E.1}$$

where:

$\vec{P}_j$	is the position of the CoM of link j, in global coordinates
${}^{i-1}_{i}\boldsymbol{R}$	is the rotation matrix of joint i
$\vec{a}_i$	is the distance vector from joint $i-1$ to i
$\vec{b}_j$	is the distance vector from joint $j-1$ to the CoM of link j, in local coordinates

As the rotation only occurs around the x-axis, ${}^{i-1}_{i}\boldsymbol{R}$ is simplified to:

$${}^{i-1}_{i}\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_i & -s_i \\ 0 & s_i & c_i \end{bmatrix} \tag{E.2}$$

where c_i and s_i are abbreviations of $\cos(\theta_i)$ and $\sin(\theta_i)$, respectively. As θ_0 and $\vec{a}_0$ describes the starting point of the chain,

$$\theta_0 = 0, \quad \vec{a}_0 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^{\mathrm{T}} \tag{E.3}$$

The movement in the x direction is neglected in this model, thus the generalized position vector is reduced to:

$$\vec{q} = \begin{pmatrix} y_1 & y_2 & y_3 & y_4 & z_1 & z_2 & z_3 & z_4 & \theta_1 & \theta_2 & \theta_3 \end{pmatrix}^{\mathrm{T}} \tag{E.4}$$

where:

y_i	is the y coordinate of the CoM of link i, in global coordinates
z_i	is the z coordinate of the CoM of link i, in global coordinates
θ_i	is the relative angle of joint i

Utilizing Equation (E.1) on Equation (E.4) yields:

$$\vec{q} = \begin{bmatrix} b_{1y} \\ a_{1y} + c_1 b_{2y} - s_1 b_{2z} \\ a_{1y} + c_1 a_{2y} - s_1 a_{2z} + c_{1+2} b_{3y} - s_{1+2} b_{3z} \\ a_{1y} + c_1 a_{2y} - s_1 a_{2z} + c_{1+2} a_{3y} \\ -s_{1+2} a_{3z} + c_{1+2+3} b_{4y} - s_{1+2+3} b_{4z} \\ b_{1z} \\ a_{1z} + s_1 b_{2y} + c_1 b_{2z} \\ a_{1z} + s_1 a_{2y} + c_1 a_{2z} + s_{1+2} b_{3y} + c_{1+2} b_{3z} \\ a_{1z} + s_1 a_{2y} + c_1 a_{2z} + s_{1+2} a_{3y} \\ +c_{1+2} a_{3z} + s_{1+2+3} b_{4y} + c_{1+2+3} b_{4z} \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \tag{E.5}$$

As the dynamic model requires $\ddot{\vec{q}}$, Equation (E.5) is differentiated twice:

$$\dot{\vec{q}} = \begin{bmatrix} 0 \\ -s_1 \dot{\theta}_1 b_{2y} - c_1 \dot{\theta}_1 b_{2z} \\ -s_1 \dot{\theta}_1 a_{2y} - c_1 \dot{\theta}_1 a_{2z} - s_{1+2} b_{3y} \dot{\theta}_1 - s_{1+2} b_{3y} \dot{\theta}_2 \\ -c_{1+2} b_{3z} \dot{\theta}_1 - c_{1+2} b_{3z} \dot{\theta}_2 \\ -s_1 \dot{\theta}_1 a_{2y} - c_1 \dot{\theta}_1 a_{2z} - s_{1+2} a_{3y} \dot{\theta}_1 - s_{1+2} a_{3y} \dot{\theta}_2 \\ -c_{1+2} a_{3z} \dot{\theta}_1 - c_{1+2} a_{3z} \dot{\theta}_2 - s_{1+2+3} b_{4y} \dot{\theta}_1 - s_{1+2+3} b_{4y} \dot{\theta}_2 \\ -s_{1+2+3} b_{4y} \dot{\theta}_3 - c_{1+2+3} b_{4z} \dot{\theta}_1 - c_{1+2+3} b_{4z} \dot{\theta}_2 - c_{1+2+3} b_{4z} \dot{\theta}_3 \\ 0 \\ c_1 \dot{\theta}_1 b_{2y} - s_1 \dot{\theta}_1 b_{2z} \\ c_1 \dot{\theta}_1 a_{2y} - s_1 \dot{\theta}_1 a_{2z} + c_{1+2} b_{3y} \dot{\theta}_1 + c_{1+2} b_{3y} \dot{\theta}_2 \\ -s_{1+2} b_{3z} \dot{\theta}_1 - s_{1+2} b_{3z} \dot{\theta}_2 \\ c_1 \dot{\theta}_1 a_{2y} - s_1 \dot{\theta}_1 a_{2z} + c_{1+2} a_{3y} \dot{\theta}_1 + c_{1+2} a_{3y} \dot{\theta}_2 \\ -s_{1+2} a_{3z} \dot{\theta}_1 - s_{1+2} a_{3z} \dot{\theta}_2 + c_{1+2+3} b_{4y} \dot{\theta}_1 + c_{1+2+3} b_{4y} \dot{\theta}_2 \\ +c_{1+2+3} b_{4y} \dot{\theta}_3 - s_{1+2+3} b_{4z} \dot{\theta}_1 - s_{1+2+3} b_{4z} \dot{\theta}_2 - s_{1+2+3} b_{4z} \dot{\theta}_3 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \tag{E.6}$$

$$\ddot{\vec{q}} = \begin{bmatrix} 0 \\ -c_1\dot{\theta}_1^2 b_{2y} - s_1\ddot{\theta}_1 b_{2y} + s_1\dot{\theta}_1^2 b_{2z} - c_1\ddot{\theta}_1 b_{2z} \\ -c_1\dot{\theta}_1^2 a_{2y} - s_1\ddot{\theta}_1 a_{2y} + s_1\dot{\theta}_1^2 a_{2z} - c_1\ddot{\theta}_1 a_{2z} \\ -b_{3y}\dot{\theta}_1^2 c_{1+2} - b_{3y}\ddot{\theta}_1 s_{1+2} - 2\,b_{3y}\dot{\theta}_1\dot{\theta}_2 c_{1+2} - b_{3y}\dot{\theta}_2^2 c_{1+2} \\ -b_{3y}\ddot{\theta}_2 s_{1+2} + b_{3z}\dot{\theta}_1^2 s_{1+2} - b_{3z}\ddot{\theta}_1 c_{1+2} + 2\,b_{3z}\dot{\theta}_1\dot{\theta}_2 s_{1+2} \\ +b_{3z}\dot{\theta}_2^2 s_{1+2} - b_{3z}\ddot{\theta}_2 c_{1+2} \\ b_{4z}\dot{\theta}_2^2 s_{1+2+3} - b_{4z}\ddot{\theta}_2 c_{1+2+3} + 2\,b_{4z}\dot{\theta}_3\dot{\theta}_1 s_{1+2+3} + 2\,b_{4z}\dot{\theta}_3\dot{\theta}_2 s_{1+2+3} \\ +2\,b_{4z}\dot{\theta}_1\dot{\theta}_2 s_{1+2+3} + b_{4z}\dot{\theta}_3^2 s_{1+2+3} - c_1\dot{\theta}_1^2 a_{2y} - s_1\ddot{\theta}_1 a_{2y} \\ +s_1\dot{\theta}_1^2 a_{2z} - c_1\ddot{\theta}_1 a_{2z} - b_{4y}\ddot{\theta}_1 s_{1+2+3} - b_{4y}\ddot{\theta}_2 s_{1+2+3} \\ -b_{4y}\dot{\theta}_2^2 c_{1+2+3} - b_{4y}\ddot{\theta}_3 s_{1+2+3} - 2\,b_{4y}\dot{\theta}_3\dot{\theta}_2 c_{1+2+3} - b_{4y}\dot{\theta}_3^2 c_{1+2+3} \\ -2\,b_{4y}\dot{\theta}_3\dot{\theta}_1 c_{1+2+3} - 2\,b_{4y}\dot{\theta}_1\dot{\theta}_2 c_{1+2+3} - b_{4y}\dot{\theta}_1^2 c_{1+2+3} + a_{3z}\dot{\theta}_1^2 s_{1+2} \\ -a_{3z}\ddot{\theta}_1 c_{1+2} + a_{3z}\dot{\theta}_2^2 s_{1+2} - a_{3z}\ddot{\theta}_2 c_{1+2} + 2\,a_{3z}\dot{\theta}_1\dot{\theta}_2 s_{1+2} - b_{4z}\ddot{\theta}_3 c_{1+2+3} \\ +b_{4z}\dot{\theta}_1^2 s_{1+2+3} - b_{4z}\ddot{\theta}_1 c_{1+2+3} - a_{3y}\dot{\theta}_1^2 c_{1+2} - a_{3y}\ddot{\theta}_1 s_{1+2} \\ -a_{3y}\dot{\theta}_2^2 c_{1+2} - a_{3y}\ddot{\theta}_2 s_{1+2} - 2\,a_{3y}\dot{\theta}_1\dot{\theta}_2 c_{1+2} \\ 0 \\ -s_1\dot{\theta}_1^2 b_{2y} + c_1\ddot{\theta}_1 b_{2y} - c_1\dot{\theta}_1^2 b_{2z} - s_1\ddot{\theta}_1 b_{2z} \\ -s_1\dot{\theta}_1^2 a_{2y} + c_1\ddot{\theta}_1 a_{2y} - c_1\dot{\theta}_1^2 a_{2z} - s_1\ddot{\theta}_1 a_{2z} \\ -b_{3y}\dot{\theta}_1^2 s_{1+2} + b_{3y}\ddot{\theta}_1 c_{1+2} - 2\,b_{3y}\dot{\theta}_1\dot{\theta}_2 s_{1+2} - b_{3y}\dot{\theta}_2^2 s_{1+2} \\ +b_{3y}\ddot{\theta}_2 c_{1+2} - b_{3z}\dot{\theta}_1^2 c_{1+2} - b_{3z}\ddot{\theta}_1 s_{1+2} - 2\,b_{3z}\dot{\theta}_1\dot{\theta}_2 c_{1+2} \\ -b_{3z}\dot{\theta}_2^2 c_{1+2} - b_{3z}\ddot{\theta}_2 s_{1+2} \\ -a_{3z}\dot{\theta}_1^2 c_{1+2} - a_{3z}\ddot{\theta}_1 s_{1+2} - a_{3z}\dot{\theta}_2^2 c_{1+2} - a_{3z}\ddot{\theta}_2 s_{1+2} \\ -s_1\dot{\theta}_1^2 a_{2y} + c_1\ddot{\theta}_1 a_{2y} - c_1\dot{\theta}_1^2 a_{2z} - s_1\ddot{\theta}_1 a_{2z} - 2\,a_{3y}\dot{\theta}_1\dot{\theta}_2 s_{1+2} \\ -b_{4y}\dot{\theta}_3^2 s_{1+2+3} - b_{4y}\dot{\theta}_1^2 s_{1+2+3} + b_{4y}\ddot{\theta}_1 c_{1+2+3} - b_{4y}\dot{\theta}_2^2 s_{1+2+3} \\ +b_{4y}\ddot{\theta}_3 c_{1+2+3} + b_{4y}\ddot{\theta}_2 c_{1+2+3} - b_{4z}\ddot{\theta}_3 s_{1+2+3} - 2\,b_{4y}\dot{\theta}_3\dot{\theta}_1 s_{1+2+3} \\ -2\,b_{4y}\dot{\theta}_3\dot{\theta}_2 s_{1+2+3} - 2\,b_{4y}\dot{\theta}_1\dot{\theta}_2 s_{1+2+3} - a_{3y}\dot{\theta}_1^2 s_{1+2} + a_{3y}\ddot{\theta}_1 c_{1+2} \\ -a_{3y}\dot{\theta}_2^2 s_{1+2} + a_{3y}\ddot{\theta}_2 c_{1+2} - 2\,a_{3z}\dot{\theta}_1\dot{\theta}_2 c_{1+2} - b_{4z}\dot{\theta}_2^2 c_{1+2+3} \\ -b_{4z}\ddot{\theta}_2 s_{1+2+3} - b_{4z}\dot{\theta}_1^2 c_{1+2+3} - b_{4z}\ddot{\theta}_1 s_{1+2+3} - b_{4z}\dot{\theta}_3^2 c_{1+2+3} \\ -2\,b_{4z}\dot{\theta}_1\dot{\theta}_2 c_{1+2+3} - 2\,b_{4z}\dot{\theta}_3\dot{\theta}_1 c_{1+2+3} - 2\,b_{4z}\dot{\theta}_3\dot{\theta}_2 c_{1+2+3} \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} \tag{E.7}$$

The kinematic model will be utilized to create the dynamic model.

E.2 Dynamics in SSP

The dynamics in SSP-L are derived by using the Lagrange-d'Alembert equation[Craig, 2005, p. 182]:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = \mathcal{F}_{i,L} \tag{E.8}$$

where:
$\mathcal{L}_L$ is the Lagrangian for SSP-L
q_i is system state i
$\mathcal{F}_{i,L}$ is external force i for SSP-L
The Lagrangian ($\mathcal{L}_L$) is defined as:

$$\mathcal{L}_L = E_{\text{kin},L} - E_{\text{pot},L} \tag{E.9}$$

where:
$E_{\text{kin},L}$ is the kinetic energy of the system in SSP-L
$E_{\text{pot},L}$ is the potential energy of the system in SSP-L
The kinetic energy is calculated using [Craig, 2005, Eq. (6.69) and (6.70), p. 182]:

$$E_{\text{kin},L} = \frac{1}{2}\Bigg(\left(\sum_{i=1}^{N} m_i(\dot{y}_i^2 + \dot{z}_i^2) \right) + \dot{\theta}_1^2 J_{x,2} + (\dot{\theta}_1 + \dot{\theta}_2)^2 J_{x,3} + (\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3)^2 J_{x,4} \Bigg) \tag{E.10}$$

The potential energy is calculated to:

$$E_{\text{pot},L} = \sum_{i=1}^{N} m_i g z_i \tag{E.11}$$

Inserting Equation (E.10) and (E.11) into Equation (E.9) yields:

$$\mathcal{L}_L = \frac{1}{2}\left(\left(\sum_{i=1}^{N} m_i({\dot{y}_i}^2 + {\dot{z}_i}^2 - 2g z_i) \right) + \dot{\theta}_1^2 J_{x,2} + (\dot{\theta}_1 + \dot{\theta}_2)^2 J_{x,3} + (\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3)^2 J_{x,4} \right) \tag{E.12}$$

Partial differentiation of Equation (E.12) with regards to $\vec{q}$ yields:

$$\frac{\partial \mathcal{L}_L}{\partial \vec{q}} = \begin{bmatrix} \frac{\partial \mathcal{L}_L}{\partial y_1} \\ \frac{\partial \mathcal{L}_L}{\partial y_2} \\ \frac{\partial \mathcal{L}_L}{\partial y_3} \\ \frac{\partial \mathcal{L}_L}{\partial y_4} \\ \frac{\partial \mathcal{L}_L}{\partial z_1} \\ \frac{\partial \mathcal{L}_L}{\partial z_2} \\ \frac{\partial \mathcal{L}_L}{\partial z_3} \\ \frac{\partial \mathcal{L}_L}{\partial z_4} \\ \frac{\partial \mathcal{L}_L}{\partial \theta_1} \\ \frac{\partial \mathcal{L}_L}{\partial \theta_2} \\ \frac{\partial \mathcal{L}_L}{\partial \theta_3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ m_1 g \\ m_2 g \\ m_3 g \\ m_4 g \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{E.13}$$

To obtain the second part of Equation (E.8), $\mathcal{L}_L$ is partially differentiated with regards to $\dot{\vec{q}}$:

$$\frac{\partial \mathcal{L}_L}{\partial \dot{\vec{q}}} = \begin{bmatrix} \frac{\partial \mathcal{L}_L}{\partial \dot{y}_1} \\ \frac{\partial \mathcal{L}_L}{\partial \dot{y}_2} \\ \frac{\partial \mathcal{L}_L}{\partial \dot{y}_3} \\ \frac{\partial \mathcal{L}_L}{\partial \dot{y}_4} \\ \frac{\partial \mathcal{L}_L}{\partial \dot{z}_1} \\ \frac{\partial \mathcal{L}_L}{\partial \dot{z}_2} \\ \frac{\partial \mathcal{L}_L}{\partial \dot{z}_3} \\ \frac{\partial \mathcal{L}_L}{\partial \dot{z}_4} \\ \frac{\partial \mathcal{L}_L}{\partial \dot{\theta}_1} \\ \frac{\partial \mathcal{L}_L}{\partial \dot{\theta}_2} \\ \frac{\partial \mathcal{L}_L}{\partial \dot{\theta}_3} \end{bmatrix} = \begin{bmatrix} m_1 \dot{y}_1 \\ m_2 \dot{y}_2 \\ m_3 \dot{y}_3 \\ m_4 \dot{y}_4 \\ m_1 \dot{z}_1 \\ m_2 \dot{z}_2 \\ m_3 \dot{z}_3 \\ m_4 \dot{z}_4 \\ \dot{\theta}_1 J_{x,2} + (\dot{\theta}_1 + \dot{\theta}_2) J_{x,3} + (\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3) J_{x,4} \\ (\dot{\theta}_1 + \dot{\theta}_2) J_{x,3} + (\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3) J_{x,4} \\ (\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3) J_{x,4} \end{bmatrix} \tag{E.14}$$

Differentiating Equation (E.14) with regards to time yields:

$$\frac{d}{dt}\frac{\partial \mathcal{L}_L}{\partial \dot{\vec{q}}} = \begin{bmatrix} m_1 \ddot{y}_1 \\ m_2 \ddot{y}_2 \\ m_3 \ddot{y}_3 \\ m_4 \ddot{y}_4 \\ m_1 \ddot{z}_1 \\ m_2 \ddot{z}_2 \\ m_3 \ddot{z}_3 \\ m_4 \ddot{z}_4 \\ \ddot{\theta}_1 J_{x,2} + (\ddot{\theta}_1 + \ddot{\theta}_2) J_{x,3} + (\ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3) J_{x,4} \\ (\ddot{\theta}_1 + \ddot{\theta}_2) J_{x,3} + (\ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3) J_{x,4} \\ (\ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3) J_{x,4} \end{bmatrix} \tag{E.15}$$

To map the Lagrangian to the joints, the Jacobian is used[Craig, 2005, p. 186]:

$$\boldsymbol{J}_{F,L}(\vec{\theta}) = \frac{\partial \vec{q}}{\partial \vec{\theta}} \tag{E.16}$$

$$\boldsymbol{J}_{F,L}(\vec{\theta}) = \begin{bmatrix} \frac{\partial q_1}{\partial \theta_1} & \frac{\partial q_1}{\partial \theta_2} & \frac{\partial q_1}{\partial \theta_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial q_{11}}{\partial \theta_1} & \frac{\partial q_{11}}{\partial \theta_2} & \frac{\partial q_{11}}{\partial \theta_3} \end{bmatrix} \tag{E.17}$$

Due to the size of $\boldsymbol{J}_{F,L}(\vec{\theta})$, it is divided into subsets:

$$\begin{bmatrix} \frac{\partial q_1}{\partial \theta_1} \\ \vdots \\ \frac{\partial q_{11}}{\partial \theta_1} \end{bmatrix} = \begin{bmatrix} 0 \\ -s_1 b_{2y} - c_1 b_{2z} \\ -s_1 a_{2y} - c_1 a_{2z} - s_1 c_2 b_{3y} + c_1 s_2 b_{3z} \\ -s_1 a_{2y} - c_1 a_{2z} - s_1 c_2 a_{3y} + c_1 s_2 a_{3z} - s_1 c_2 c_3 b_{4y} - c_1 s_2 s_3 b_{4z} \\ 0 \\ c_1 b_{2y} - s_1 b_{2z} \\ c_1 a_{2y} - s_1 a_{2z} + c_1 s_2 b_{3y} - s_1 c_2 b_{3z} \\ c_1 a_{2y} - s_1 a_{2z} + c_1 s_2 a_{3y} - s_1 c_2 a_{3z} + c_1 s_2 s_3 b_{4y} - s_1 c_2 c_3 b_{4z} \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{E.18}$$

$$\begin{bmatrix} \frac{\partial q_1}{\partial \theta_2} \\ \vdots \\ \frac{\partial q_{11}}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -c_1 s_2 b_{3y} + s_1 c_2 b_{3z} \\ -c_1 s_2 a_{3y} + s_1 c_2 a_{3z} - c_1 s_2 s_3 b_{4y} - s_1 c_2 s_3 b_{4z} \\ 0 \\ 0 \\ s_1 c_2 b_{3y} - c_1 s_2 b_{3z} \\ s_1 c_2 a_{3y} - c_1 s_2 a_{3z} + s_1 c_2 s_3 b_{4y} - c_1 s_2 c_3 b_{4z} \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{E.19}$$

$$\begin{bmatrix} \frac{\partial q_1}{\partial \theta_3} \\ \vdots \\ \frac{\partial q_{11}}{\partial \theta_3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -c_1 c_2 s_3 b_{4y} - s_1 s_2 c_3 b_{4z} \\ 0 \\ 0 \\ 0 \\ s_1 s_2 c_3 b_{4y} - c_1 c_2 s_3 b_{4z} \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{E.20}$$

The Jacobian is applied to the Lagrange-d'Alembert equation by using Equation (E.21):

$$\vec{\tau}_L = \boldsymbol{J}_{F,L}^{\mathrm{T}}(\vec{\theta})\vec{\mathcal{F}}_L = \left(\frac{\partial \vec{q}}{\partial \vec{\theta}}\right)^{\mathrm{T}} \left(\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\vec{q}}}\right) - \frac{\partial \mathcal{L}}{\partial \vec{q}}\right) \tag{E.21}$$

where:
$\vec{\tau}$ is the torque excerted on the links

This yields the resulting moment ($\vec{\tau}$) on each of the joints:

$$\vec{\tau}=\begin{pmatrix} (s_1 b_{2y}+c_1 b_{2z})m_1\ddot{y}_2 \\ +(s_1 a_{2y}+c_1 a_{2z}+s_1 c_2 b_{3y}+c_1 s_2 b_{3z})m_3\ddot{y}_3 \\ +(s_1 a_{2y}+c_1 a_{2z}+s_1 c_2 a_{3y}+c_1 s_2 a_{3z} \\ +s_1 c_2 c_3 b_{4y}+c_1 s_2 s_3 b_{4z})m_4\ddot{y}_4 \\ -(c_1 b_{2y}-s_1 b_{2z})(m_2 g+m_2\ddot{z}_2) \\ -(c_1 a_{2y}-s_1 a_{2z}+c_1 s_2 b_{3y}-s_1 c_2 b_{3z})(m_3 g+m_3\ddot{z}_3) \\ -(c_1 a_{2y}-s_1 a_{2z}+c_1 s_2 a_{3y}-s_1 c_2 a_{3z} \\ +c_1 s_2 s_3 b_{4y}-s_1 c_2 c_3 b_{4z})(m_4 g+m_4\ddot{z}_4) \\ -(J_{x,2}+J_{x,3}+J_{x,4})\ddot{\theta}_1-(J_{x,3}+J_{x,4})\ddot{\theta}_2-J_{x,4}\ddot{\theta}_3 \\ \\ (-c_1 s_2 b_{3y}+s_1 c_2 b_{3z})(-m_3\ddot{y}_3) \\ +(-c_1 s_2 a_{3y}+s_1 c_2 a_{3z}-c_1 s_2 c_3 b_{4y}-s_1 c_2 s_3 b_{4z})(-m_4\ddot{y}_4) \\ +(s_1 c_2 b_{3y}-c_1 s_2 b_{3z})(-m_3 g-m_3\ddot{z}_3) \\ -m_4(g+\ddot{z}_4)(s_1 c_2 a_{3y}-c_1 s_2 a_{3z}+s_1 c_2 s_3 b_{4y}-c_1 s_2 c_3 b_{4z}) \\ -(\ddot{\theta}_1+\ddot{\theta}_2)J_{x,3}-(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)J_{x,4} \\ \\ -m_4\ddot{y}_4(-c_1 c_2 s_3 b_{4y}-s_1 s_2 c_3 b_{4z}) \\ -m_4(g+\ddot{z}_4)(s_1 s_2 c_3 b_{4z}-c_1 c_2 s_3 b_{4z}) \\ -(\ddot{\theta}_1+\ddot{\theta}_2+\ddot{\theta}_3)J_{x,4} \end{pmatrix} \tag{E.22}$$

The model of Strider in SSP-L is given by combining Equation (E.7) with Equation (E.22). The model of Strider in SSP-R can be calculated using the same principle as was used in calculating the model for SSP-L. The final part of the motivating example deals with combining SSP-L and SSP-R into the model for the Double Support Phase. By inspection of Equation (E.7) and Equation (E.22), it is seen that the equation can be subdivided into the parts seen in

$$\vec{\tau} = \boldsymbol{M}(\vec{\theta})\ddot{\vec{\theta}} + \boldsymbol{V}(\theta, \dot{\vec{\theta}}) + G(\vec{\theta}) \tag{E.23}$$

where:

$\boldsymbol{M}(\vec{\theta})$	is the mass matrix.
$\vec{V}(\theta, \dot{\vec{\theta}})$	is a vector of centrifugal and Coriolis terms.
$\vec{G}(\vec{\theta})$	is a vector of gravity terms.

By isolating $\ddot{\vec{\theta}}$ using Equation (E.24), the model can be implemented in simulink (see Figure E.3).

$$\ddot{\vec{\theta}} = \boldsymbol{M}^{-1}(\vec{\theta})\left(\vec{\tau} - \boldsymbol{V}(\theta, \dot{\vec{\theta}}) - G(\vec{\theta})\right) \tag{E.24}$$

Figure E.3: Simulink implementation of the model for Strider.

E.3 Dynamics of Strider in DSP

The dynamics of Strider are computed as a combination of SSP-L and SSP-R, by using Equation (E.25):

$$\vec{\tau}_{\text{DSP}} = \rho\vec{\tau}_L + (1-\rho)\vec{\tau}_R \tag{E.25}$$

where:

$\vec{\tau}_{\text{DSP}}$	is the torque of the individual links of Strider in DSP
$\vec{\tau}_L$	is the torque of the individual links of Strider in SSP-L
$\vec{\tau}_R$	is the torque of the individual links of Strider in SSP-R
ρ	is the weighting between the SSP-L and SSP-R

ρ is calculated by calculating the distance from the origo of each SSP to the ZMP (ZMP is calculated using Equation (2.8). Due to the fact that the x-direction is ignored in this example, ZMP is a scalar):

$$\rho_1 = \frac{y_{\text{ZMP}}}{\Delta y} \tag{E.26}$$

$$\rho = \begin{cases} 1 & \rho_1 \geq 1 \\ \rho_1 & 0 < \rho_1 < 1 \\ 0 & \rho_1 \leq 0 \end{cases} \tag{E.27}$$

where:

y_{ZMP}	is the y coordinate of the ZMP
Δy	is the distance between the legs

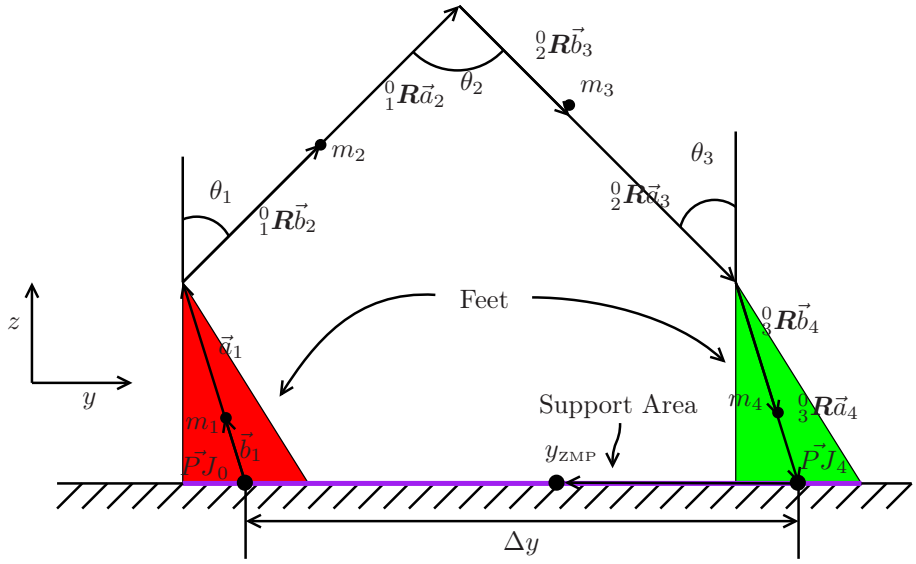


Figure E.4: Strider in DSP-L.

Appendix F

Dynamic Gait Trajectories

To avoid replication in the main report, the dynamic gait trajectories is described in this appendix. Even though the dynamic walk trajectories cannot be simulated correct due to a inexact ZMP estimation, then the Matlab functions, that have to be used in order to obtain the best trajectory have been developed, such that future groups can use it. The time line that is used in the following sections are described in Equation (6.8).

F.1 Foot Trajectory for Dynamic Gait

The walking pattern is given for a dynamic gait trajectory in Table 6.2 on page 103. Figure F.1 on the next page is a step for the right foot in the duration T_{step}, all lengths at the figure is described during this section.

In order to be able to create the trajectory for the foot, a time line has to be defined as in Equation (6.8). The second part of T_{cycle} the right foot is on the ground. The time line for the dynamic gait trajectory is only equal to the static gait time line symbolic, i.e. distance and speed for dynamical and statically walking are very different. The distance between the feet is set to 0.28, which is the same as the hip width. Movement of the foot is divided in to three steps; first the rotation of the foot should be defined. This is done in Equation (F.1)

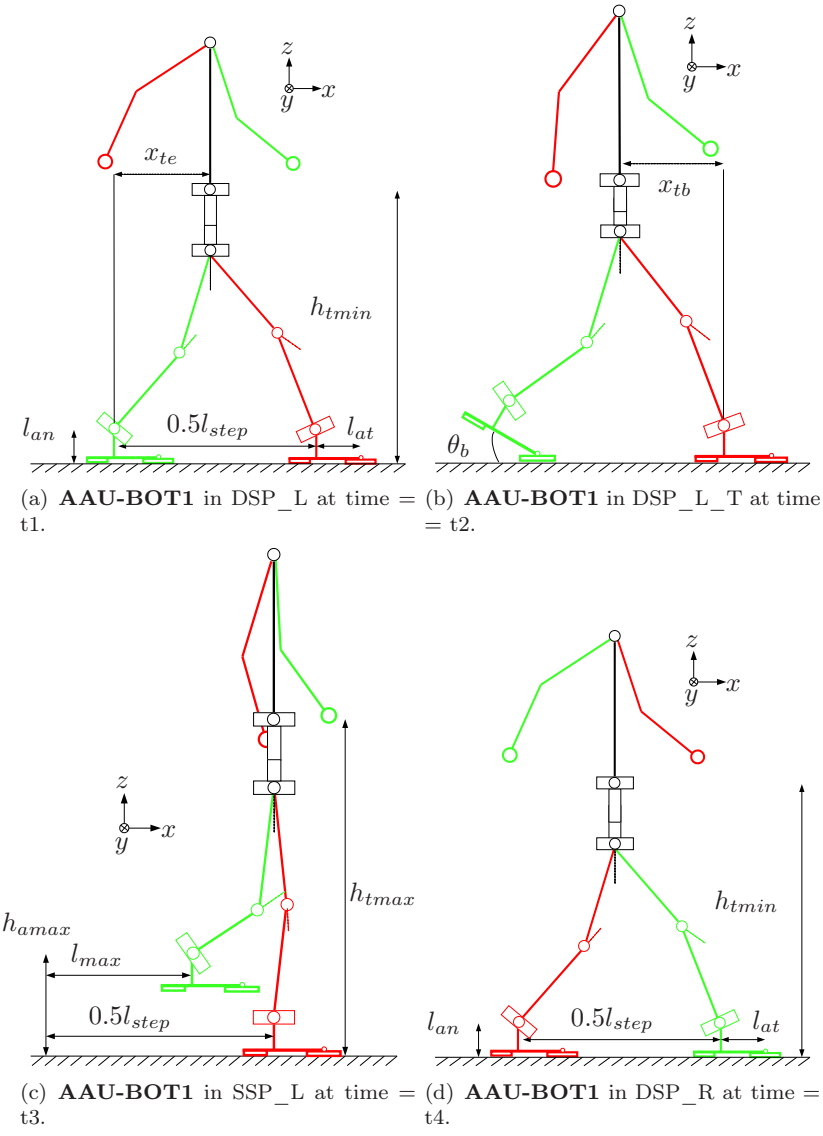
$$\begin{array}{lll} \theta_a(t) & = 0 & , t = t1 \\ \theta_a(t) & = \theta_b & , t = t2 \\ \theta_a(t) & = 0 & , t = t3 \text{ and } t4 \end{array} \tag{F.1}$$

where:

θ_a	is the rotation of the toe joint during T_{step}.
θ_b	is the rotation of the toe joint during toe off.

Secondly the transverse motion of the foot must be defined, this is done in Equation (F.2).

$$\begin{array}{lll} x_a(t) & = -l_{at} - 0.5l_{step} & , t = t1 \\ x_a(t) & = -l_{at} - 0.5l_{step} + l_{an}sin(\theta_b) + l_{at}(1 - cos(\theta_b)) & , t = t2 \\ x_a(t) & = -l_{at} - 0.5l_{step} + l_{max} & , t = t3 \\ x_a(t) & = -l_{at} - l_{step} & , t = t4 \end{array} \tag{F.2}$$



(a) **AAU-BOT1** in DSP_L at time = t1.

(b) **AAU-BOT1** in DSP_L_T at time = t2.

(c) **AAU-BOT1** in SSP_L at time = t3.

(d) **AAU-BOT1** in DSP_R at time = t4.

Figure F.1: Sketch of a step for the right foot during T_{step}.

where:

$x_a(t)$	is the transverse movement i the x-direction of the ankle during T_{step}.
l_{an}	is the height of the ankle from the ground.
l_{at}	is the transverse distance between the ankle and toe.
θ_b	is the rotation of the toe joint during toe off.

Third and last the vertical movement must be defined as in Equation (F.3).

$$\begin{array}{lll} z_a(t) & = l_{an} & ,t = t1 \\ z_a(t) & = l_{at}sin(\theta_b) + l_{an}cos(\theta_b) & ,t = t2 \\ z_a(t) & = h_{amax} & ,t = t3 \\ z_a(t) & = l_{an} & ,t = t4 \end{array} \tag{F.3}$$

where:

$z_a(t)$	is vertical movement of the ankle during T_{step}.
l_{an}	is the height of the ankle from the ground.
l_{at}	is the transverse distance between the ankle and toe.
h_{amax}	is maximum ankle height during T_{step}.
θ_b	is the rotation of the toe joint during toe off.

In order to achieve dynamic gait the constraints in Equation (F.4) must be applied. All rotational, vertical and transverse movement, and the constraints can be used to derive a different 3^{rd} order spline interpolation function as described in Section 6.3.2 on page 106.

$$\begin{array}{ll} \dot{\theta}_b(t1) & = 0 \\ \dot{\theta}_b(t2) & = 0 \\ \dot{x}_a(t1) & = 0 \\ \dot{x}_a(t2) & = 0 \\ \dot{z}_a(t1) & = 0 \\ \dot{z}_a(t2) & = 0 \\ \dot{z}_a(t3) & = 0 \\ \dot{z}_a(t4) & = 0 \end{array} \tag{F.4}$$

F.2 Torso Trajectory for Dynamic Gait

The value of the used symbols in the following equations can be seen in Table 6.2 on page 103. The constraints for the vertical movement of the torso can be found in Equation (F.5) and in Table F.6.

$$\begin{array}{lll} z_t(t) & = h_{tmin} & ,t = t1 \\ z_t(t) & = h_{tmax} & ,t = t3 \\ z_t(t) & = h_{amin} & ,t = t4 \end{array} \tag{F.5}$$

where:

$z_t(t)$	is vertical movement of the torso during T_{step}.
h_{tmax}	is the maximum height of the torso.
h_{tmin}	is the minimum height of the torso.

$$\begin{array}{ll} \dot{z}_t(t3) & = 0 \\ \dot{z}_t(t4) & = 0 \end{array} \tag{F.6}$$

The normal rotation in the pelvis is [Vaughan et al., 1992] ±18 degrees for a normal male. To ensure that the dynamic walking results in maximal stability, severally torso rotation trajectory have been simulated. The constraints for the pelvis rotation can be seen in Equation (F.7) and Equation (F.8).

$$\begin{array}{rll} -4.5^\circ & \leq \theta_{pelvis}(t) \leq -18^\circ & ,t = t1 \\ 4.5^\circ & \leq \theta_{pelvis}(t) \leq 18^\circ & ,t = t4 \end{array} \tag{F.7}$$

$$\begin{array}{ll} \dot{\theta}_{pelvis}(t1) & = 0 \\ \dot{\theta}_{pelvis}(t4) & = 0 \end{array} \tag{F.8}$$

The movement of the torso in the transverse plane quit different during dynamic gait, since it should have a smooth sine trajectory, and it should move forward with a constant speed in the start and in the end of T_{step} to complete the cyclic motion. Equation (F.9) and Equation (F.10) describes the constraint for the torso's forward motion. The parameters used in the equations can be seen in Figure F.1 on page 204.

$$\begin{array}{lll} x_t(t) & = -l_{at} - 0.5l_{step} + x_{te} & ,t = t1 \\ x_t(t) & = -l_{at} - x_{tb} & t =, t2 \\ x_t(t) & = -l_{at} - x_{te} & t =, t4 \end{array} \tag{F.9}$$

where:

$x_t(t)$	is movement of the torso in the x-direction during T_{step}.
l_{at}	is the transverse distance between the ankle and toe.
x_{tb}	is the length from the left ankle to the torso during toe off phase.
x_{te}	is the length from the right ankle to the torso at time = t4.

$$\begin{array}{ll} \dot{x}_t(t2) & = V_{speed} \\ \dot{x}_t(t4) & = V_{speed} \end{array} \tag{F.10}$$

To be able to generate a trajectory with high stability as described in Section 6.3.1 on page 102, small variations for x_{tb} and x_{te} have been simulated. The interval of the variations has been found via simulation. The variations are listed in Equation (F.11)

$$\begin{array}{l} 0.2l_{step} \leq x_{tb} \leq 0.8l_{step} \\ 0.2l_{step} \leq x_{te} \leq 0.6l_{step} \end{array} \tag{F.11}$$

In order to obtain a ZMP that is within the support polygon during dynamic gait, constraints in the movement in the y-direction must be as Equation (F.12) and Equation (F.13). The parameters used in the equations can be seen in Figure 6.4 on page 111.

$$\begin{array}{lll} y_t(t) & = -y_{tmid} & ,t = t1 \\ y_t(t) & = -y_{tmid} + y_{tmin} & ,t = t3 \\ y_t(t) & = -y_{tmid} & ,t = t4 \end{array} \tag{F.12}$$

where:

$y_t(t)$	is movement of the torso in the y-direction during T_{step}.
y_{mid}	is distance from center of the pelvis to one hip.
y_{tmin}	is the length in the y-axis from the torso to the right foot.

$$\begin{aligned} \dot{y}_t(t1) &= 0 \\ \dot{y}_t(t3) &= 0 \\ \dot{y}_t(t4) &= 0 \end{aligned} \tag{F.13}$$

In order to obtain a good stability, y_{tmax} has been tested in an interval given in Equation (F.14).

$$0 \leq y_{tmin} \leq 0.7 y_{mid} \tag{F.14}$$

Appendix G

Alternative FTS DAQ

This Appendix describes two alternative solutions to measure the signals from the Force Torque Sensor located in each foot. In the beginning of the project an analog strain gauge data acquisition (DAQ) system seemed appropriate, but this solution was discarded due to the fact that it would take too long time [Bisgaard, 2007a] to develop and test a FTS DAQ system.

Secondly a digital solution was examined and it looked very promising, but unfortunately the product could not meet the requirements that was listed in the product sheet (the company had made a creative product description) and was therefore also discarded.

Before both solution were discarded they were designed and developed, and the result of that is therefore described here.

G.1 Analog FTS DAQ

Since it was decided to use single supply, the following considerations must be taken into account:

- Signal swing limited, therefore more sensitive to errors caused by offset voltage, bias current, finite open-loop gain, noise, etc.
- More likely to have noisy power supply because of sharing with digital circuits
- Rail-to-rail op amps needed to maximize signal swings.

For the highest precision and performance, the three op amp instrumentation amplifier (in-amp) topology is the optimum for bridge and other offset transducer applications where high accuracy and low nonlinearity is required. While there are many good precision single supply amplifiers (some rail-to-rail), the highest performance instrumentation amplifiers are still specified for dual supply operation according to [Garcia, 2000]. To be able to get the performance of a dual supply instrumentation amplifier, an AD620 dual supply instrumentation amplifier is chosen and wired so it is possible to use it in combination with the rail to rail op-amp, AD822, and thereby use 2.5 V as reference instead of GND. Two different circuits are made since there are 2 different strain gauge types each full bridges. Figure G.1 shows the simplified diagram of the FTS circuit. The strain gauge circuits have the following properties listed below, which each will be discussed.

- 3 op amp instrumentation amplifier.
  Choosing this type of amplifier ensures high performance, low non-linearity and it

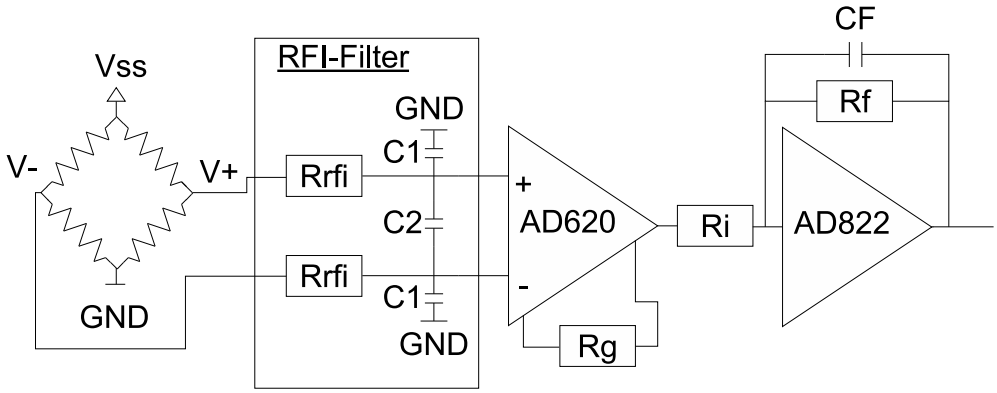


Figure G.1: Simplified FTS circuit.

is good at rejecting common mode noise according to [Garcia, 2000] and [Morris, 2005, p. 88].

- Radio Frequency interference (RFI) filter in the input
  To avoid RFI, a RFI filter has been implemented in the differential input signal, since strong RFI is sometimes rectified inside the in-amp and appears as DC-offset on the output of the in-amp (3 amp instrumentation amplifier), and thereby reduces the amplified common mode rejection. The RFI filter is designed via Analog Devices approach [Kitchin et al., 2003, 3]. According to Analogs design rules, R_{rfi} should be between 2-10k Ω, the differential bandwidth frequency should be 10 times larger than the sample frequency. C1 should be 10 % of the value of C2 or smaller. R_{rfi} is chosen to 2 kΩ, the band width should be 10 kHz, since [Kitchin et al., 2003, 3] recommend to have a band width 10 times higher than the measured signal. Using Equation (G.1) C1 and C2 can be found. C1 is calculated to 0.378 nF and C2 is calculated to 3.78 nF.

$$BW_{\text{Diff}} = \frac{1}{2 \cdot \pi \cdot R_{rfi} \cdot (2 \cdot C_2 + C_1)} \tag{G.1}$$

- Active output filter, Low impedance reference for the AD620 and Rail to rail output op amp.
  To ensure that the noise do not affect the final analog output a 10 kHz low pass filter has been designed via the Equation (G.2).

$$BW_{\text{out}} = \frac{1}{2 \cdot \pi \cdot R_f \cdot C_F} \tag{G.2}$$

  The AD620 requires low reference impedance this is done by using an AD822 amplifier as a buffer. The AD822 amplifier is a rail to rail op amp, and a dual package AD822 is used to buffer and as final output amplifier/filter.

Bridge Supply

According to [Morris, 2005, 250] the maximum current should be between 5-50 mA in a full bridge circuit. Therefore the supply voltages is chosen to 5 V and 3.3 V to the 350

Ω bridge and for the 120 Ω respectively. This gives the following current at 14.089 mA and 27.05 mA. The voltage at 3.3 V is ensured by a TPS7433D voltage regulator and the 5 V is ensured by a 78L05 from Maxim-IC.

Required gain and A/D requirement

Gain in the FTS-circuit must be calculated such that the absolute maximum strain gauge signal is amplified to 2.5 V abs. The signals have to swing around 2.5 V, where 2.5 V is equal to no impact to the full bridge. The Gain factors for the strain gauge are 2.09 and 2.10 for the 350 Ω and 120 Ω strain gauges respectively. The maximum stretch of each strain gauge is guessed to 190 μm. Equation (G.3) is the maximum resistance changes [Haffgaard, 2005]. In the following equations the measured resistant value is used see Tablefig:jk:FTSSGsketch

$$\frac{\Delta R}{R} = K \cdot \frac{\Delta L}{L} \Rightarrow \Delta R_{max} = K \cdot R \cdot \frac{\Delta L}{L} \tag{G.3}$$

$$\Delta R_{\text{max },350} = 2.09 \cdot 355 \cdot \frac{190 \cdot 10^{-3}}{7.6} = \pm 18.5488\Omega \tag{G.4}$$

$$\Delta R_{\text{max },120} = 2.10 \cdot 122 \cdot \frac{190 \cdot 10^{-3}}{8.8} = \pm 5.5316\Omega \tag{G.5}$$

Via the Equation (G.6) the maximum output from the bridge can be calculated.

$$V_{\text{Bridge, max, Out}} = \left(\frac{R + \Delta R_{max}}{2 \cdot R + \Delta R R_{max}} + \frac{R}{2 \cdot R} \right) \cdot V_{ss} \tag{G.6}$$

$$V_{\text{Bridge, 120, max, Out}} = \left| \left(\frac{122 + 5.5316}{2 \cdot 122 + 5.5316} - \frac{122}{2 \cdot 122} \right) \right| \cdot 3.3 = 36.577\text{mV} \tag{G.7}$$

$$V_{\text{Bridge, 350, max, Out}} = \left| \left(\frac{355 + 18.5486}{2 \cdot 355 + 18.5486} - \frac{355}{2 \cdot 355} \right) \right| \cdot 5 = 63.649\text{mV} \tag{G.8}$$

Since the AD620 is an in-amp and it has the best noise properties most of the gain is amplified in this. The AD822 amplifies the signal 2 times, so the signal becomes rail to rail(0-5 V). Via Equation (G.9) and (G.9) the total gain can be found.

$$G_{\text{tot, 120}} = \frac{V_{ss}}{V_{\text{Bridge, 120, max, Out}}} = \frac{5}{0.036577} = 136.7 \approx 137 \tag{G.9}$$

$$G_{\text{tot, 350}} = \frac{V_{ss}}{V_{\text{Bridge, 350, max,Out}}} = \frac{5}{0.063649} = 78.6 \approx 79 \tag{G.10}$$

The gain in the AD620 in the 120 Ω bridge circuit must be 68.5 and 39.5 in the 350 Ω bridge circuit. Via Equation (G.11) the resistor R_g can be found for both circuits.

$$Gain_{\text{tot}} = \left(\frac{49.9k}{R_g} + 1 \right) \cdot \left(\frac{R_f}{R_i} \right) \tag{G.11}$$

Table G.1 gives the value of the main components. Beside the main components, several noise decoupling capacitors have been placed in the circuit.

The strain gauge signals must be converted in an AD-converter. Group [Pedersen et al., 2007] used a Spider8 16 bit ADC strain gauge amplifier to measure the strain gauges signals. It is evaluated that measurement circuit must have the same resolution, which gives a maximum resolution of 0.0305 N. The final print circuit board (PCB) can be seen in Figure G.2

R_g = Gain resistor connected to the AD620.
R_f = Resistor in the output filter.
R_i = Input resistor to the AD822.

Table G.1: Essential components in the 120 Ω bridge circuit and the 350 Ω bridge

Name	**Value 120 Ω**	**Value 350 Ω**	**Unit**
R_{rfi}	2k	2k	Ω
R_i	25k	25k	Ω
R_f	50k	50k	Ω
R_g	740	1296	Ω
C_1	0.378	0.378	nF
C_2	3.78	3.78	nF
C_F	0.318	0.318	nF

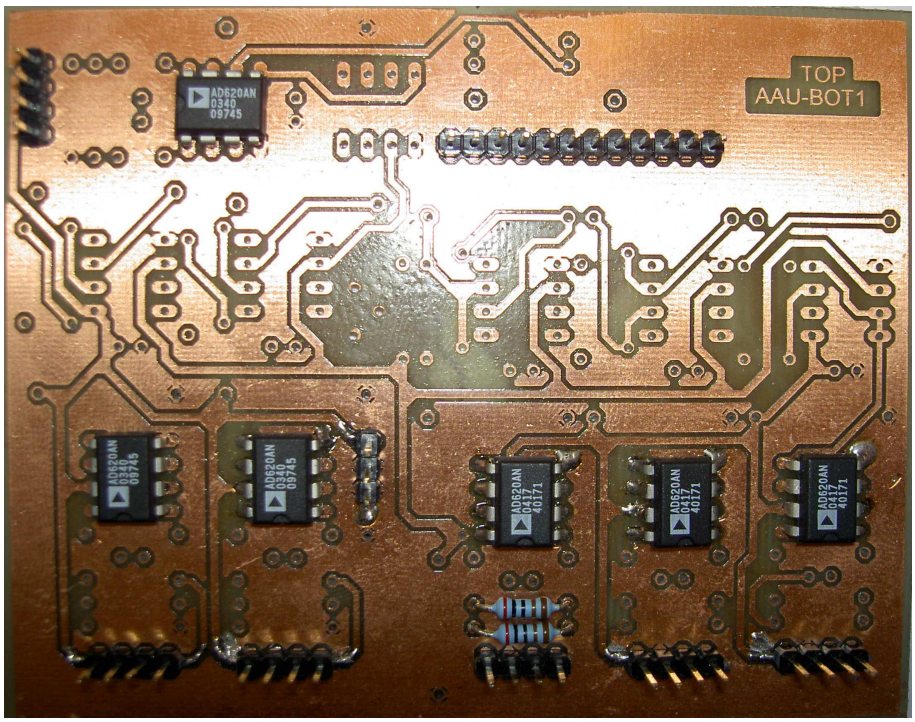


Figure G.2: Final PCB.

G.2 Alternative Digital FTS DAQ

The FTS consist of 6 full-bridge strain gauges. This gives 12 analog signals that can be either feed back to the on-board computer directly and AD converted, or it can be converted through an ADC at the strain gauge and transmitted on a bus. A long wire with analog signals from the strain gauges to the on-board computer will with a high probability give a lot of noise. Therefore it is chosen to AD convert the signals and transmit it on a bus to the on-board computer. A company called Mantracourt is developing transducers for strain gauges. The unit called DSC contains an amplifier, filter and an AD Converter. The DSC comes with different interfaces: CANopen, MODBUS, ASCII, MantraBUS. The pro and cons have to be found for the different bus types before a certain type can be selected.

- CANopen
  - Same bus as used with the EPOS 70/10
  - CRC code
  - Max sampling rate of 200Hz
- MODBUS (RS485)
  - Need a new protocol stack (MODBUS protocol)
  - CRC code
  - Max sampling rate of 500Hz
- ASCII protocol (RS485)
  - Simple to use (terminal program)
  - No CRC code
  - Max sampling rate of 500Hz
- MantraBUS protocol (RS485)
  - Mantracourts own protocol, bound to their windows SW
  - CRC code
  - Max sampling rate of 500Hz

All the protocols runs on a serial bus and it is possible to have multiple nodes on the same bus. The first protocol is the CANopen protocol and the good thing about this is that the EPOS amplifier also runs with the CANopen protocol. In this way the knowledge obtained by implementing the EPOS, can be reused when the DSC for the strain gauges is implemented. Unfortunately the DSC with CAN is only able to run with a sampling rate of 200Hz.

The DSC with other protocol versions can sample with a sampling rate of 500 Hz. The first of these versions is the ASCII protocol, this protocol is straightforward to use, but it features no CRC code. Another version is the MantraBUS which runs with the SW version from Mantracourt (the manufacture) and is optimized for their Windows API software. The MODBUS is mostly used by the industry and is comparable with the CANopen protocol, as it has CRC and it own protocol stack.

Appendix H

Calibration and Test of the FTS's and Amplifiers

When the Force Torque Sensor (FTS) is used to measure strain gauges, small changes in resistance are measured. The voltages are sampled and amplified in 6 strain gauge amplifiers. The strain gauges and amplifiers have different gains and different offsets. In order to get forces and torques that can be used for a ZMP estimation, a calibration is needed. [Flay and Vuletich, 1995] and [Löffler et al., 2004b] have used the least square method to calibrate FTS's and [Pedersen et al., 2007] used this method as well to calibrate the first generation of the FTS to **AAU-BOT1**. The second FTS version has been optimized, such that the weight is reduced. Equation (H.1) describes how the calibration matrix is found via the least square method, which is a method that uses the orthogonality principle [Lay, 2003].

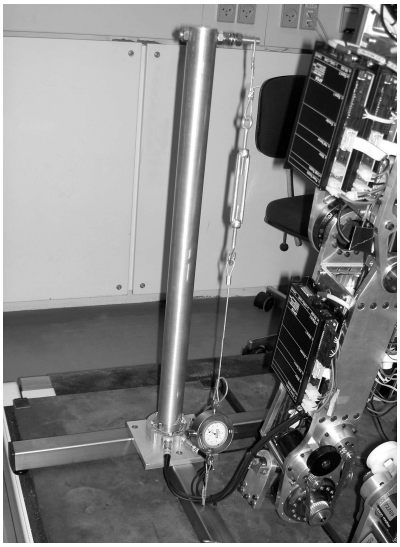
$$\boldsymbol{C} = \boldsymbol{F}\boldsymbol{V}^T(\boldsymbol{V}\boldsymbol{V}^T)^{-1} \tag{H.1}$$

C is the calibration matrix, which has to be multiplied with the measured strains in order to get forces and torques.
F is the actual forces and torques caused by the calibration loads.
V is the measured strains

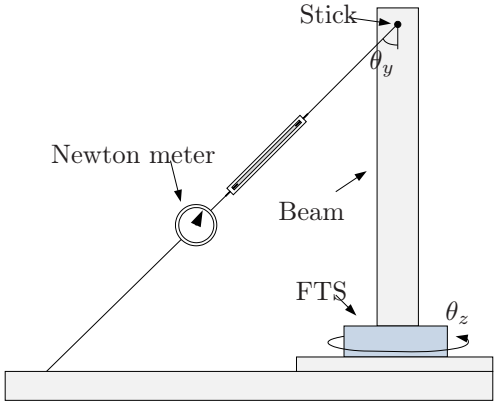
H.1 Calibration Test Method

In order to obtain a good calibration of the FTS it must be excitated as much as possible. To excitate the FTS a calibration test rig has been developed in collaboration with PhD. student Mads Soelver Svendsen. The calibration have been done twice, because the original test rig did not excitate the FTS around the Z-axis, to accommodate this problem a stick is mounted as in Figure H.1 on the next page. If the wire is not attached to the stick, the calibration will give inaccurate results, because M_z is not measured.

Calibration with the least square method uses data from 6 different loading situations (M1-M6). At each loading situation 4 measurements have been measured with 4 different weights. In Table H.1 on the following page the different loading situations are listed. Each measurement used to the calibration is the mean value of 2001 measurements. By using a Tait-Bryan matrix [Samin, 2005] the force can be decomposed in F_x, F_y, F_z and



(a) Picture of the FTS calibration test rig.



(b) Sketch of the calibration test rig.

the torque M_x, M_y, M_z are found by multipling a skew matrix by the forces [Samin, 2005].

Table H.1: Calibration loading situation.

Loading situation	θ_x**[degrees]**	θ_y**[degrees]**	θ_z**[degrees]**	**Loads [kg]**
M1	0	15.9688	40	5,15,25,35
M2	0	15.9688	0	5,15,25,35
M3	0	15.9688	-40	5,15,25,35
M4	0	30.1204	40	5,15,25,35
M5	0	30.1204	0	5,15,25,35
M6	0	30.1204	-40	5,15,25,35

To test the result of the calibration, 4 measurements have been tested. In Table H.2 the loads from the test have been listed. The result of the calibration is discussed in Section H.2 on the facing page.

Table H.2: Loads to test the final calibration matrix.

Loading situation	θ_x**[degrees]**	θ_y**[degrees]**	θ_z**[degrees]**	**Loads [kg]**
M6	0	30.1204	-40	10,20,30,40

H.2 Results of Calibration

Table H.3 shows the result of the calibration, notice that only one sensor is calibrated. Due to one of the amplifiers in the left leg is defect, which means that the left FTS cannot be calibrated. The amplifier broke down after a short circuit in one of the EPOS's which uses the same power supply as the amplifiers. The amplifier must be repaired before a calibration can be accomplished, but because of delivering time it is not possible to get a new amplifier before the handover of this thesis. However it has be tested that the principle works since the right and left FTS's were calibrated with the original calibration test rig, but these measurements were discarded since the torque M_z was not excitated, which resulted in high gain errors. Table H.3 is the result of the new calibration and it shows a high RMS error in the force F_z, i.e. 16.693%. Figure H.2(c) shows the error is an offset error. The offset error in the F_z direction can be corrected by adding 50N to F_z. When the offset is added, the RMS error is reduced to 2.103%. After examine the possibilities of errors, it has been concluded that the test rig is designed wrong. The beam was mounted in the center hole of the FTS, which gave a stiffness of FTS in the vertical direction and the strains were reduced. In order to correct this problem, the beam has to be redesigned, such that it is possible to mount the beam in the same way as the FTS is mounted on the ankle. A picture of the center bolt that caused the stiffness can be seen in Figurejk:fig:FTSCenterBolt. In section H.3 future proposals to improvements of the calibration is described.

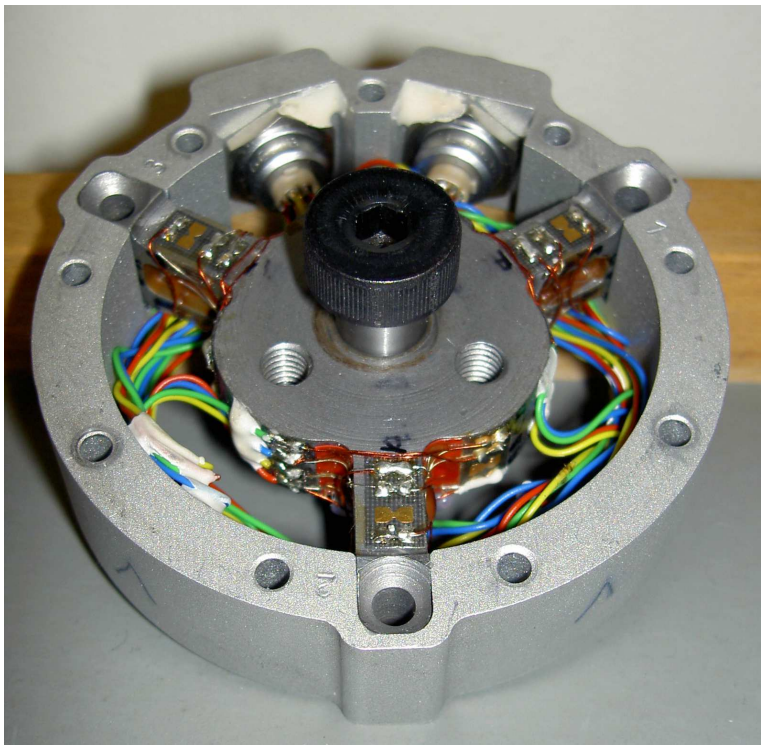


Figure H.1: The FTS seen from below where the center bolt is mounted.

Beside from the high RMS error in F_z, the maximal RMS error is 2.403%. [Flay and Vuletich, 1995] describes that a maximum RMS error at 6% RMS error is normal, and an average

RMS error around 2-4% is normal, that is why the results of the FTS amplifier and the calibration is concluded to be a succes, at least for the right FTS. The result of the calibration can be seen in Figure H.2 on the next page.

Table H.3: Result of the RMS error at the right FTS and amplifier.

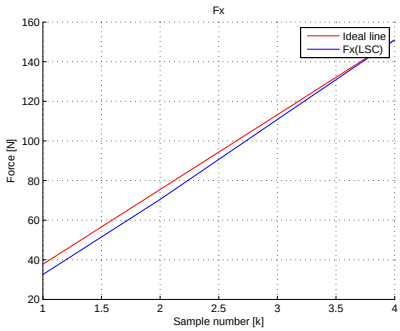
Force/Torque	**RMS error**
F_x	2.373 %N
F_y	2.096 %N
F_z	16.693 %N
M_x (roll)	2.349 % Nm
M_y (pitch)	2.228 % Nm
M_z (yaw)	2.403 % Nm

The final result for the calibration matrix can be found in Equation (H.2). In the equation there is not added 50 N on F_z since this offset error should disappear as soon as the calibration test rig is redesigned.

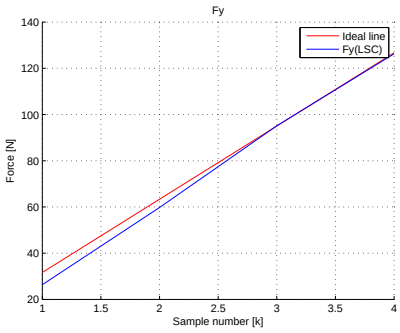
$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} -0.007 & -0.0263 & -0.0086 & 0.0176 & 0.0035 & -0.0193 \\ 0.2093 & -0.1552 & 0.2375 & -0.1925 & -0.0229 & -0.0107 \\ 0.017 & 0.0073 & 0.0197 & -0.0227 & -0.0036 & 0.0116 \\ -0.0029 & -0.0096 & -0.0082 & -0.0037 & 0.0222 & -0.0098 \\ 0.0002 & 0.0009 & 0.0007 & 0.0005 & -0.0021 & 0.0009 \end{bmatrix} \begin{bmatrix} V_{b1} \\ V_{s1} \\ V_{b2} \\ V_{s2} \\ V_{b3} \\ V_{s3} \end{bmatrix} \tag{H.2}$$

Where:

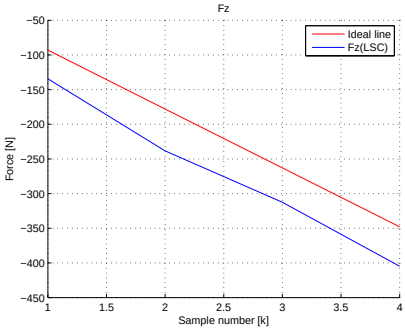
F_x	is the measured force in x-direction.
F_y	is the measured force in y-direction.
F_z	is the measured force in z-direction.
M_x	is the measured torque in x-direction.
M_y	is the measured torque in y-direction.
M_z	is the measured torque in z-direction.
V_{b1}	is the measured bending strain in bridge 1.
V_{s1}	is the measured shear strain in bridge 1.
V_{b2}	is the measured bending strain in bridge 2.
V_{s2}	is the measured shear strain in bridge 2.
V_{b3}	is the measured bending strain in bridge 3.
V_{s3}	is the measured shear strain in bridge 3.



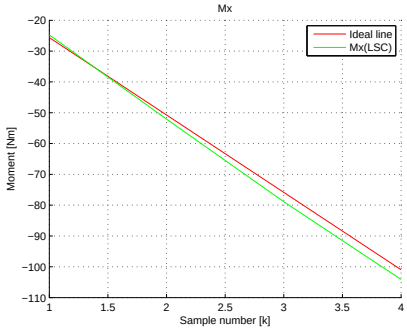
(a) F_x accuracy.



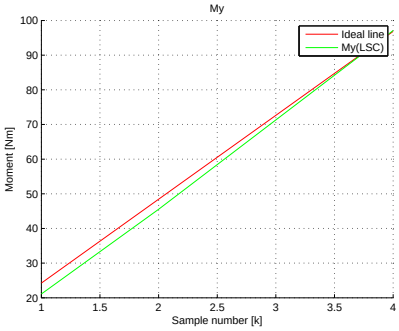
(b) F_y accuracy.



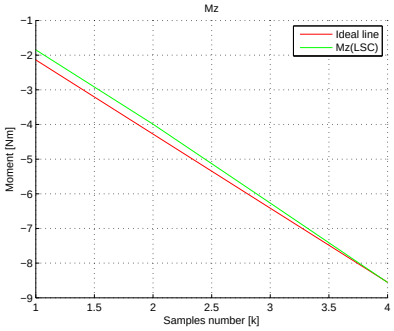
(c) F_z accuracy.



(d) M_x accuracy.



(e) M_y accuracy.



(f) M_z accuracy.

Figure H.2: Calibration results for the right FTS and amplifier.

H.3 Future Work

To achieve a better calibration of the FTS, i.e. improve the calibration results of the F_z axis, the following suggestion have been made :

- Changes the beams design, such that the mounting plate that has to be attached to the test rig, it must similar to the one on the ankle. This should give a better result in the F_z direction.
- Use solid weights instead of a Newton meter. This minimizes human reading errors. Furthermore it was also noticed that the Newton meter drifted a bit.
- Make sure that the entire test rig is designed in E.g. Solid Works. If mechanical part is done well, it will possible to obtain the exact dimensions of the calibration test rig.
- The beam must be made in a stiff material such that all applied force and torque is transferred directly to FTS.
- The test rig must be redesigned such that it is possible to use weights
- The amplifier with HW ID 14692 must be repaired or changed.

Appendix I

Test of CAN Throughput at Different Samplerates

To test the function of the designed CAN driver, the Actuator Server and the Sensor Server, the throughput of the CAN is tested.

I.1 Method

The CAN throughput is tested by putting a ramp as the input to the left arm, with the rest of the inputs set to 0. The test is set to run for 20 seconds to ensure that any effects of initializing are neglible. By doing this, the measurement of the relative angle of the left arm should be a different value at each sample. The test is carried out at three different sample rates, 100Hz, 200Hz and 250Hz. Afterwards, the last 1000 samples are tested, if any of them are equal to the sample before, there has been an error.

I.2 Result

The result from the 200Hz test can be seen in Figure I.1, the errors from the tests are counted and yields the following:

- Errors at 100Hz: 0%
- Errors at 200Hz: 1.9%
- Errors at 250Hz: 13.6%

I.3 Discussion

Although the amount of errors at 250Hz are fairly high compared to the others, the effect on **AAU-BOT1** are not noticeable by the human eye during the test. This is due to two design features:

- The CAN/EPOS driver uses PDO-frames to communicate, which does not use any retransmission strategies.

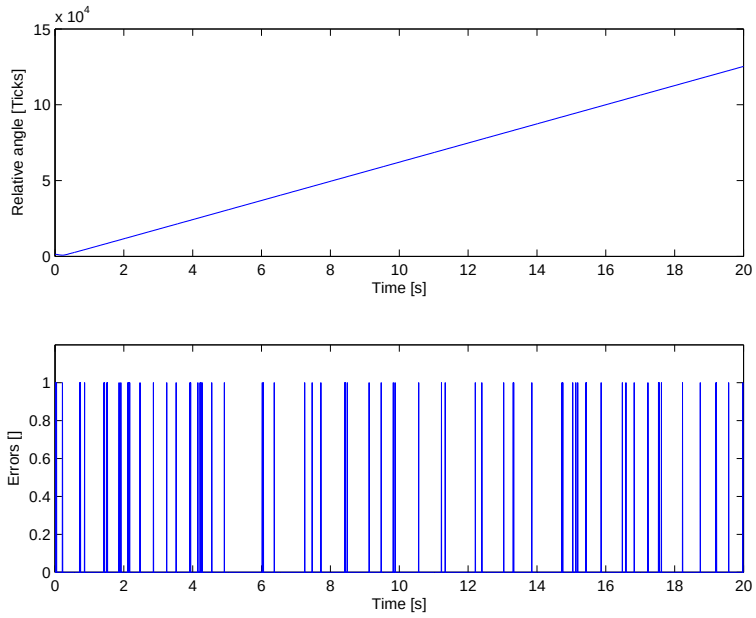


Figure I.1: Result from the 200Hz Test.

- The EPOS amplifiers are in position mode which means that if a frame is lost, the only consequence is that the next position command will make the DC motor move twice as far.

The Actuator Server and the Sensor Server should be optimized to ensure that all samples comes through, or a different sample rate should be chosen.

I.4 Summary

The CAN driver, the Actuator Server and the Sensor Server 1 are fully functional, and 98.1% of the frames are both sent and recieved by lowering the sample rate to 200Hz.

Appendix J

CAN Frame Overview

In this appendix, the individual PDO frames used to communicate with the EPOS amplifiers are defined. The RxPDOs[1] are defined in Table J.1 to J.4. The TxPDOs[2] are defined in Table J.7. To start the EPOS amplifiers, an operational mode (see Table J.5) and a `sync` command (see Table J.6) is sent.

Table J.1: PDO frames containing the velocity commands. These are sent from the on-Board computer to the EPOS amplifiers.

Adress	Length	Data	
0x201	8	$u_{\omega 2}$	$u_{\omega 3}$
0x202	4	$u_{\omega 4}$	
0x203	8	$u_{\omega 5}$	$u_{\omega 6}$
0x204	8	$u_{\omega 7}$	$u_{\omega 18}$
0x205	8	$u_{\omega 8}$	$u_{\omega 9}$
0x206	8	$u_{\omega 10}$	$u_{\omega 19}$
0x207	8	$u_{\omega 11}$	$u_{\omega 12}$
0x208	4	$u_{\omega 13}$	
0x209	8	$u_{\omega 15}$	$u_{\omega 16}$
0x20A	4	$u_{\omega 17}$	

[1] RxPDOs are sent from the on-Board computer to the EPOS amplifiers.
[2] TxPDOs are sent from the EPOS amplifiers to the on-Board computer.

Table J.2: PDO frames containing the position commands. These are sent from the on-Board computer to the EPOS amplifiers.

Adress	Length	Data	
0x301	8	$u_{\theta 2}$	$u_{\theta 3}$
0x302	4	$u_{\theta 4}$	
0x303	8	$u_{\theta 5}$	$u_{\theta 6}$
0x304	8	$u_{\theta 7}$	$u_{\theta 18}$
0x305	8	$u_{\theta 8}$	$u_{\theta 9}$
0x306	8	$u_{\theta 10}$	$u_{\theta 19}$
0x307	8	$u_{\theta 11}$	$u_{\theta 12}$
0x308	4	$u_{\theta 13}$	
0x309	8	$u_{\theta 15}$	$u_{\theta 16}$
0x30A	4	$u_{\theta 17}$	

Table J.3: PDO frames containing the current commands. These are sent from the on-Board computer to the EPOS amplifiers.

Adress	Length	Data			
0x401	6	u_{I2}	u_{I3}	u_{I4}	
0x402	8	u_{I5}	u_{I6}	u_{I7}	u_{I18}
0x403	8	u_{I8}	u_{I9}	u_{I10}	u_{I19}
0x404	6	u_{I11}	u_{I12}	u_{I13}	
0x405	6	u_{I15}	u_{I16}	u_{I17}	

Table J.4: PDO frames containing the control word. It is heard by all the EPOS amplifiers.

Adress	Length	Data
0x501	2	Control word

Table J.5: Operational mode command.

Adress	Length	Data	
0x000	2	0x01	0x00

Table J.6: Sync Command.

Adress	Length
0x080	0

Table J.7: PDO frames sent from the EPOS amplifiers to the on-board computer. These are sent every time a sync command is recieved.

Adress	Length	Data	
0x182	8	ω_2	θ_{rel2}
0x183	8	$\omega_{3,1}$	$\theta_{rel3,1}$
⋮	⋮	⋮	⋮
0x19F	8	$\omega_{12,2}$	$\theta_{rel12,2}$

0x282	8	ω_2	I_2	θ_{abs2}
0x283	8	$\omega_{3,1}$	$I_{3,1}$	$\theta_{abs3,1}$
⋮	⋮	⋮	⋮	⋮
0x29F	8	$\omega_{12,2}$	$I_{12,2}$	$\theta_{abs12,2}$

0x382	8	θ_{rel2}	I_2	θ_{abs2}
0x383	8	$\theta_{rel3,1}$	$I_{3,1}$	$\theta_{abs3,1}$
⋮	⋮	⋮	⋮	⋮
0x39F	8	$\theta_{rel12,2}$	$I_{12,2}$	$\theta_{abs12,2}$

Appendix K

Node Overview

To identify the individual EPOS amplifiers, Table K.1 is used. The Node IDs are determined by the Joint #, however at the joints with double actuated joint nodes, the second node is equal to the first Node ID + 19. The Input # and Output # are the index used in the `.mat` files.

Table K.1: Identification of nodes. The Input # and Output # is the used index in the `.mat` files.

Joint Name	Joint #	Node ID	Input #	Output #	CAN #
Right Ankle roll	2	0x002	1	1	5
Right Ankle Pitch(1)	3	0x003	2	2	5
Right Ankle Pitch(2)	3	0x016	2	18	5
Right Knee Pitch(1)	4	0x004	3	3	5
Right Knee Pitch(2)	4	0x017	3	19	5
Right Hip Roll(1)	5	0x005	4	4	3
Right Hip Roll(2)	5	0x018	4	20	3
Right Hip Pitch	6	0x006	5	5	3
Right Hip Yaw	7	0x007	6	6	3
Left Hip Yaw	8	0x008	7	7	2
Left Hip Pitch	9	0x009	8	8	2
Left Hip Roll(1)	10	0x00A	9	9	2
Left Hip Roll(2)	10	0x01D	9	21	2
Left Knee Pitch(1)	11	0x00B	10	10	4
Left Knee Pitch(2)	11	0x01E	10	22	4
Left Ankle Pitch(1)	12	0x00C	11	11	4
Left Ankle Pitch(2)	12	0x01F	11	23	4
Left Ankle Roll	13	0x00D	12	12	4
Waist Yaw	15	0x00F	13	13	1
Waist Roll	16	0x010	14	14	1
Waist Pitch	17	0x011	15	15	1
Right Arm	18	0x012	16	16	3
Left Arm	19	0x013	17	17	2

Appendix L

List of Acronyms

Acronym	Description
ADC	Analog to Digital Converter
CAN	Controller Area Network
CANopen	A higher-layer protocol for CAN.
CoM	Center of Mass
CoP	Center of Pressure
CPU	Central Processing Unit
CPG	Central Pattern Generator
DC	Direct Current
DoF	Degrees of Freedom
DSP	Double Support Phase
DSP-X	Double Support Phase Left/Right
DSP-X-H	Strike Double Support Phase Left/Right - Heel-strike
DSP-X-T	Double Support Phase Left/Right Toe
DSP-X-TH	Double Support Phase Left/Right Toe Heel
FTS	Force Torque Sensor
FZMP	Fictitious Zero Moment Point
GCC	GNU Compiler Collection
GCoM	Ground projection of Center of Mass
IMU	Inertia Measurement Unit
LAN	Local Area Network
OP-AMP	Operation Amplifier
PCB	Printed Circuit Board
PDO	Process Data Object
PSU	Power Supply Unit
SA	Support Area
SDO	Service Data Object
SNR	Signal to Noise Ratio
SSP	Single Support Phase
SSP-X	Single Support Phase Left/Right
SSP-X-T	Single Support Phase Left/Right Toe
TLA	Three Letter Acronym
USB	Universal Serial Bus
ZMP	Zero Moment Point

Appendix M

Contents of the enclosed CD

The enclosed CD has the following contents:

- `/bibliography/`: Contains some of the papers from the bibliography.
- `/datasheets/`: Contains the data sheets of the hardware of **AAU-BOT1**.
- `/doc/`: Contains the documentation of the source code in `/src/`.
- `/images/`: Contains the raw images of this report.
- `/maple/`: Contains the Maple code of the models.
- `/masterThesis/`: Contains this report, in pdf and ps format.
- `/matlab/`: Contains the Matlab code of the models.
- `/src/`: Contains the source code of the software that was designed in Chapter 4.
- `/webots/`: Contains the Webots model and the source code.

Index

Balanced gait, 30

Center of Mass, 28
Center of Pressure, 30
Controller structure, 119

DC Motor, 35
    Actuation Limits, 74
    Amplifiers for, 37
        Feedback Control of, 72, 130–132
    Double Actuated Joints, 73
        Control of, 132
    Gearing, 72
    Model of, 71
        Verification of, 163
Definition of coordinate system, 23
Double Support Phase (DSP), 26
Dynamic Model, 83–89
    DSP, 89
    DSP Example, 200–201
    SSP, 85
    SSP Example, 196–200
Dynamically balanced gait, 30
    Trajectory for, 203

Fictitious Zero Moment Point, 30
Foot Model, 188
Force Torque Sensor, 38–43
    Alternative FTS DAQ, 209–213

Gait, 23
Ground projection of Center of Mass, 29, 138

Human gait, 24

Inertia Measurement Unit, 45
Inverse Kinematics, 91
    Verification of, 169

Kalman Filter, 125
Kinematics, 77–83
    Example, 193–196
    Verification of, 166

Observer, 139
On-board Computer, 43

Phase Estimator, 89, 139
Posture Control, 121
Potentiometers, 37

Single Support Phase (SSP), 26
Software, 49
    Actuator Server, 57
    EPOS/CAN Driver, 57
    FTS driver, 60
    S-Functions, 51
    Sensor Server, 54
    Shared Memory Server, 54
    Webots, 66
Statically balanced gait, 30
    Trajectory for, 102
Step, 23
Supervisor, 140
Support Area, 28
    Example, 193, 201
Support Phases, 26–28
System Representation, 77

Trajectory Generation, 97
Transformation matrices, 78

Walk, 23

Zero Moment Point, 29, 89, 138, 139, 201
ZMP estimator, 139