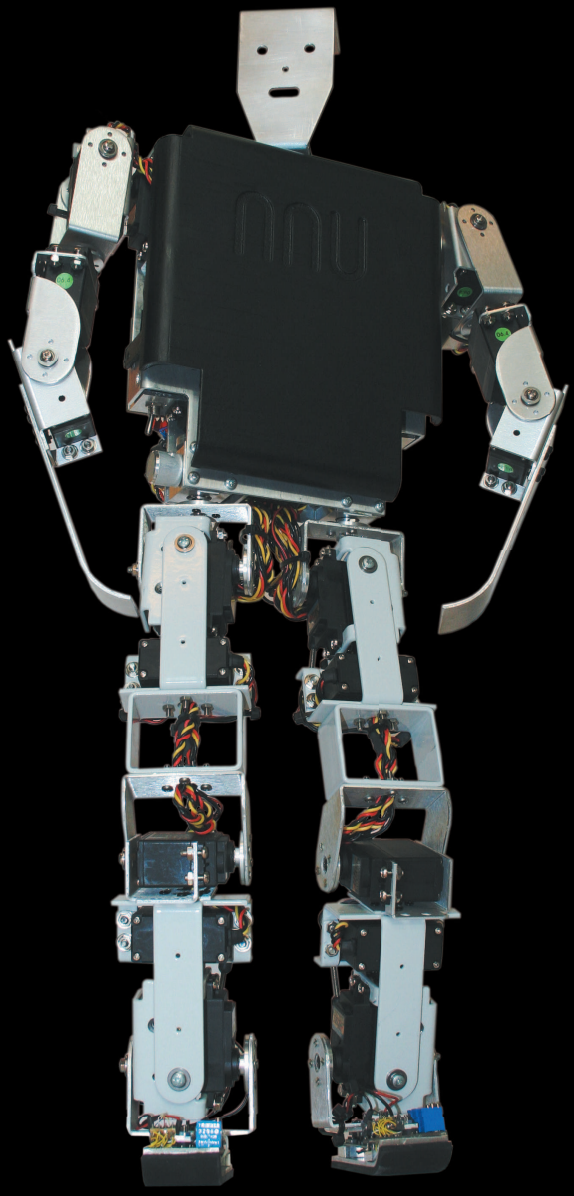
Development, Modeling And Control of A Humanoid Robot



Master's Thesis
Jens Christensen - Jesper Lundgaard Nielsen
Mads Sølver Svendsen - Peter Falkesgaard Ørts
Aalborg University 2007

AALBORG UNIVERSITY
DEPARTMENT OF ELECTRONIC SYSTEMS
SECTION OF AUTOMATION AND CONTROL

DEVELOPMENT, MODELING AND CONTROL OF

A HUMANOID ROBOT

INTELLIGENT AUTONOMOUS SYSTEMS

GROUP 1033

MASTER'S THESIS - SPRING 2007

JENS CHRISTENSEN - JESPER LUNDGAARD NIELSEN
MADS SØLVER SVENDSEN - PETER FALKESGAARD ØRTS

AALBORG UNIVERSITY

Department of Electronic Systems
Fredrik Bajers Vej 7 A1
9220 Aalborg, Denmark
Phone: +45 96358600
Web: http://es.aau.dk/

TITLE: Development, Modeling and Control of a Humanoid Robot

PROJECT SEMESTER:
9th and 10th semester, 4th September 2006 - 7th June 2007

PROJECT GROUP:
1033

GROUP MEMBERS:
Jens Christensen
Jesper L. Nielsen
Mads S. Svendsen
Peter F. Ørts

SUPERVISORS:
Jan Helbo
Dan D. V. Bhanderi

NUMBER OF COPIES:
14

NUMBER OF PAGES THESIS:
164

TOTAL NUMBER OF PAGES:
277

APPENDED DOCUMENTS:
1 CD-ROM

FINISHED:
7th June 2007

ABSTRACT:

This master's thesis concerns the development, modeling and control of a humanoid robot, which enables human-like walk.
As the focus is to obtain human-like walk, the robot is designed to resemble human proportions and a special joint has been developed to resemble the hip joint of humans, and thereby enabling walking in curved paths. Furthermore the hardware necessary to obtain a fully autonomous system is developed and implemented. The result of the design phase is a humanoid robot, called "Roberto", measuring 58 cm and with 21 actuated degrees of freedom.
A complete dynamical model describing the system has been developed. The model is a hybrid model which enables simulation of complete walking cycles. A novel solution of the dynamics of the robot during double support phase has been given.
To enable human-like walk a set of trajectories has been developed, based on the zero-moment point and dynamical simulations. The trajectories are simulated, and human-like walk is obtain on the model. To maintain stability during walk with the real robot, two controllers have been developed, a posture controller and a zero-moment point controller. It was found that the controllers were able to track a zero-moment point reference and a inclination reference given to the system.
Human-like walk was not obtained on the real system, due to system limitations. If a new interface to the DC-motors in the servos was developed, and a faster on-board computer was chosen, human-like walk should be possible.

AALBORG UNIVERSITET

Institut for Electroniske Systemer
Fredrik Bajers Vej 7 A1
9220 Aalborg Ø
Tlf: +45 96358600
Web: http://es.aau.dk/

TITEL: Development, Modeling and Control of a Humanoid Robot

PROJEKTPERIODE:
9. og 10. semester,
4. september 2006 - 7. juni 2007

PROJEKTGRUPPE:
1033

GRUPPE MEDLEMMER:
Jens Christensen
Jesper L. Nielsen
Mads S. Svendsen
Peter F. Ørts

VEJLEDERE:
Jan Helbo
Dan D. V. Bhanderi

ANTAL KOPIER:
14

SIDER I HOVEDRAPPORT:
164

SIDER IALT:
277

BILAG:
1 CD-ROM

AFSLUTTET:
7. juni 2007

SYNOPSIS:

Dette speciale omhandler udvikling, modellering og kontrol af en menneskelignende robot, hvorpå menneskelignende gang ønskes implementeret.
Der er i dette speciale fokuseret på at opnå menneskelig gang og robotten er derfor designet med menneskelige proportioner. Dertil er et specielt led blevet konstrueret, der ligner den menneskelige hofteskål og giver robotten mulighed for at dreje under gang. Der er ydermere udviklet og implementeret hardware, der gør robotten fuldt ud autonom. Resultatet er en menneskelignende robot, kaldet ”Roberto”, der er 58 cm høj og har 21 aktuerede frihedsgrader.
Der er blevet udviklet en komplet dynamisk model, som beskriver alle input og output af systemet. Modellen en hybrid model, der muliggører simulering af en komplette gang cykler. En ny løsning er blevet foreslået, der beskriver dynamikken af robotten, når den har begge ben på jorden.
Et sæt af menneskelignende gangtrajektorier er blevet udviklet, som er baseret på zero-moment point og dynamiske simuleringer. Disse trajektorier er blevet simuleret og menneskelig gang blev opnået på modellen. For at opretholde stabilitet under gang, med den udviklede robot, er to regulatorer blevet designet. Disse kontrollerer kropsholdningen samt positionen af zero-moment point. Regulatorerne var i stand til at følge en reference, både et zero-moment point og en given orientering.
Menneskelig gang blev ikke opnået på den rigtige robot, grundet begrænsninger i systemet. Det blev vurderet, at hvis der blev konstrueret et nyt interface print til DC-motoren inde i servoerne, ville dette give bedre resultater. Ydermere blev det anbefalet at implementere en hurtigere computer på robotten. Med disse opgraderinger skulle det være muligt at opnå menneskelig gang med robotten.

PREFACE

This master's thesis is written at the Department of Electronic Systems at the Section of Automation and Control at Aalborg University, under the Master's program "Intelligent Autonomous Systems".

It is the documentation of the work performed by the group members in their 9^{th} and 10^{th} semester. A part of the project was to construct a humanoid robot. This new humanoid robot was named "Roberto". In order to investigate the area of biped robotics, the group members have been on a study trip to Tokyo. This included, among others, a visit to the University of Tokyo and a visit to their robotic laboratory.

Throughout this project, MATLAB has been used for capturing, processing and representation of data, specifically MATLAB version 7.3, R2006b. Simulink version 6.5, R2006b has been used for building the model and for making an interface to the robot. SolidWorks 2006, SP0.0 has been used to design the mechanical parts of the robot, and to retrieve all kinematic information. The mechanical design can be found on the enclosed CD-ROM. Maple 10, has been used for large algebraic computations and for optimization of the equations used in the model.

References to literature will be done using the Harvard method. Throughout the thesis figures, tables and equations are numbered consecutively inside each chapter. Whenever an illustration can be made more expressive, a red colour is used to represents the left side and a green colour for the right side, as used in maritime navigation.

On the last page, a CD-ROM is enclosed, which contains literature, model files, drawings of the robot etc. A complete description of the contents on the CD-ROM can be found in Appendix O on page 277. Furthermore a list of the acronyms used throughout this thesis can be found in Appendix N on Page 275.

Aalborg University, 2007

Jens Christensen

Jesper L. Nielsen

Mads S. Svendsen

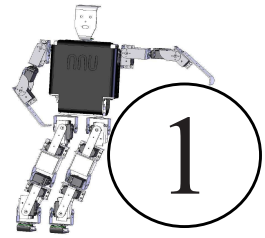
Peter F. Ørts

TABLE OF CONTENTS

List of Figures

List of Tables

INTRODUCTION

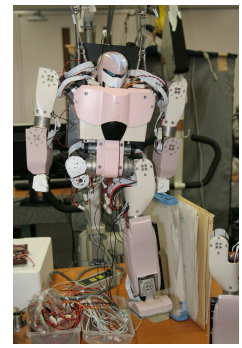


1.1 Background

Conventional robot manipulators have been studied for many years, and are greatly utilized in the industry to improve the output of the production and ease the heavy workload previously imposed on humans. In the medical world robots have also been used to aid in surgeries that require high accuracy. Furthermore traditional wheeled robots have been used to perform tasks where movement of the robot is necessary. However in the recent years more and more interest has been given toward humanoid or biped robots. The advantage of these two-legged robots is, that they are often capable of performing more versatile and demanding tasks, than the traditional robots.

(a) ASIMO (130 cm)

(b) PINO(70 cm)



(c) UT-μ (58 cm)

Figure 1.1: *Three different biped robots. ASIMO has been developed by Honda, PINO has been developed by ZMP Inc. and UT-μ has been developed by students at the University of Tokyo. The height of the biped is given belov each picture.*

Biped robots could be used to assist humans in carrying heavy materials around, entering high risk areas such as an atomic power plant, aid in the household, etc. An advantage of the biped robots, compared to the wheeled, is the ability to move around in human environments, where different obstacles or stairs should be surmounted. A

human sized robot is at the moment under development at Aalborg University. The goal of this robot is to be able to investigate different types of prothetic legs before they are used on humans. It is expected that within the coming years, the need for two legged robots will increase, and as the knowledge in the area is expanded, the number of different tasks that can be solved by robots will increase rapidly.

The number of successfully developed biped robots is relatively small, but several companies have invested large amounts of money and research into developing such robots. For instance Hondas ASIMO, which can be seen in Figure 1.1(a), is one of the most advanced in the field. Also the Japanese company ZMP Inc. has developed a humanoid called PINO, which is shown in Figure 1.1(b). Several universities have also joined the race in developing humanoid robots. The Technical University of Munich has developed the robot Johnnie, and at the University of Tokyo students have developed the small UT-μ, seen in Figure 1.1(c). They have also developed the human sized robot UT-θ, which is seen in Figure 1.2.

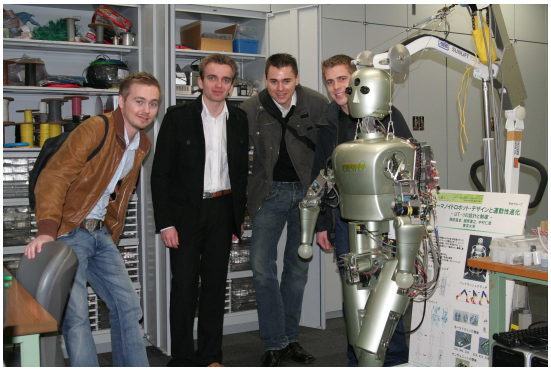


Figure 1.2: *A picture of the group members and UT-θ, taken at the University of Tokyo during a study trip in February 2007. UT-θ is* $150\,\mathrm{cm}$ *high.*

The task of making a humanoid robot walk is however not trivial, since the system is a highly complex dynamical system often with a large number of degrees of freedoms. This is also what makes the area very interesting, and properly why a large number of researchers have dedicated themselves to solve these challenging problems.

To the best of our knowledge, no one has obtained real human-like walk on a biped robot. This master's thesis will therefore focus on developing a humanoid robot and obtaining human-like walk. The development is realized in a number of steps:

- Development of a fully autonomous robot platform.
- Development of a complete dynamical model of the biped robot.
- Creation of dynamic human-like walking trajectories based on human gait analysis.
- Implementation of controllers to suppress external disturbances and model uncertainties.

The following will give an outline on how the above has been achieved.

1.2 Thesis Outline

This thesis is a documentation of the development, modeling and control of a humanoid robot. An illustration of the platform is shown in Figure 1.3.

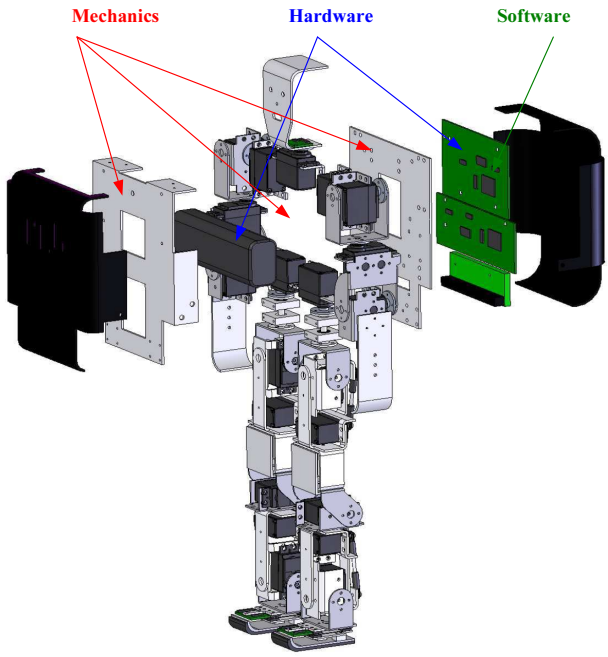


Figure 1.3: *Exploded view of the developed humanoid.*

This platform also gives an outline of the thesis:

Analysis: First some conceptual knowledge is given in Chapter 2 in order to equip the reader with a basic understanding of the area. In Chapter 3 an analysis of the human gait is given to determine key parameters in human locomotion.

Development: The development of the biped robot is documented through three chapters. Chapter 4 gives a description of the mechanics. This includes the design of all joints and the torso on which the hardware is mounted. The limbs are designed to match human proportions. Next, the hardware, on-board computer, power supply, actuators and sensors, is explained in Chapter 5. The operating system and drivers are then described in Chapter 6 along with the developed software.

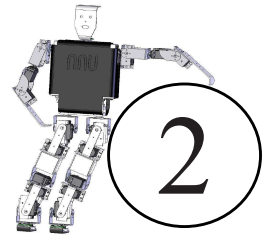
Modeling: Having given a description of the physical platform, a model of the system is then derived. This model is a complete input to output model of the system, and is described in Chapter 7. The next chapter, Chapter 8, is a description of the inverse kinematic model for the robot. This is necessary in order to create a usable control input for the robot.

Control: Then the control for the robot is explained, starting in Chapter 9 with a derivation of the walking trajectories used for making the robot perform dynamic

walk. Next, the controllers used to maintain balance and suppress external disturbances are explained in Chapter 10.

Epilogue: In Chapter 11, an epilogue, with discussion and conclusions of the obtained results, is given.

CONCEPTUAL KNOWLEDGE



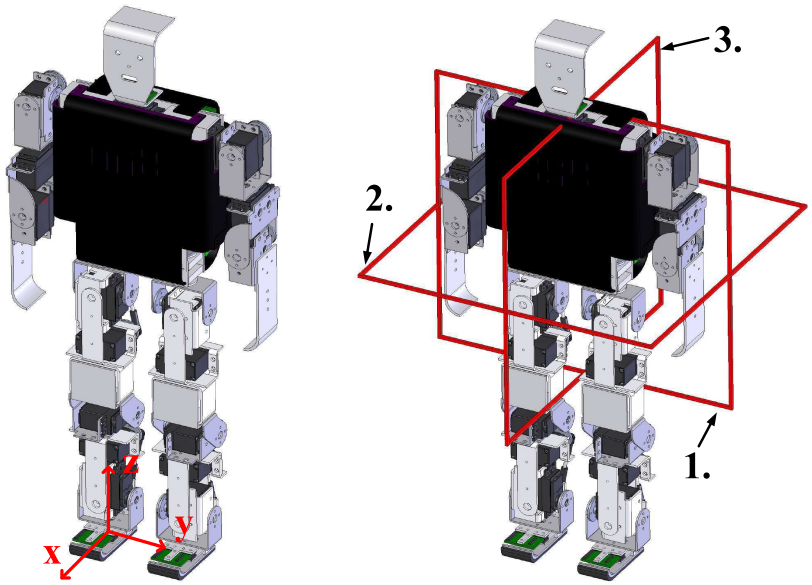
2

Chapter contents

IN THIS CHAPTER *some of the basic notions and terms used in the area humanoid robotics will be presented. Especially those terms used throughout this thesis will be covered. This is done to build up a common understanding of the terms that are used. First some basic information on the representation of kinematic of the biped robot is examined in Section 2.1. Then, in Section 2.2, different terms attached to the locomotion of humanoid robots will be covered. Afterward some terms concerning the stability of a humanoid robot will be examined, which is done in Section 2.3, at the end of the chapter. Both the zero-moment point is examined and the centre of pressure will be covered. The zero-moment point is the measure of stability in the field of humanoid robotics.*

2.1 Representation

It is necessary to define how the coordinates are represented, unless otherwise specified positions are given in the global reference frame. The global reference frame is a right-handed XYZ-coordinate system, that is place in the right foot. The coordinate system is illustrated in Figure 2.1(a), and is orientated with the x-axis pointing forward, the y-axis pointing to the left, and the z-axis pointing upward intersecting the ankle joint. This means that the x-direction is the walking direction.



(a) Reference frame (b) Planes

Figure 2.1: *2.1(a) shows the position and orientation of the global reference frame. 2.1(b) shows the three planes of motion. 1. is frontal plane, 2. is horizontal plane and 3. is sagittal plane.*

Further more, when talking about motion of a humanoid, it is often desired to address motion in certain planes. For this reason three planes perpendicular planes are specified:

1. Frontal plane
2. Horizontal plane
3. Sagittal plane

where the numbers refer to Figure 2.1(b), where the three planes are illustrated.

2.2 Humanoid Robots Locomotion

This section will mainly rely on [Vukobratović et al., 2006] and [Azevedo et al., 2005] who have made a contribution toward a unification in the area of humanoid robots.

Walk In the area of humanoid locomotion walk is defined as: *Movement by putting forward each foot in turn, not having both feet off the ground at once* [Vukobratović et al., 2006].

Gait The term gait and walk is not the same, gait refers to the *manner of walking*. Hence when a humanoids walk, the can have different gaits. If a vector, $\boldsymbol{\theta}(t)$ is defined to contain all the joint angles, then a time history of $\boldsymbol{\theta}(t)$ represents the specific gait.

Step A step is defined as: *in the direction of motion, during the contact with the ground, the leg from the front position with respect to the trunk comes to the rear position, then it is deployed from the ground and in the transfer phase moves to the front position, to make again contact with the ground, and the cycle is repeated* [Vukobratović et al., 2006]. Note that the duration of a step runs from the foot has a certain position until it reaches that position again.

Support Phases A step can be divided into a large number of phases but must at least contain the following two: a single support phase (SSP) where only one foot is in contact with ground, and a double support phase (DSP) where both feet are in contact with ground. The SSP can be divided into two different phases, namely the left single support phase (SSP-L), where the left leg is the supporting leg, and the right single support phase, where the right leg is the supporting leg. SSP-L and SSP-R can be further divided into the lift-off phase and the impact phase. The impact phase starts when the heel of the rear foot leaves the ground and ends when the toe leaves the ground, afterwhich the SSP starts. The impact phase is when the heel of the front foot hits the ground.

Periodic Gait A gait is periodic if the same step is repeated identically, thus $\boldsymbol{\theta}(t) = \boldsymbol{\theta}(t+T)$, where T is the duration of one step.

Symmetric Gait A gait is symmetric if the step can be divided into two equal time periods, and the right leg in one period behaves as left leg in the other period. Thus $\boldsymbol{\theta}_{\text{right}}(t) = \boldsymbol{\theta}_{\text{left}}(t+T/2)$ where $\boldsymbol{\theta}_{\text{right}}$ are the joint angles on the right leg, and $\boldsymbol{\theta}_{\text{left}}$ are the corresponding joint angles on the left leg.

2.3 Stability

As Section 2.2 this section will mainly build on [Vukobratović et al., 2006] and [Azevedo et al., 2005], but also [Bachar, 2004] will be used as a source of inspiration.

In biped robotics, different types of stability during gait have been developed and these types are divided in two different categories.

Static stable gait Which is characterized by its ability to maintain stability at all instances of the walk, due to the fact that system dynamics is neglected because of slow movements. This means that when the gait is static stable, it is actually possible to interrupt the biped robot, and it will maintain its current position, until the the walk is resumed. One of the disadvantages of this type of walk is that the motion has to be slow, otherwise the system dynamics can not be neglected. Further, the CoM of the robot must remain within the support area at all time, and depending on the size of the feet, the support area can be relatively

small. Trying to keep the CoM inside this little area can make the walk appear duck-like. Therefore a disadvantage is that the walking does not resemble human walk very well, as humans emphasize the dynamics of the walk, to minimize the energy usage while walking [Collins et al., 2001]. In [Christensen et al., 2006] statical stable walk has successfully been implemented on a 21 DoF biped robot.

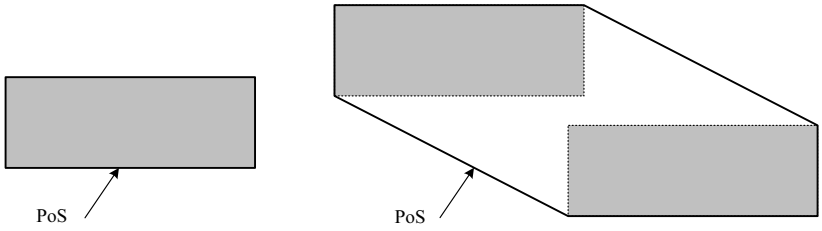
Dynamic Stable Gait This is more difficult to define, but the gait is defined as dynamical stable, when dynamical balance is maintained. The rest of this section is devoted to investigate when the biped robot can be designated as dynamical stable.

The term dynamic stability is difficult to define in the case of a biped robot, though it has a very precise definition in the area of system control, biped locomotion cannot be classified as stable in the sense of a classical dynamical system. In [Vukobratović et al., 2006] the dynamical balance of a human or humanoid is formulated as: *if there is no rotation of the supporting foot (or feet) about its (or their common) edge during walking.*

A common way, in the area of robotics, to investigate if this criteria is fulfilled, is to calculate the ZMP. The ZMP was introduced by [Vukobratović and Juriĉić, 1969]. ZMP is the point on ground, where the moments around any axis, passing through this point and being tangential to the ground, is equal to zero. This can be expressed as:

$$\sum M_{\mathrm{x}} = 0 \text{ and } \sum M_{\mathrm{y}} = 0 \tag{2.1}$$

where M_{x} and M_{y} are the moments around the x-axis and the y-axis, respectively. Now if this point is kept within the convex hull of all contact points between the biped robot and the ground, dynamical stability is ensured. The convex hull of all contact points is named the polygon of support PoS, and is illustrated in Figure 2.2, where Figure 2.2(a) shows the PoS in SSP, and Figure 2.2(b) shows the larger PoS that is obtained in DSP.



(a) Single support phase (b) Double support phase

Figure 2.2: *The PoS is the convex hull of all contact points. If the ZMP is located inside PoS, dynamical stability is ensured. The gray area represents the foot soles.*

2.3.1 The Zero-Moment Point

The ZMP concept was introduced in [Vukobratović and Juriĉić, 1969], and has since been widely use as a measure of stability of biped robots. In the literature different definitions of the ZMP can be found, and it is therefore chosen to explain the concept.

The ZMP is defined in the horizontal plane, and is the point at which all moments are zero, hence the name. From [Bachar, 2004], it is given that the ZMP must apply the following two equations:

$$\sum_{i=1}^{n} \left(m_i \left(\boldsymbol{r}_i - \boldsymbol{p}\right) \times \left(\ddot{\boldsymbol{r}}_i + \boldsymbol{g}\right) + \boldsymbol{I}_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \boldsymbol{I}_i \boldsymbol{\omega}_i\right) = \boldsymbol{M} \tag{2.2}$$

$$\boldsymbol{M} \times \boldsymbol{g} = \boldsymbol{0} \tag{2.3}$$

where:

n		:is number of links
m_i		:is the mass of link i
$\boldsymbol{r}_i$	$\equiv [x_i \; y_i \; z_i]^{\mathrm{T}}$	:is the position of CoM of link i
$\boldsymbol{p}$	$\equiv [x_{\mathrm{ZMP}} \; y_{\mathrm{ZMP}} \; z_{\mathrm{ZMP}}]^{\mathrm{T}}$	:is the position of ZMP
$\boldsymbol{g}$	$\equiv [g_x \; g_y \; g_z]^{\mathrm{T}}$	:is the gravity acceleration
$\boldsymbol{I}_i$	$\equiv \mathrm{diag}(I_{ix}, I_{iy}, I_{iz})$	:is the inertia tensor of link i
$\boldsymbol{\omega}_i$	$\equiv [\omega_{ix} \; \omega_{iy} \; \omega_{iz}]^{\mathrm{T}}$	:is the angular velocity of link i
$\boldsymbol{M}$	$\equiv [M_x \; M_y \; M_z]^{\mathrm{T}}$	:is the moment around ZMP

Now from [Bachar, 2004], it is given that Equation (2.2) and (2.3) can be combined into one equation:

$$\sum_{i=1}^{n} \left(m_i(\boldsymbol{r}_i - \boldsymbol{p}) \times \ddot{\boldsymbol{r}}_i + \boldsymbol{I}_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \boldsymbol{I}_i \boldsymbol{\omega}_i - m_i(\boldsymbol{r}_i - \boldsymbol{p}) \times \boldsymbol{g}\right) = [0 \; 0 \; *]^{\mathrm{T}} \tag{2.4}$$

In [Huang et al., 2001] and in [Peng et al., 2005] the solution to the two scalars representing the ZMP is given as:

$$x_{\mathrm{ZMP}} = \frac{\sum_{i=1}^{n} m_i \left(x_i(\ddot{z}_i + g_z) - \ddot{x}_i z_i\right) - I_{iy}\dot{\omega}_{iy}}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g_z)} \tag{2.5}$$

$$y_{\mathrm{ZMP}} = \frac{\sum_{i=1}^{n} m_i \left(y_i(\ddot{z}_i + g_z) - \ddot{y}_i z_i\right) - I_{ix}\dot{\omega}_{ix}}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g_z)} \tag{2.6}$$

where the height of the ground is set to zero. In [Erbatur et al., 2002] and [Bachar, 2004] a more simple method to calculate the ZMP was proposed; assuming that the mass of link i is uniformly distributed about the centre of mass the inertia can be

ignored, resulting in equations that would be more suited for on-line calculation:

$$x_{\text{ZMP}} = \frac{\sum_{i=1}^{n} m_i \left(x_i(\ddot{z}_i + g_z) - \ddot{x}_i z_i\right)}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g_z)} \tag{2.7}$$

$$y_{\text{ZMP}} = \frac{\sum_{i=1}^{n} m_i \left(y_i(\ddot{z}_i + g_z) - \ddot{y}_i z_i\right)}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g_z)} \tag{2.8}$$

In Equation (2.5) to (2.8) it is further assumed that the ground is horizontal, resulting in $\boldsymbol{g} = [0\ 0\ g_z]^{\mathrm{T}}$. In [Vukobratović and Juriĉić, 1969] it is stressed that the ZMP is only defined within the PoS and can only move to the edge of the foot. If the ZMP is on the edge of the foot then the robot will start to overturn, which means that the foot is no longer fixed to the ground, and the ZMP is no longer defined. However, Equation (2.5) and (2.6), and their simplified equvalents Equation (2.7) and (2.8), can still yield a ZMP outside PoS which is covered next.

2.3.2 The Fictitious Zero-Moment Point

When the ZMP leaves the PoS it is known as the Fictitious ZMP (FZMP). FZMP expresses the point where Equation (2.1) is satisfied, however the assumption that the PoS is fixed to the ground is no longer valid since the robot will have initiated a tilt and PoS is reduced to a line. This ZMP will not reveal any information about the dynamic stability of the robot which is why it is called FZMP. When humans walk the FZMP can be observed at the end of SSP, just before the heel strikes, and at this stage the human gait is no longer dynamic stable [Vukobratović and Juriĉić, 1969]. Keeping the ZMP within the PoS ensures a dynamic stable gait, however, it is not a necessary condition in achieving a stable gait.

2.3.3 The Centre of Pressure

The ZMP and the centre of pressure (CoP) are the same point as long as the ZMP is inside the PoS. The CoP is found much simpler, if pressure measurements are available, and can be used to estimate the ZMP, as long as it is inside the PoS. The equation for calculating the CoP in SSP is as follows, [Erbatur et al., 2002]:

$$\boldsymbol{p}_{\text{CoP}} = \frac{\sum_{i=1}^{n_{\text{f}}} f_i \boldsymbol{r}_{i\text{f}}}{\sum_{i=1}^{n_{\text{f}}} f_i} \tag{2.9}$$

where n_{f} is the number of pressure sensors on the foot, f_i is the force applied on sensor i, and $\boldsymbol{r}_{i\text{f}}$ is the position of sensor i, as illustrated in Figure 2.3. If the biped robot is in DSP, the CoP can be found by interpolation the CoP calculated for each foot, and

weighing by the total force applied to all the sensors:

$$\boldsymbol{p}_{\text{CoP}} = \frac{\sum_{i=1}^{n_{\text{fr}}} f_{i\text{r}} \boldsymbol{r}_{i\text{r}} + \sum_{i=1}^{n_{\text{fl}}} f_{i\text{l}} \boldsymbol{r}_{i\text{l}}}{\sum_{i=1}^{n_{\text{fl}}} f_{i\text{l}} + \sum_{i=1}^{n_{\text{fr}}} f_{i\text{r}}} \tag{2.10}$$

where $n_{i\text{r}}$ and $n_{i\text{l}}$ are the total number of sensor on right and left foot respectively, $f_{i\text{r}}$ and $f_{i\text{l}}$ are the forces applied to sensor i on right and left foot respectively and $\boldsymbol{r}_{i\text{r}}$ and $\boldsymbol{r}_{i\text{l}}$ are the positions of the the sensors on the right and left foot respectively.

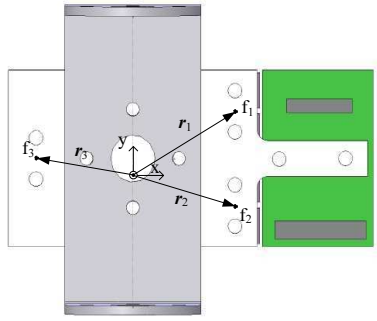
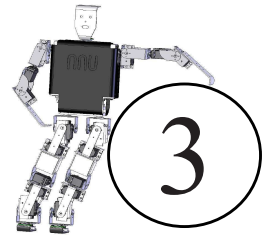


Figure 2.3: *Top view of the foot. $\boldsymbol{r}_i$ is the position of sensor i given in the global reference frame, f_i is the measured force of sensor i.*

Analysis of Human Gait



3

Chapter contents

THE PREVIOUS *chapter gave an understanding of some of the important aspects used in humanoid robotics and expressed an objective for this master thesis. In this chapter an analysis of the posture of the human body, both during gait in a straight line and during gait that follows a curved path is given. The analysis is performed to reveal some basic information on how human-like walk should be obtain with a humanoid robot. The analysis emphasizes how commands are given to each limb, which can be utilized in the control of the system. Further the analysis directs attention to some important issues when designing the mechanics of the humanoid robot. The purpose of this is to further analyze the objective to elaborate it.*

3.1 Human Gait

This section describes the human gait, and is inspired from [Popovic, 2006]. It is emphasized how commands to each limp is given from the central nervous system, located in the brain. The information regarding how the human body controls stability could be used for controller design.

The human body is highly unstable as the base of support is limited compared to the body mass. Furthermore a large part of the body mass (head, arms and trunk) is located high above ground, causing the CoM to be located at $65\,\%$ of the body height above ground. The SSP is approximately $80\,\%$ of the walking cycle, resulting in a relatively little PoS most of the cycle. For this reason learning bipedal walk is a complicated problem that takes years to learn. Motion relies on learning and experience from earlier mistakes. Newborn babies show an ability to walk when holding them under the armpits and this is inherited so that it is performed instinctively. This walk is enforced through learning. Humans enforce their walking throughout life, and if the learned walk is degraded by impairment the walk can be adjusted to fit the impairment. The visibility of the impairment is larger when the desired task is complex.

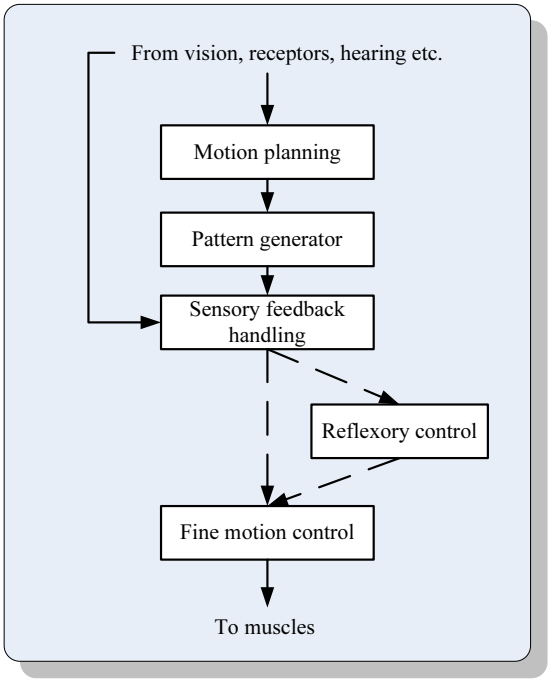


Figure 3.1: *The human motion control division.*

Human motion control can be divided into different subsystems. A system that handles the generation of rhythmic patterns, a system used to handle responses from sensory feedback and a system to handle planning of movements by responding to visual feedback. Accurate control of the muscles to increase precision is kept on a lower layer in the control of the body. The division of the human motion control system can be seen

from Figure 3.1. Human reflexes are not used in ordinary mode, but are an underlaying structure which is ready to take over if the normal procedures does not fit the conditions, which is also shown in Figure 3.1.

The stand of a human or biped robot is the posture where the net torque and force generated by gravity and muscles are zero, and thus the balance is concentrated around keeping an upright equilibrium position. Walking is caused by a desire to move the body from one point in space to another, which means that all movements, but reflective movements, are planned before they are initiated, this is also seen from Figure 3.1. Walking initiates when the stance equilibrium is disrupted by internal forces of muscle activity. The torques move the CoM out of the stable stance region and thereby initiates a fall. This is the beginning of walk. The gravity is used to bring forward the CoM, and to prevent a fall the counter leg is used to support the falling body. The counter leg takes over support of the body and regains stability using the momentum gained by the gravity. Then upright double support position is regained. This cycle is repeated, and thus walking is characterized by cyclic repetition of movements. Locomotion is established by cooperation of the entire body. The arms are used to gain or remove energy from the moving body, and to maintain stability. When walking, the body has a certain momentum, and by forcing the arms back this momentum can be decreased by moving the CoM.

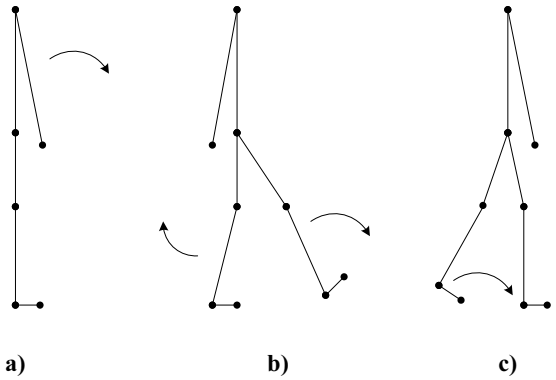


Figure 3.2: *The human motion cycle. a) The cycle is initiated by internal muscle activity, and the upper body moves forward in a free fall. b) The direction of the fall is controlled by the stance leg, and the swing leg is moved forward to support the body. c) The stance leg is shifted and initial position is regained by moving the new swing leg.*

When walking, the human tries at all time to benefit from the gravity, thereby minimizing the energy used to move the body. According to [Popovic, 2006] human walk can be described using an inverted multi-link pendulum, moving the CoM in the direction of the movement. The movement of the swing leg can be seen as a normal pendulum, and is used to prevent falling. The inverted pendulum motion is repeated when the swing leg contacts the ground. At normal speed the muscles exhibit low activity in the swing phase, they only burst during the initiation or end of a swing. When walking, there are three different phases. DSP, SSP-L and SSP-R, as explained in Section 2.2. Between these phases is the impact/switch phase, where the foot, not supporting the

weight, impacts the ground. During DSP the muscles are active and used to establish initial conditions of the upcoming SSP. Simultaneous control of both internal and external forces are used, meaning that the muscles are controlled and at the same time the dynamics given by the movement in a gravity field is exploited. The most important feature when walking is to keep a vertical posture of the upper body, using one or two legs.

As earlier explained, a central pattern generator is used to generate the cyclic motion. The pattern generator can be activated with signals from the brain. The pattern generator activates the muscles at the appropriate time so the cyclic motion is kept, called time keeping, and furthermore it generates the appropriate patterns to the muscles for the desired task. Phase division can be applied to many of the patterns generated, e.g. a swing phase and a stance phase for a leg. When an obstacle is detected, the initiated pattern is disrupted, and a decision to avoid it must be taken. Four major possibilities are available in this situation. These are lifting the leg over the obstacle, slowing down the pace by taking smaller steps, regulate the sideways step width or completely change direction by initiating a rotated movement on the supporting leg. The last option requires the most energy.

The human sensor system is altered to fit the current condition. This means that when walking, the sensors are mainly used to sensor stable walk and obstacles, and are not used for other purposes.

Humans adjust the walk to fit the terrain, load and unexpected events. Main sensors used to control the walk is pressure feedback from the sensory system located in the skin, balance input from sensors located in the brain used to adjust the posture of the body and the last main sensor is the visual input which is among others used to detect obstacles and thereby give a reflectory movement of the body, see [Kuo, 1997].

3.2 Posture During Straight Gait

The previous section gave a description of the human gait. In this section the posture of a human body during walk will be analyzed from test results. The basis for this analysis will be the results of a test conducted at the Center of SMI (Sensory-Motor Interaction) at Aalborg University, previously described in [Christensen et al., 2006]. These results comprise 14 joint trajectories of a normal human walking a straight line measured in three dimensional space, henceforth referred to as straight human gait. In this section it is emphasized how the body posture is, seen from a human control/balance point of view.

3.2.1 Determination of Phase Transitions

In [Christensen et al., 2006] different phase transitions for the straight walk was found, the date is listed in Table 3.1.

What is worth noticing in Table 3.1, is that during straight human walk, the SSP is long compared to the full cycle. The test subject is in SSP $92\,\%$ of the cycle. In Section 3.1 it was stated that the human body minimizes the use of energy while in SSP and enables the muscles while in DSP. Since the DSP only last $8\,\%$ of the full cycle, which

	Duration	Percentage
SSP-L	0.59 s	49 %
DSP	0.10 s	8 %
SSP-R	0.51 s	43 %
Total phase	1.20 s	100 %

Table 3.1: *Duration of the different phases during straight walk. The results are found for three steps for one test subject, which is why there is an asymmetric difference from left and right stride.*

means that the muscles only have a short time interval to guide the body in the right direction. The planning of the motion therefore needs to be precise. It also indicates that the CoM is located outside the stable region of support in most of the walking cycle.

In order to get a more valid data set, it has been chosen to use data from [Borghese et al., 1996], where a number of walking experiments performed on several subjects has been documented. In Table 3.2, the data from the test subject is shown. It should however be noted that, the data of Table 9.1 can not be found directly in [Borghese et al., 1996], but can be derived from the presented data.

	Duration	Percentage
DSP	0.24 s	20%
SSP	0.97 s	80%
Total phase	1.21 s	100%

Table 3.2: *Data for the timing of the phases found in [Borghese et al., 1996].*

Note that the there is only one time for the SSP in Table 3.2, which is because the two SSP's are of equal length. What is also worth noting is that the duration of the cycle for the test object in Table 3.1 and the test subject in 3.2 are the same.

3.2.2 Straight Walking Strategy

The walking cycle of a human test subject, recorded in the test laboratory, is depicted in Figure 3.3 and 3.4 from different perspectives. Figure 3.3 is the gait cycle in the sagittal plane. The first five instances of the sagittal figure, shows how the left leg is swung forward until it is completely stretched. This illustrates that the motion of the leg, while swinging, can be seen as a pendulum. The last five figures show the DSP, lift off, and beginning of the right swing phase. From Figure 3.3**(i)** it should be noticed that as the right foot lifts off, only the left foot supports the body. The upper trunk is behind the supporting foot, and therefore also the CoM. This means that if the body did not use the kinetic energy stored in the movement, it would fall. From Figure 3.3 the posture of the upper body should be noticed. While the legs and arms move in cyclic motions, the upper body, and specially the spinal column, is stable in its motion. As the spinal column is base for the head, the motion of the head is also stable and steady.

In Figure 3.4 the gait cycle is depicted in frontal view . From this figure it can be seen that while walking straight, there is little motion in the frontal plane. It can also be seen that the movement of the spinal column is limited. This emphasizes the fact that

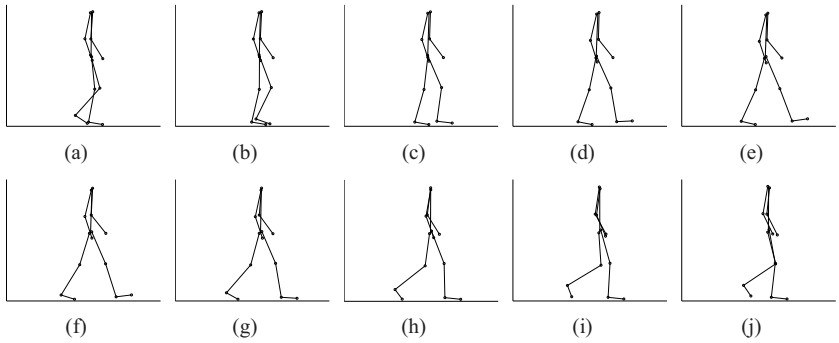


Figure 3.3: *Sagittal view of a gait cycle. Figure **a** to **d** is the SSP-R. Figure **e** and **f** is the DSP and Figure **g** to **j** is the SSP-L.*

when walking, the human body tries to give a steady base for the head.

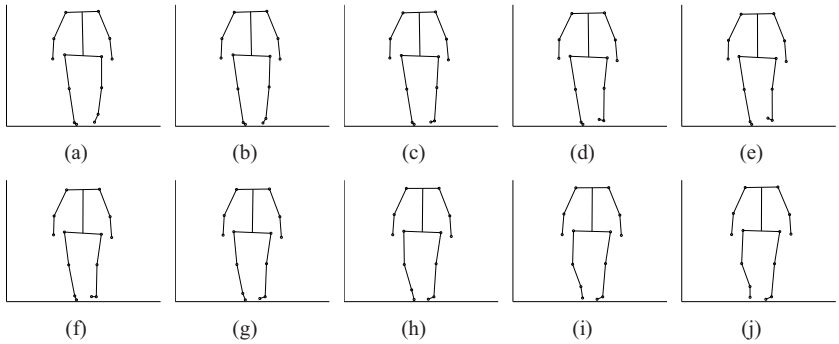


Figure 3.4: *Frontal view of a gait cycle. Figure **a** to **d** is the SSP-R. Figure **e** and **f** is the DSP and Figure **g** to **j** is the SSP-L.*

From Figure 3.3 and 3.4 it can be seen that when walking in a straight line, the strategy of the human body is to maintain stability and upright posture of the upper body. This strategy makes good sense, as the balance sensor is located in the inner ear [Kuo, 1997].

3.3 Curved Gait

The analysis given in this section is inspired from the results of [Courtine and Schieppati, 2004]. The walk of humans is often comprised of several curves, as normal walk almost never consists of only a straight line. It is therefore important to give an analysis of the curved gait as well. In [Courtine and Schieppati, 2004] it is suggested that curved walk is a modification of straight walk including body rotation.

This rotation of the body takes place during the stance phase of the external leg and the rotation, which takes place in the hip, thereby turns the upper body and free limbs with

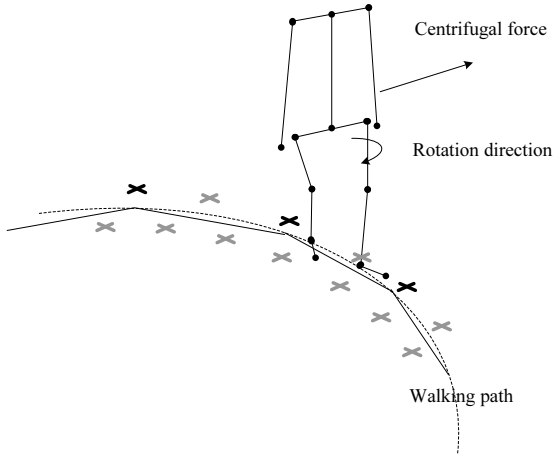


Figure 3.5: *Body rotation results in a centripetal force, when walking along a curved path. Humans segmentalize the curve to ease the turning. The external stride is longer than that of the internal stride. The black crosses is the step where the actual turn is taken place, where gray crosses is a normal straight step.*

regards to the stance leg. When turning on the external stride, the energy added to the system by the centrifugal force is transfered to the rotation of the upper body. Had this rotation taken place on the internal stride, the energy would not be transferred to the rotation of the upper body. Rotation of the body takes place during the toe-off phase, and the rotation of the body results in a rotation of the foot as well. The rotation of the body is larger than the rotation of the foot. Turning the body results in a centrifugal force, and thereby in movement of the body CoM toward the inner foot as the body is leaned inward to counteract the centrifugal force, shown in Figure 3.5. When the velocity of the turn increases, so does the angular momentum of the trunk.

Obviously the stride length of the internal and external foot differ, the internal being the smallest, which is depicted in Figure 3.5. The external stride length does not differ significantly from the stride length at straight walk. When walking a planned curved path, humans mostly change the direction at the beginning of the curve. The body velocity decreases at the beginning of the curved path until reaching a velocity fit for the turning radius of the curve. The larger the turning radius of the curve the smaller the velocity of the body.

If the curved path has a sharp turn, humans tend to divide the path up in segmented pieces, and thus not walk in a continuous path. This results in small jerks in body direction when walking curved paths, and thus also fast changes in the angular acceleration of the body.

As it is suggested that the curved gait is similar to that of straight gait modified with angular rotation, the body posture must be somewhat similar to that of the straight walk. But as the velocity and turning ratio increases so does the tilt of body, as the CoM is moved toward the center of rotation to counteract the centrifugal force. Therefore it is assessed that when walking in a curved path it is both important to keep a straight and steady posture of the upper body, but at the same time it is also important to adjust

the tilt of the posture to follow the centripetal force. This means that the posture not necessarily needs to be upright as well.

3.4 Thesis Objectives

In Section 3.1 the overall objective of this thesis was found to be:

> *Develop a humanoid platform and through this investigate how human-like walk can be implemented on this platform.*

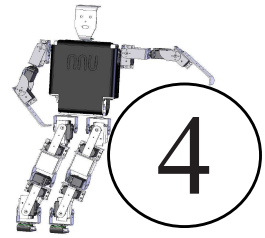
In the analyze given in this chapter, the human walking properties for straight and curved walk were given. It was stated that human-like walk exploits the gravity in its dynamic movements. Humans fall out of the SSP, and the walk can therefore be categorized as unstable, but still they walk in a stable manner, as the fall is caught by the swing leg.

The overall objective is now outlined:

- As the walk should resemble human walk, the proportions of the robot platform should resemble human proportions, thus a humanoid robot must be developed
- The robot platform should be independent of external power supply and external processing power
- The architecture of the robot platform should resemble that of Figure 3.1, meaning that:
  - A trajectory planning and pattern generator should be generated
  - The planned trajectories should exploit the dynamics, but still ensure that the robot walks in a stable manner
  - Sensor feedback in the manner of humans should be provided
  - Control, based sensor feedback, should be used to maintain stability

The above outlines the main objectives of this thesis, and the following will be a documentation of how these objectives are achieved.

MECHANICS



Chapter contents

IN THIS CHAPTER *the design of the mechanics is described. The robot is designed to match human proportions, to make it possible to resemble human walk. The chapter therefore starts out with a description of the human body, which can be found in Section 4.2. In Section 4.3 the design og the special multi-DoF joint is explained. The torso is designed such that on-board computer, servo motors and batteries can be mounted, this is explained in Section 4.4. Next, in Section 4.5 the design of the arms and legs is explained, and last Section 4.5.3 holds a description of the special foot design, that is constructed such that the force distribution can be measured. SolidWorks is used to create a 3D-representation of the robot.*

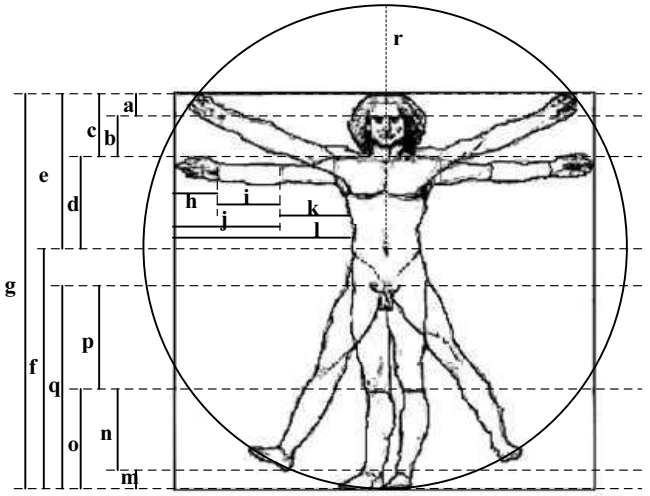


Figure 4.1: *The human proportions depicted in Leonardo da Vinci's Vitruvian Man [Gielo-Perczak, 2001]. When a human stretches out the arms, the body makes a square consisting of the legs, arms and head. The distance from the navel to the feet, makes the radius of the circle which encircle the body.*

4.1 Introduction to Mechanical Design

As it was stated in Section 3.4 on page 20 on of the thesis objectives was to construct a humanoid robot, on which the architecture of Figure 3.1 on page 14 could be implemented. On this humanoid robot it should be possible to implement human-like walking patterns, and the robot should therefore be constructed with human proportions. This chapter describes the construction of the mechanics for this humanoid robot, and first is an introduction to the human proportions given.

4.2 Description of the Human Body

This section is inspired by [Popovic, 2006] and [Gielo-Perczak, 2001], and will serve as a base for the design of the biped robot. When designing a robot that should resemble human walk, an analysis of the human body must be made. This section gives a description of the human proportions. The human proportions is used to design the mechanical parts of the biped robot, as described later in this chapter. Leonardo da Vinci's Vitruvian man, depicts the human proportions, as shown in Figure 4.1. He discovered that there was a connection between the proportions of different limps of the human body and the number phi:

$$\varphi = \frac{1}{\varphi} = \frac{1}{1.618} = 0.618 \tag{4.1}$$

If a circle is drawn with center in the naval of the body, and a radius going from the naval to the feet, this circle encircles the entire body. A square, consisting of the span of the arms and the full body height (which is equal), is drawn. Then the ratio between the radius of the circle and the length of a side in the square equals φ. These proportions are depicted in Figure 4.1.

Body segments (A:B)	A [cm]	B [cm]	[A:B]	ϵ [%]
a:b	11.9	19.9	0.596	3.24
b:c	19.9	29.8	0.668	8.09
c:d	29.8	39.8	0.749	21.2
d:e	39.8	69.6	0.572	7.44
e:f	69.6	105.5	0.660	6.8
f:g	105.5	175	0.603	2.43
h:i	20.1	25.9	0.776	25.57
i:j	25.9	45.8	0.566	8.41
k:j	29.8	45.8	0.651	5.34
j:l	45.8	73.6	0.622	0.65
m:n	21.9	37.8	0.579	6.31
n:o	37.8	59.7	0.633	2.43
p:o	41.8	59.7	0.700	13.27
o:q	59.7	95.5	0.625	1.13

Table 4.1: *The ratio between body segments. ϵ is the percentage deviation from $\varphi = 0.618$ [Gielo-Perczak, 2001].*

In [Gielo-Perczak, 2001] the proportions shown in Figure 4.1, has been analyzed statistically. The results are shown in Table 4.2, and it it seen that there is a close statistic relation between φ and the ratio of different body segments.

One of the goals of the project is to design a biped robot that is able to perform dynamic walk, resembling humans as much as possible. In doing so the efficiency of bipedal locomotion can be very high as shown by [Collins et al., 2001] where a three dimensional passive dynamic walker with human leg proportions was constructed. Human-like stable gait with low power consumption was achived.

In order to achieve this the following key points have been determined and should be emphasized in the design:

- **Resemble the proportions of a human, that is, the length of the limbs should follow the ratios found in humans**
  The factor φ in Equation (4.1) describes the proportions found in humans.

- **Joints on humans with more than one degree of freedom (DoF) in the same plane should be designed as such on the robot**
  To compare the robot and a human the joints should be alike thereby making it possible to directly compare trajectories.

- **Under own power carry batteries and electronics**
  The robot has to be designed to house the batteries and electronics somewhere in the body such that the robot can be autonomous.

The following will describe how the biped robot has been designed. The parts in the design will not undergo a stress analysis to determine if they are strong enough under the applied load. Instead it is intuitively assessed if a part is strong enough. First is the design of the ankle and hip joint constructed.

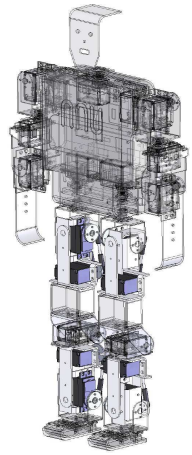


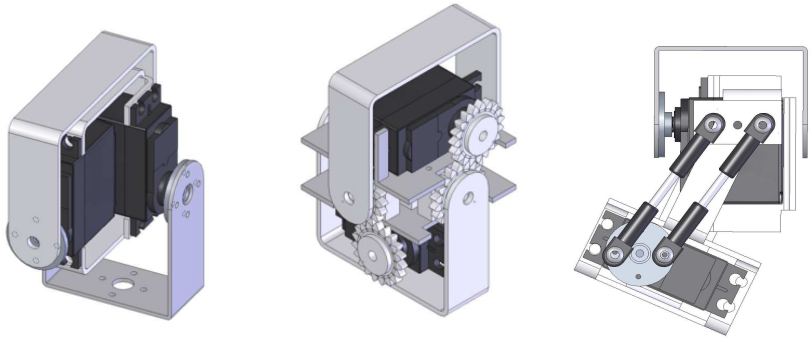
Figure 4.2: *Placement of the multi DoF joint in the complete construction.*

4.3 Multi Degree of Freedom Joint

This section gives a description of the design of the multi degree of freedom (DoF) joint. This joint fits into the complete construction as seen on Figure 4.2, where it is the part which is not transparent.

In a human the joint in the hip and ankle have more than one degree of freedom in the same plane. The joint in this section can be used both in the ankle and hip implementing movement in the x and y-direction. It has been chosen to use standard-sized ($4\,\mathrm{cm} \times 2\,\mathrm{cm} \times 3.6\,\mathrm{cm}$) servomotors as it is assessed that they have a good size to torque ratio.

One approach could be to design brackets that place the shafts of two servo motors in the same plane as shown on Figure 4.3(a). However, this design is asymmetric and wide which to some extent clashes with the desire to make the robot human-like.



(a) Joint design1 (b) Joint design 2 (c) Joint design 3

Figure 4.3: *Three different joint designs*

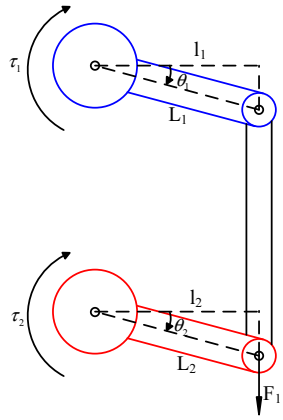


Figure 4.4: *One of the levers of joint design 3. Used to determine the transmission ratio of the torque.*

To make the joint symmetric gears can be used as shown in Figure 4.3(b). However, this design is fairly complex and more prone to backlash, introduces by the gears, than the previous design. Figure 4.3(c) shows the design that was used in the biped robot. The design is rather compact, making it ideal for implementation on the robot, and any additional backlash added to the joint is limited to the axis driven by the levers. By using two levers the strain on the horn will be symmetric about the shaft thereby not adding additional strain lengthwise to the shaft which would have been the case if only one lever was used.

To show that using the design of Figure 4.3 the energy lost in the transfer using the levers, can be considered zero, a small example is given. This principle is shown in Figure 4.4 where the force acting on servo horn one is τ_1.
To determine τ_2 the force F_1 has to be determined:

$$F_1 = \frac{\tau_1}{l_1} \tag{4.2}$$

Where l_1 is equal to $\cos(\theta_1)L_1$ because only the vertical component of the force is considered. Because of this the torque τ_2 can be determined using F_1:

$$\tau_2 = F_1 L_2 \cos(\theta_2) \tag{4.3}$$

$$\tau_2 = \tau_1 \frac{L_2 \cos(\theta_2)}{L_1 \cos(\theta_1)} \tag{4.4}$$

As the arms connected to the levers are the same length in both ends Equation (4.4) reduces to:

$$\tau_2 = \tau_1 \tag{4.5}$$

Thereby Equation (4.5) states that there is a one-to-one transmission of the torque, neglecting the loss due to friction in the bearings on the levers.

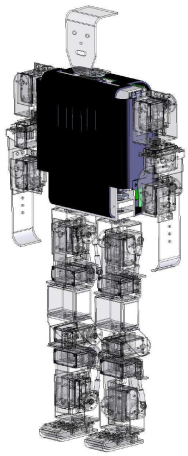


Figure 4.5: *Placement of the torso in the complete construction.*

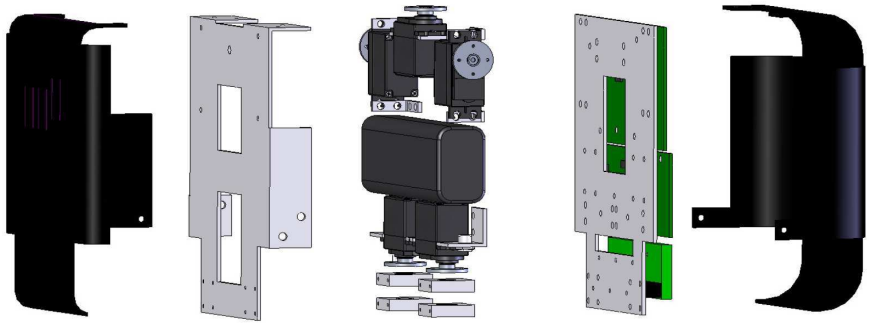


Figure 4.6: *An exploded view of the torso showing the interconnection of the parts.*

It is therefore selected that the multi DoF joint depicted in Figure 4.3c should be used as the ankle and hip joint. Next is the design of the torso given.

4.4 Torso

The design of the torso is the base of the robot. It connects all limbs and further holds the on-board computer and batteries. The placement of the torso as connection joint is seen from Figure 4.5 as the part which is not transparent.

The torso needs five servo motors to drive the limbs, one for each leg and arm plus rotation of the head. This puts a lower limit on how small the torso can be. Therefore the torso is designed first and the limbs are designed to match the size of the torso, so that all match the human proportions given in Section 4.2.

The torso has a single aluminum part, which the servo motors and the on-board electronics are attached to. This aluminum part is seen from the exploded view of the torso depicted in Figure 4.6.

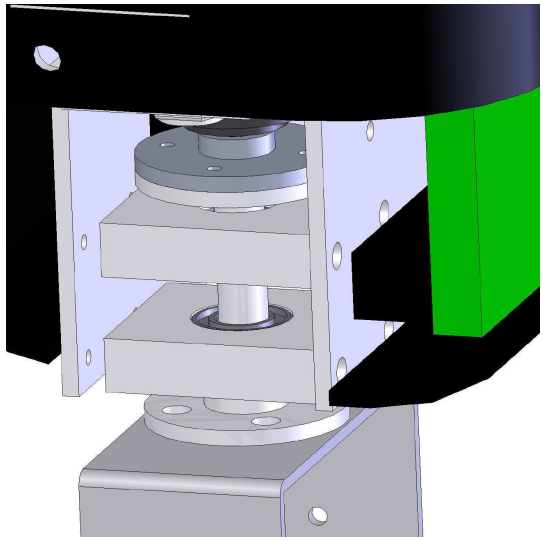


Figure 4.7: *A close up on the additional shaft located in the torso.*

Two of the servo motors in the torso implement the leg motion in the z-direction, but the leg can not be connected directly to the shaft as this would be to weak. Instead an additional shaft is implemented in the torso that offloads the shaft of the servo motor. This is depicted on Figure 4.7.

It has been chosen to enclose the battery inside the torso. As the battery is one of the heavier parts of the biped robot it can also have a great influence on the centre of mass of the torso. Therefore placing it outside the torso gives an asymmetric distribution of the CoM on the finished biped robot. To enclose the battery and stiffen the torso, an aluminum plate is designed which fits the holes on the ball bearing sockets and distance pieces that are connected at the shoulders. At the top and sides of the frontal torso plate, pieces of the aluminum plate is bend in a $90\,\mathrm{deg}$ angle. The piece to the right closes the battery compartment in one end and the piece on the left side is shorter so the battery can be inserted in the compartment. The pieces on the top of the plate closes the shoulders. The large extruded cuts of frontal and back plate reduces the weight and eases the routing of cables. On the back plate the on-board electronics is mounted. As it is considered inevitable that the biped robot falls the electronics is placed behind a plastic shield to avoid damage. This shield is a prototype made using a selective laser sintering (SLS) technique, and should therefore be stronger than ordinary plastic constructions. To give the biped robot a more appealing appearance a shield, covering the chest, is also designed. This also gives the possibility to house even more electronics if this is necessary on future revisions of the biped robot. All the parts of the torso can be seen from Figure 4.6.

4.5 Dimensioning the Limbs

As the design of the torso has been completed, the dimensions found can be used to design the additional limbs of the robot. These are seen from Figure 4.8 including the

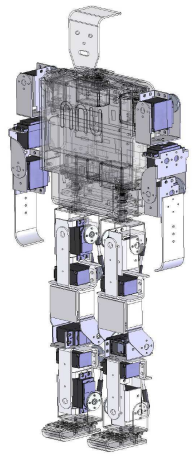


Figure 4.8: *Placement of arms and legs in the complete construction.*

multi DoF joint.

In the following section the legs, arms and feet will be dimensioned using known properties of the human body as described in Section 4.2 on page 22. The dimensions of the torso will be used as the starting point to dimension the limbs as the height of the torso is known. First is the complete proportions of the legs designed.

4.5.1 The legs

According to Section 4.2 on page 22 the average human body is $7 - 7.5$ heads tall. Of this the torso is about 3 heads tall and the legs are the remaining 3 heads which gives a relation between the known torso height and the unknown length of shin and thigh. Furthermore, it is known that dividing the length of the thigh, L_{thigh}, with the length of the shin, L_{shin}, equals φ. Knowing the length of the legs makes it possible to dimension the thigh and the shin in the following way:

$$\varphi = \frac{L_{\text{thigh}}}{L_{\text{shin}}} \tag{4.6}$$

As mentioned the height of the torso, L_{torso}, can be expressed as:

$$L_{\text{torso}} = L_{\text{thigh}} + L_{\text{shin}} \tag{4.7}$$

Combining Equation (4.6) and (4.7) gives:

$$\varphi = \frac{L_{\text{thigh}}}{L_{\text{torso}} - L_{\text{thigh}}} \tag{4.8}$$

$$L_{\text{thigh}} = \frac{\varphi L_{\text{torso}}}{\varphi + 1} \tag{4.9}$$

Using the property of the powers of φ: $\varphi^{n+2} = \varphi^{n+1} + \varphi^n \rightarrow \varphi^2 = \varphi + 1|_{n=0}$:

$$L_{\text{thigh}} = \frac{\varphi L_{torso}}{\varphi^2} \tag{4.10}$$

$$L_{\text{thigh}} = \frac{L_{\text{torso}}}{\varphi} \tag{4.11}$$

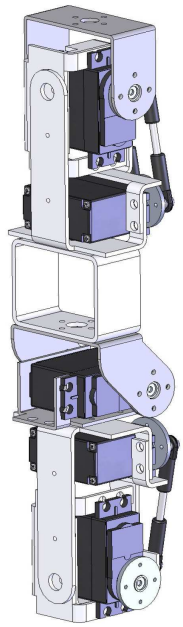


Figure 4.9: *The left leg shown assembled.*

The height of the torso is determined to be $L_{\text{torso}} = 216\,\text{mm}$. Using equation 4.11 and 4.7 L_{thigh} and L_{shin} can be determined:

$$L_{\text{thigh}} = \frac{216\,\text{mm}}{1.618} = 133.5\,\text{mm} \tag{4.12}$$

$$L_{\text{shin}} = 216\,\text{mm} - 133.5\,\text{mm} = 82.5\,\text{mm} \tag{4.13}$$

From Figure 4.9 the length of the multi-DoF joint is $71.57\,\text{mm}$ leaving the shin $82.5\,\text{mm}$ $-71.57\,\text{mm} \approx 11\,\text{mm}$ to short and the thigh $133.5\,\text{mm} - 71.57\,\text{mm} \approx 63\,\text{mm}$. To design the shin an with correct proportions a servo bracket is mounted on the multi-DoF assembly to give the knee joint. The bracket adds $14\,\text{mm}$ to the shin which gives an error of $3\,\text{mm}$ or approximately $3.5\,\%$. This is, however, accepted due to the uncertainty connected with the human proportions. To complete the knee another bracket is attached to the shaft of the servo motor in the knee. This adds $25.22\,\text{mm}$ to the length of the thigh leaving it $63\,\text{mm} - 25.22\,\text{mm} \approx 38\,\text{mm}$ to short. This is added by adding an extension part between the hip joint and the knee, as seen on Figure 4.9 along with the other mentioned brackets used in the leg.

4.5.2 The Arms

The arms on the robot are solely used to shift weight which means that the arms will not have hands that can manipulate the surroundings. Therefore it is enough to give the arms two degrees of freedom in the shoulder. However, it has been chosen to give the arms four degrees of freedom so that the "fingers" can be placed arbitrarily within its reach making a redesign of the arms easier should it be equipped with hands. Standard servo brackets from Lynxmotion [`http://www.lynxmotion.com`] will be used where ever possible. Figure 4.10 shows the design of the left arm. All brackets are standard brackets from Lynxmotion except what makes up the lower arm and the hand.

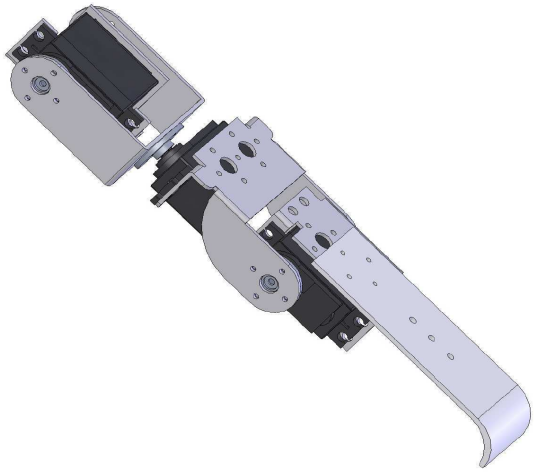


Figure 4.10: *The arm shown assembled*

This part has been fitted with additional holes where weights can be mounted in order to increase the momentum of the arm.

4.5.3 The feet

The feet can either be passive or active. An active foot is able to bend its toes actively meaning that the toe joint is actuated. A passive foot is not actuated in the toe joint, and it is therefore not possible to control the bending of the toe. An active foot more closely mimics the human foot but becomes heavier and bigger, because of the added servo motor. Because the toes are used actively during human locomotion the foot should be active.

However, making the foot active also clashes with the design goals as it would make it larger than what is considered acceptable in regards to the human proportions. This has given the following compromise in the design; the foot is passive however the toes are always bend. This shortens the foot making the task of keeping the balance of the biped robot harder when standing still because the polygon of support is reduced. Figure 4.11 shows an exploded view of the foot including the base of the foot where the toes are bend in a curve.

This enables the robot to "roll" onto the toes instead of bending them. To determine the correct size of the feet the members of the project group have measured their feet. The length from the heel to approximately the middle of the first joint of the big toe on the right foot is measured. This is used as the length of the foot. To determine the width of the foot the widest part of the foot is measured. The length and width are both divided by the test subjects height to give ratios that can be used in the design of the feet. The measurements are shown in Table 4.2.

The width and length of the robots feet can now determined using the height of the robot which from Solid Works is determined to be $58\,\mathrm{cm}$. This yields:

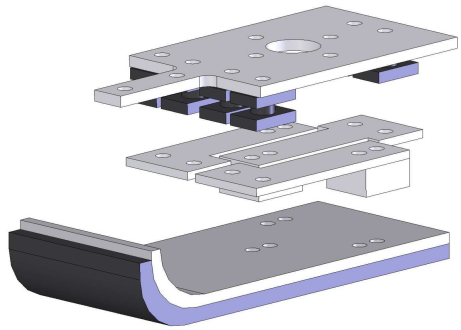


Figure 4.11: *Exploded view of the foot, including all the part which it consists of.*

Height(H)[cm]	Width(W)[cm]	Length(L)[cm]	W/H	L/H
173	9.1	16.5	0.0526	0.0954
175	8.5	17.4	0.0486	0.0994
186	9.7	18.8	0.0522	0.101
172	8.4	17.6	0.0488	0.102
	Average:		**0.05055**	**0.09945**

Table 4.2: *Length, without the toes, and width of the right foot of the group members.*

$$l_{\text{foot}} = 580\,\text{mm} \cdot 0.09945 = 57.681\,\text{mm} \approx 58\,\text{mm} \tag{4.14}$$

$$w_{\text{foot}} = 580\,\text{mm} \cdot 0.05055 = 29.319\,\text{mm} \approx 30\,\text{mm} \tag{4.15}$$

In order to uniquely determine the centre of pressure (CoP) three pressure points has to be measured. Figure 4.11 shows the foot where three plates, called strain plates, are attached, on which the strain will be measured. In order to distribute the forced applied to the entire foot to the strain plates a solid plate, the top plate, has to be mounted to the strain plates. An ideal strain gauge is only sensitive to strain in the direction it is designed to measure. However the strain gauges will be influenced by strain perpendicular to the direction that is to be measured. If pressure is placed on one strain plate the other strain plates are twisted which is referred to as mechanical cross talk. In order to minimize cross talk the top plate is mounted to the strain plates through a membrane which is similar to the feet discussed in [Löffler et al., 2004]. The membrane also functions as a spring/damper system which reduces the nonlinearities associated with the impact phase as suggested by [Buschmann et al., 2006] and [Collins et al., 2001].

The dimensions of the strain plates has been determined with the exception of the thickness and the material. A feasible choice for the material is steel and aluminum as these are available. It has been chosen to use spring steel due to a higher elasticity than aluminum and ordinary steel. The spring steel is able to bend and then return to the initial position without any memory of the bend.

It should, as a minimum, be possible to measure the weight of the robot. The robot weighs $m_{\text{robot}} = 3.7\,\text{kg}$ or $36.3\,\text{N}$. However, the robot could have accelerations that makes it necessary to measure above $36.3\,\text{N}$. Therefore the minimum measurable

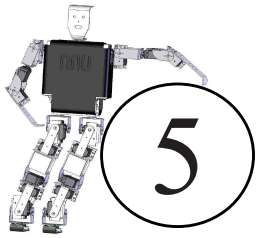
weight has been set to $2m_{\text{robot}} = 7.4\,\text{kg}$. Each strain plate should therefore be able to measure $m_{\text{min,meas}} \frac{1}{3} 7.4\,\text{kg} \approx 2.5\,\text{kg}$. The system that measures the force is designed and tested in Appendix D on page 191.

This completes the design of the foot and thereby the design of the robot.

4.6 Partial Conclusion to Mechanical Design

In this chapter a complete mechanical solution for a biped robot with human proportions was given. Several mechanical parts were custom designed, while some relied on available design from external manufactures. A multi DoF joint was designed which enabled three axis rotation in one point. This multi DoF joint was similar to that of the human hip, thus enabling movements in all directions. Thereby it was possible to make the robot turn as well. Furthermore a custom made foot was designed with possibility to measure the weight distribution. The design also made sure that an on-board hardware solution could be implemented, including a power supply. A SolidWorks drawing of the complete humanoid robot can be seen from Appendix M.

HARDWARE



Chapter contents

THIS CHAPTER *gives a description of the entire hardware system. To realize a wireless, fully controllable and autonomous robot, an on-board computer needs to be mounted on the robot. To retrieve information from the servo motors and other sensors located on the robot, an interface print is developed. This interface print gathers all sensor signals, and sends them to the on-board computer. This chapter starts out with an overall system description, and will then describe the chosen sensors. Afterward the actuators will be described. This chapter will also include a description of how the system is powered.*

5.1 Hardware Overview

This section gives a description of the entire hardware system, located on the robot. Each hardware module will be given a short description and a reference will be given to the specific section describing that module more detailed. Figure 5.1 shows the entire system, and is used as reference figure in this section.

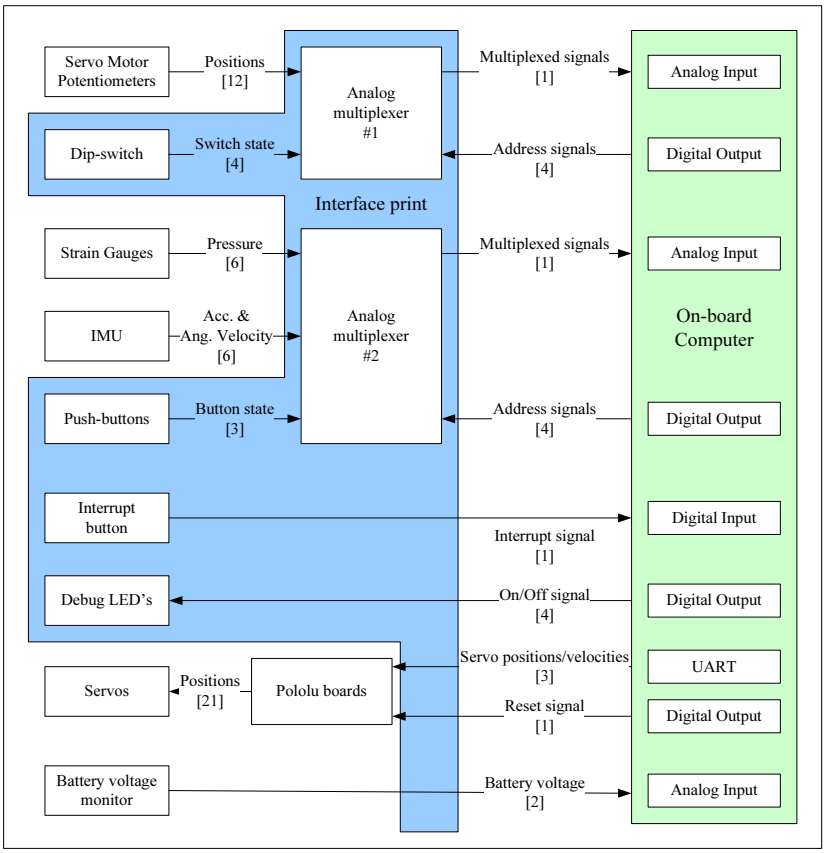


Figure 5.1: *Hardware overview, showing the entire system. The number in the brackets refers to the number of signal wires. The on-board computer is shown in the right side of the figure, and marked with a green box. The interface print is marked with a blue box and is shown in the left side of the figure. Boxes that are located outside the interface print and the on-board computer are connected to the interface print using a number of connectors. As seen all external hardware is connected to the interface print, even if the signal should just be routed unchanged to/from the on-board computer.*

To collect information on the state of the system, a number of sensors is placed on the robot. As seen in Figure 5.1 the sensors are strain gauges, IMU, and servo potentiometers. These will be described in Section 5.2. Further more, to move the robot a number of actuators must be inserted which is explained in Section 5.3. The system consists of an on-board computer that controls the whole system, this is described more detailed in Section 5.4. The on-board computer is connected to the interface print using a 40

pin connector cable. The interface print contains several electronic circuits used to gather measurements from the large number of sensors and actuators, providing communication between the on-board computer and the surroundings. The interface print is described in Section 5.5.

5.2 Sensor Determination

For the purpose of getting feedback from the biped robot, and therefrom generating control signals, several sensors are available. In this project, the choice of sensors is based on experience from [Christensen et al., 2006] and studies of human gait, described in Section 3.1. In [Christensen et al., 2006], it was chosen to use accelerometers and force sensing resistors (FSR). The accelerometers were used for fault detection purpose, to determine if the robot was about to fall. The FSR's were used to determine the weight distribution on the feet. This weight distribution was used as control input for a fuzzy logic controller.

5.2.1 Strain Gauges Mounted on the Feet

Humans use signals from the receptors under the foot sole to control their stability, and based on this it is chosen that the robot should provide some kind of feedback on the force distribution under the feet. In [Christensen et al., 2006] they implemented FSR to obtain the feedback, experience with FSR's as the only control input however showed two major problems. These problems were:

- **Drift in signals:** When exposing the sensors to a static load, i.e. a solid mass, the output from the sensors drifted, meaning that the measurement could not be used to determine the exact load over time. The output started to drift immediately after the mass had been placed on the sensor and more than a minute passed before a steady-state was reached.

- **Poor repeatability:** If the same load was applied to the FSR several times, the output deviates significantly compared to each other. Hence, the reliability of the sensors was not as good as expected, and not suitable for the purpose.

Based on these experiences, it has been concluded that FSR's are not applicable in this project. Load cells are not considered an option due to their high price and large physical size. It could be a problem to mount several load cells on such small feet as those used in this project. This narrows the choice down to strain gauges. These have been chosen due to their high precision, low price and high reliability. In order to retrieve the information by measuring the change in resistance, a strain gauge print has been developed. This print consists of three Wheatstone bridges, that converts the resistance change into a voltage signal. This signal is amplified an filtered, and then led to the interface print. A strain gauge print is mounted on each foot, to reduce the amount of noise on the signals. Figure 5.2 shows a picture of the foot with the strain gauge print mounted on top of it. A more detailed description of the development of this print can be found in Appendix D on page 191.

5.2.2 Potentiometer Signals

To obtain further feedback from the robot, it has been chosen to modify the servo motors in the legs, to provide the internal potentiometer signal. This modification is

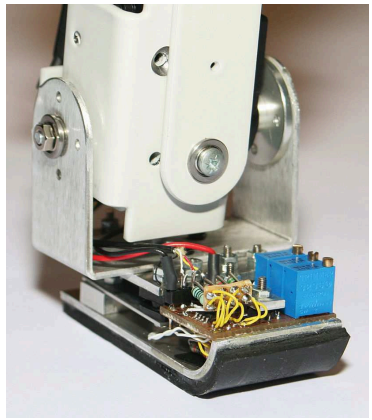


Figure 5.2: *Picture of the right foot. On top of the foot, the strain gauge print can be seen.*

described in Appendix B on page 175. The potentiometer signal provides the actual position of the servo motor shaft and is used for internal feedback inside each servo motor. It has been chosen to use this signal as control signal due to experience from [Christensen et al., 2006] where the load from the robot made the servo motors change their position. It was concluded that the internal controller is a P-type controller, meaning that the servo motors not necessarily reached the correct position when being exposed to a load. The potentiometer signals from the arms have not been extracted, due to the fact that the load in the arm joints is small compared to the load in the leg joints.

5.2.3 Inertia Measurement Unit

When a human walks, the stability is controlled by using signals gathered from the inner ear, as explained in Section 3.2.2 on page 17. The goal of this project is to obtain human-like walk. To resemble the human perception, it has therefore been decided to place a five DoF inertia measurement unit (IMU) in the head of the robot. In Figure 5.3 a picture shows the location of the IMU.

The IMU consists of a small printed circuit board (PCB), where an IDG300 dual-axis gyroscope [InvenSense Inc, 2006] and an ADXL330 three-axis accelerometer [Devices, 2006] are mounted. The output from the IMU is three analogue signals describing the acceleration of the unit in all three axis, and two analogue signals describing the angular velocity of the unit in terms of the roll and pitch. The analogue signals are routed directly to the interface print, where they are routed to the analogue multiplexer.

5.3 Actuators

As actuators for the biped robot it has been chosen to use two different servo motors. The legs are exposed to high torques since they will be carrying the body during walk. The servos in the knees will be exposed to a load during stance, due to the design of the knee. However the servo motors in the arms only have to move their own weight, as this project focuses on the walk of the biped and the arms are only used as stabilizing factor by moving masses.

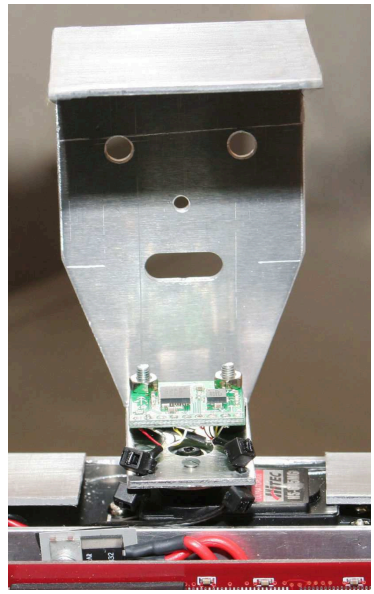


Figure 5.3: *A picture of the head of the robot. Below the mouth, the IMU is seen.*

5.3.1 Servo Motors

The servo motors where chosen based on experience gained from [Christensen et al., 2006]. In [Christensen et al., 2006, pp. 206] applying the designed walking patterns showed that some of the servo motors where unable to deliver the required torque in all phases. This biped robot was $30\,\mathrm{cm}$ high and weighed $1.7\,\mathrm{kg}$ and the maximum torque required was found to be $1.1\,\mathrm{Nm}$. The biped robot designed in this project is $58\,\mathrm{cm}$ high and weighs $3.6\,\mathrm{kg}$, which means that the servo motors have to be even stronger. In [Santos and Silva, 2005] a biped robot was developed that weighed $5\,\mathrm{kg}$ and was $60\,\mathrm{cm}$ high. The model of this biped showed a maximum torque of $2.6\,\mathrm{Nm}$. With the above mentioned in mind, it is chosen to equip the legs with the HSR-5995TG from Hitec which has a maximum torque of $2.9\,\mathrm{Nm}$. Additionally the HSR-5995TG uses titanium gears which makes it more durable, compared to steel gears. The simulated walking pattern in [Christensen et al., 2006, pp. 207] showed that the required torque in the arms never exceeded $0.04\,\mathrm{Nm}$. The arms on the biped robot related to this project are longer but lighter, they are, however, designed so that the weight can be increased, should this be necessary. So to ensure that added weight can be moved satisfactory the HS-645MG servo motor from Hitec, which has a maximum torque of $0.92\,\mathrm{Nm}$, is placed in the arms. The specifications for the two servo motors are shown in Table 5.1.

5.3.2 Servo Controllers

This section seeks to describe the interface to the servo motors, and explains how the servo controller boards work. It defines the protocol used to control the servos. The source of information is the data sheets for the servo motors [HiTec, 2005] and [HiTec, 2006], as well as the manual for the servo controller board [Robotics and Electronics, 2005].

	HSR-5995TG	HS-645MG
Operating voltage	7.4 V	6 V
Running current	380 mA	450 mA
Stall current	5.2 A	N/A
Standing torque	2.9 Nm	0.92 Nm
Stall torque	3.8 Nm	N/A
Pulse rate	$\frac{90\,\text{deg.}}{400\,\mu\text{s}}$	$\frac{45\,\text{deg.}}{400\,\mu\text{s}}$
Pulse range ± 90 deg.	$1100\,\mu\text{s} - 1900\,\mu\text{s}$	$700\,\mu\text{s} - 2300\,\mu\text{s}$
Neutral	$1500\,\mu\text{s}$	$1500\,\mu\text{s}$
Dead band width	$2\,\mu\text{s}$	$8\,\mu\text{s}$
Speed (No load)	8.73 rad/s	5.22 rad/s
Clock wise direction	Increase pulse period	Increase pulse period
Gear	Titanium	Steel/nylon
Weight	62 g	55.2 g

Table 5.1: *Specifications for HSR-5995TG and HS-645MG. Values indicated are maximum ratings from [HiTec, 2005] and [HiTec, 2006].*

Motor Interface Description

The position of each servo motor is determined from the width of the pulse given to the servo motor. A pulse of $1500\,\mu\text{s}$ sets the middle position. For the HS-645MG the range is $800\,\mu\text{s}$ to each side, which corresponds to ± 90 deg.. The HSR-5995TG uses a pulse at half the time, to set a similar position. The servo control signal is updated with 50 Hz, i.e. every 20 ms.

This section describes the control of a HS-645MG. Same description goes for the HSR-5995TG if the pulse widths are divided by two. On Figure 5.4 the servo control signal for the HS-645MG is depicted. The working range of the motor is illustrated with the dashed lines at $700\,\mu\text{s}$ and $2300\,\mu\text{s}$, corresponding to ± 90 deg. respectively. The maximal turning rate, at 6 V supply, is 5.24 rad/s. To create these control signals, a

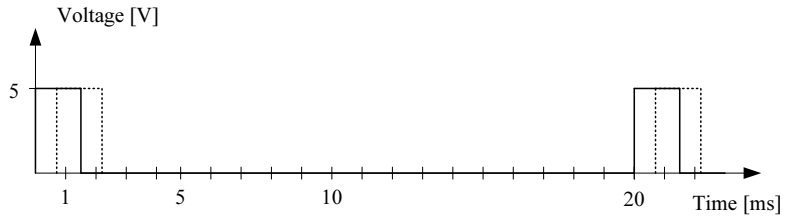


Figure 5.4: *The control signal to the servo motors. The solid line corresponds to a position of* 0 deg.. *The dashed lines show the pulse widths corresponding to a position of* ± 90 deg..

Pololu servo controller board is used. The board is very small in size, only $23\,\text{mm} \times 23\,\text{mm}$, and capable of controlling up to 8 servo motors. A further description of the servo controllers and the motor interface is given in Appendix A on page 173.

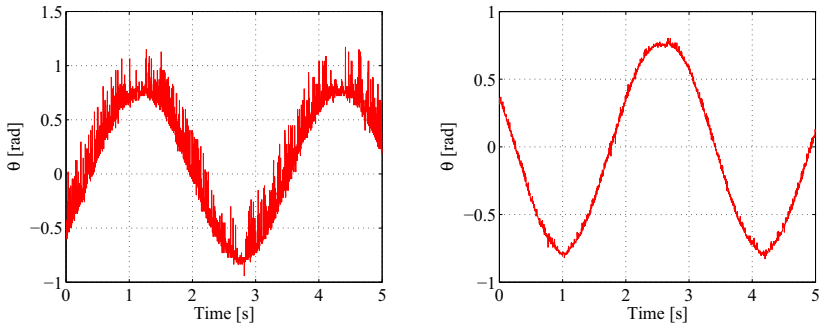
5.3.3 Feedback from Servo Motors

This section will concern the extraction of the position measurement from potentiometers inside the servo motor. It was determined in Section 5.2.2 on page 35 that these

signals should be provided.

From the HSR-5995TG it is possible to reach the potentiometer inside the servo motor, allowing feedback from the servo motor. In Appendix B on page 175 it is explained how the servos are dismantled, and re-engineered to provide the feedback.

It is now possible to measure the position of the servo motor, which was tested by applying a sine wave as an input. The measured signal from the potentiometer can be seen in Figure 5.5(a). The signal contains a significant level of noise, since the signal to noise ratio (SNR) is only 14.1.



(a) Unscreened, SNR $= 14.1$ (b) Screened and filtered, SNR $= 102.5$

Figure 5.5: *Measurement from the potentiometer, where a sine wave is given as a reference for the servo motor. 5.5(a) shows the raw measurements, no filtering, no screen etc. In 5.5(b) the signal wire is screened and filtered through the filter in Equation* (5.1).

The relative large noise contribution on the signal is most likely generated by the servo motor, which generates noise while turning. Furthermore the signal wire runs close to the discrete control signal from the servo motor driver boards. In order to shield from this imposed noise, a wire with a grounded screen is used for the position measurement from the servo motor. This improves the measurements, and the SNR can now be calculated to 37.3, which is an improvement of 165%. A commmon mode instrumential amplifier is implemeted, and the potentiometer signal is the filtered using a second order Sallen and Key filter described in Equation (5.1).

$$\frac{V_o(s)}{V_i(s)} = \frac{5050}{s^2 + 100.5s + 5050} \tag{5.1}$$

This improves the SNR to 102.5, compared to the unscreened and unfiltered signal of Figure 5.5(a), this is an improvement of $626.9\,\%$.In Figure 5.5(b) the screened and filtered signal can be seen. The filter in Equation (5.1) is implemented in hardware, and placed on the interface print, where it is applied to each of the twelve potentiometer measurements. The complete arangement can be seen in Figure 5.6.

A more thorough description of the extraction of the measurements and the derivation of the filter is seen from Appendix B on page 175.

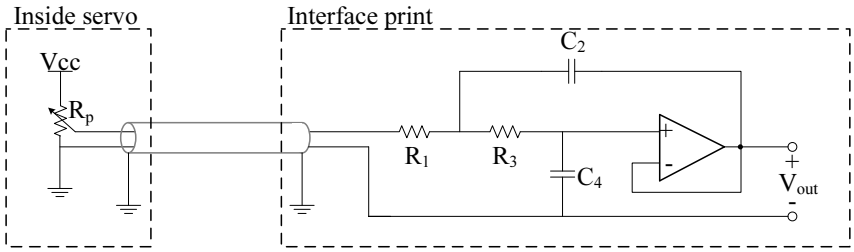


Figure 5.6: *The realization of the Sallen and Key filter, including the screened wire. The values of the components can be found in Appendix B.*

5.4 Choosing an On-board Computer

This section describes the on-board computer that has been chosen for this project. Several micro processor developer boards are available on the market, each with their own advantages and disadvantages. It has been chosen to use a TS-7400 from Technologic Systems, [Technologic Systems, 2006], due to the following reasons:

- **Fast 200 MHz ARM9 processor with floating point unit(FPU):** The ARM9 processor, from Cirrus Logic, is an EB9302 with an integrated math engine. This math engine provides hardware floating point operations, which is far more effective than processors that emulates an FPU in software.

- **On-board 1/10/100 Mbit LAN:** A standard LAN-connector is located on the PCB, allowing easy interfacing and communication with the on-board computer.

- **Two USB 2.0 compatible connectors:** These can be used to interface standard USB hardware. One of these ports is used to connect a WLAN USB dongle (802.11G) to provide a wireless connection between the Simulink PC and the robot.

- **SD-card socket:** One of the great features of this board, compared to many other developer boards, is the SD-card socket that is located on the PCB. This provides the possibility of installing a complete Debian (or any other) Linux distribution. A 1 GB SD-card has been bought for the purpose. To shorten the time for starting a Debian distribution on the on-board computer, it has been chosen to buy a SanDisk Extreme III SD-card that, at the time of writing, is among the fastest SD-cards available. The transfer rate is specified to $20\,\mathrm{MB/s}$.

- **Three TTL level UARTs:** To position the servo motors, three servo controllers are used. These are interfaced by using one UART, with TTL levels, for each servo controller.

- **Boots Linux out of the box:** A small Linux distribution is located in the flash RAM when receiving a TS-7400. This small Linux distribution makes it possible to get in contact with the board, either by Telnet using the LAN, or by a serial RS232 connection. All hardware located on the board is fully supported by the small Linux distribution, named TS-Linux.

- **Four 12-bit A/D converters are located on the board:** The micro processor board provides four 12-bit A/D converters, that are easily accessed from user code. These A/D converters will be used to collect the measurements from all

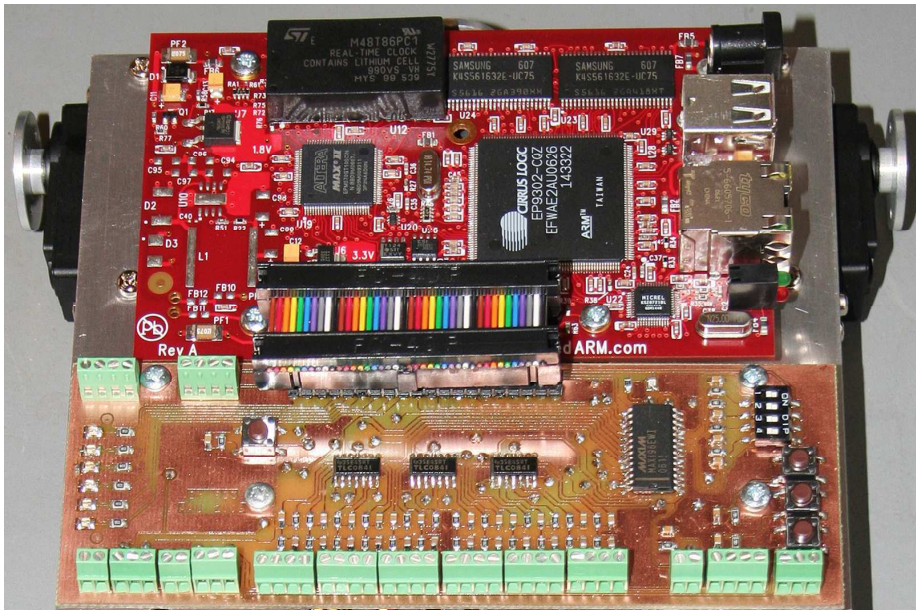


Figure 5.7: *Picture of the on-board computer in the top, and the interface print in the bottom. The two boards are connected with a 40-pin cable, seen in the middle.*

the sensors on the robot, meaning the potentiometer signals, strain gauge measurements, IMU data and battery voltage.

- **20 digital I/O:** The board provides 20 digital input/output that can be accessed directly through the 40-pin connector.

- **Small size:** The board only measures $7.4\,\mathrm{cm} \times 12\,\mathrm{cm}$, making it suitable for the robot.

To interface the on-board computer, the 40-pin connector is used. The interface print is equipped with a similar connector, making it possible to connect the two boards by a single cable. The on-board computer provides a $5\,\mathrm{V}$ supply as well as $3.3\,\mathrm{V}$ supply in the 40-pin connector, which is used to supply all external prints, both the interface print, the strain gauge prints and the Pololu boards. A picture of the on-board computer along with the interface print can be seen in Figure 5.7.

Besides from the key features mentioned in the list, Technologic System provides a wide range of code examples, test programs, Linux distributions and well documented manuals, which ease the development of the robot platform.

5.5 Description of the Interface print

The interface print can be seen in Figure 5.1 and consist of the following components:

- **twelve low-pass filters for the potentiometer signals:** The signals from the potentiometers, located inside the servo motors, are noisy and should be filtered before the signals are usable in a control system. To ease the on-board computer,

these filters are implemented in hardware. Section B.2 on page 178 describes the design of the filters in details. It has been chosen to use a second order Sallen and Key filter. On the interface print, 12 identical filters for the potentiometer signals are placed.

- **two 16-channel analogue multiplexers:** To make it possible to read all analogue signals with only four analogue inputs available on the on-board computer, two 16-channel analogue multiplexers are used. Section 5.5.1 gives a description of the chosen multiplexers. The analogue multiplexers reduce the number of needed analogue inputs from 31 down to 2, for the sensors.
- **one 4-bit dip-switch:** To make it possible to set some parameters permanently externally, a dip-switch with four on/off switches is inserted. The dip-switch is connected to one of the multiplexers.
- **three pushbuttons:** When developing code, it is preferable if it is possible to change some parameters, break code, initiate function calls, etc. by externally pushing some buttons. To realize this, three pushbuttons have been mounted on the interface print. These buttons are connected directly to one of the multiplexers.
- **one interrupt button:** If the code should end up ind a dead-lock, an interrupt switch have been mounted on the interface print. This is connected directly to a digital input on the on-board computer, that can be used as external interrupt.
- **four light emitting diodes (LED's):** For debugging purpose, four LED's has been place on the interface print. The state of the LED's can be changed independently of each other.
- **Connectors for external modules:** The strain gauges and the IMU are connected to the interface print by using the corresponding connectors. On each foot, an amplifier circuit to the strain gauges measurements is located. This circuit is described in Section 5.2.1 on page 35. Further more the interface print contains connectors for the Pololu boards and the battery voltage monitor, these signal are just routed unchanged to the on-board computer.

5.5.1 Description of Multiplexer

To reduce the total number of analogue signals going to the on-board computer, two analogue multiplexers are used. These two multiplexers, reduces 31 analogue signals down to 2. It has been chosen to use a MAX396, see [Maxim Integrated Products, 2006], from Maxim Semiconductors, due to following properties:

- **Single-supply operation with wide voltage supply range ($2.7\,\mathrm{V} - 16\,\mathrm{V}$) :** The possibility for using single-supply voltage supply is preferable since no negative voltage supply is located on the on-board computer.
- **Low power consumption ($< 10\,\mu\mathrm{W}$):** Because of a power consumption less than $10\,\mu$W, it can be neglected compared to the power consumption of the other components located on the robot.
- **Maximum analogue input voltage same as the supply voltage:** This ensures that no other hardware is needed to convert the levels of the signals before the they are multiplexed.

- **Low cross talk between channels ($-92\,\mathrm{dB}$) :** The low cross talk between the channels in the multiplexer ensures that one specific measurement will not be affected by any of the other measurements.

- **High speed operation (Transition time less than $250\,\mathrm{ns}$) :** The multiplexer is fast enough to allow sampling all 16 input channels with a frequency of $250\,\mathrm{kHz}$. The specifications for the ARM9 processor, states that the internal A/D-converter is limited to 3750 samples per second [Cirrus Logic, 2004]. This means that the MAX396 is 67 times faster than the internal A/D-converter located inside the ARM9, which means that the multiplexer is fast enough for this purpose.

To retrieve the 32 signals, four address wires are used, to address the specific analogue signals. The same address signals are being used for both multiplexers, as the same sample rate is wanted for all sensors. Using the same address signals, simplifies the interface to the multiplexers. 32 signals plus the two signals from the voltage monitor circuit have to be sampled. The ADC in the on-board computer can, as mentioned, sample the signals with a frequency of $3750\,\mathrm{Hz}$, the delay between the first measurement and the last measurement can then be found to $t_{\text{delay}} = \frac{34-1}{3750Hz} = 8.8\,\mathrm{ms}$. This delay is found acceptable.

A picture of the interface print can be seen in Figure 5.7, where it is placed below the on-board computer.

5.6 Battery Pack and Voltage Monitor Circuit

In this project, one of the goals is to construct a robot capable of walking without any cables attached. To realize this, the robot needs a battery pack that is capable of providing the power that is needed by motors and on-board computer. The batteries should have a high capacity to provide a reasonable running time of the robot. It is therefore chosen to use two Lithium Polymer (LiPo) batteries, each with two cells and a capacity of $3700\,\mathrm{mAh}$. Each cell has an output voltage of $3.7\,\mathrm{V}$, they are coupled in series to achieve an output voltage of $7.4\,\mathrm{V}$. The batteries can provide $70\,\mathrm{A}$ of peak current, which is more than sufficient for this project. Setting the two batteries in parallel will result in one battery, with an output voltage of $7.4\,\mathrm{V}$ and a capacity of $7400\,\mathrm{mAh}$ capable of providing $140\,\mathrm{A}$ peak current. The batteries should, however, be charged individually. To provide this possibility, the batteries are not soldered directly together in parallel, but is set in parallel by a switch instead. This is described in Section C.1 on page 185. To reduce the total size of the two batteries, consisting of four LiPo cells, the batteries have been disassembled, and then combined into one single battery pack. The output from this battery, is two $7.4\,\mathrm{V}$ outputs and a common ground.

The voltage of each cell in the battery should never drop below $3.2\,\mathrm{V}$, therefore the battery voltage has to be monitored. Should the voltage drop below $3.2\,\mathrm{V}$, the capacity of the battery will be reduced, as well as the battery life time. It can even be difficult to recharge the batteries, as the internal safety circuits of some LiPo chargers do not allow such low voltage. To monitor the battery voltage, two identical voltage divider circuits have been designed to divide the battery voltage to a level that is acceptable for the A/D converter located on the on-board computer. Because the maximum input voltage to the on-board computer is $3.3\,\mathrm{V}$, it is chosen to divide the voltage by three.

Choosing $R_1 = 6.8\,\mathrm{k\Omega}$, R_2 is calculated to $R_2 = 3.3\,\mathrm{k\Omega}$ in a voltage divider circuit. This results in an output voltage on approximately one third of the battery voltage.

The battery voltage should be monitored by the main program, and should provide the user with some alarm-signal if the voltage is close to the limit. Automatic shut down is not possible due to the hardware design.

When the batteries has to be recharged, the robot should be set into charging mode by setting the switches in the correct combination. Table C.1 on page 187 shows the combinations of the two main switches. After the robot has been set into charging mode, the two batteries should be recharged by using a prober LiPo charger and a compatible balancing circuit. The balancing circuit ensures that the voltage difference between the two cells in a battery never exceeds $5\,\mathrm{mV}$.

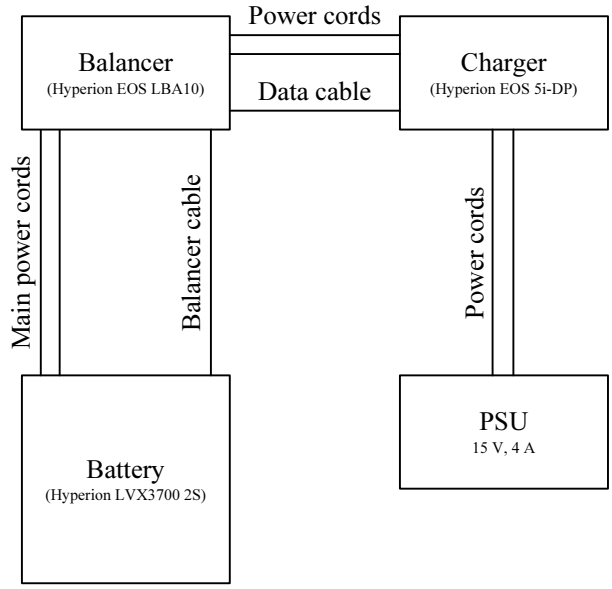


Figure 5.8: *LiPo battery recharging setup.*

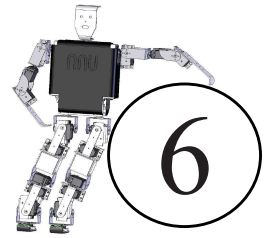
The setup shown in Figure 5.8 controls the recharging process, and stops when both cells in the battery has been recharged. It is important that both batteries are balanced and has the same voltage before they are put in parallel by the main switch.

5.7 Partial Conclusion to Hardware

The preceding sections have described the hardware that is required on the robot. In order to implement it, Orcad has been used to create a print layout that implements the filters, multiplexers, connectors etc. on a single PCB, called the interface print. It is designed such that it can be mounted on the back of the robot. The interface print is the link between the chosen sensors/actuators and the on-board computer. Figure 5.7 shows a picture of the interface print and the on-board computer, placed on the back of

the torso. It has been chosen to actuate the robot with 21 servo motors, and to equip it with accelerometer, gyroscope, potentiometers and strain gauges for sensor feedback. In Appendix C on page 185 is a description of the hardware wiring and the connectors implemented on the robot given. In this appendix a silk screen of the interface print can be seen, where the placement of the connectors is also seen.

SOFTWARE



Chapter contents

THIS CHAPTER *describes the software located on the robot and on the Simulink PC. The chapter starts out with an overall system description, and will then proceed with driver design and development. These drivers are interfaced from the Simulink PC to provide a simple interface to the actuators and sensors located on the robot.*

6.1 System Overview

This chapter describes the software design. The task of the software is to mimic the nervous system. It is responsible for sending signals to the muscles (actuators) and returning signals from the receptors (sensors). From Figure 3.1 on page 14 it can be seen that the software implements sensory feedback and motion control.

Figure 6.1 is used to show the basic parts of the entire system. The system consists of two units, the biped robot and a PC running MATLAB Simulink. The robot consists of an on-board computer, that is described in Section 5.4 on page 40, and an interface print, that is described in Section 5.7 on page 44. The interface print is used to gather information from all sensors, and to filter the potentiometer signals. This interface print needs to be interfaced from the on-board computer by a set of drivers.

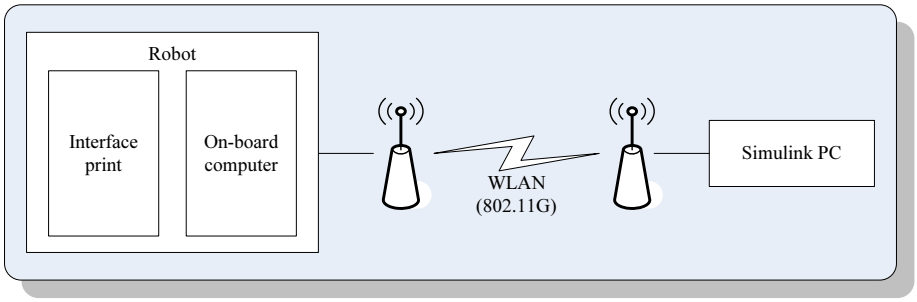


Figure 6.1: *System description. A wireless connection is established between the robot and a computer running Simulink.*

The on-board computer comes preinstalled with Technologic Systems TS-Linux in the flash memory. TS-Linux is a compact Linux distribution which boots in a few seconds and can be accessed through a console or telnet. A consequence of TS-Linux's compact size is the absence of a compiler which means that code has to be compiled using a cross compiler. A cross compiler is a compiler that runs on a normal PC which is able to compile code that can be executed on another platform.

TS-Linux differs from other Linux distributions because it can function as a bootloader, meaning it can boot other kernels or operating systems. This means that more general Linux distributions can be booted where a compiler is included. It has been chosen to install a Debian ARM distribution on the SD memory card since code can be compiled directly on the on-board computer and the wireless LAN (WLAN) adapter is supported. Furthermore, by avoiding altering the TS-Linux in the flash memory, the on-board computer can always be booted should the Debian installation become corrupted. The on-board computer is equipped with a Maverick FPU which calculates floating point numbers in hardware as opposed to doing this in software which is slower. The Debian installation provided by Technologic Systems is based on kernel 2.4.26 with GCC 3.3.5 which does not support the Maverick FPU which the CPU on the on-board computer uses. Support has however been implemented starting from GCC 4.1 on 2.6 kernels with Vector Floating Point patches. Complete root file systems can be obtained from [Applieddata.net, 2007], [CodeSourcery, 2007], and [Debian, 2007] or compiled from scratch. However, Technologic Systems only supplies a

object file for the UART which can not be linked with object files compiled with GCC 4.1. Since the UART is essential for the robot to function it is necessary to forgo the possibility of using the FPU by compiling the code with GCC 4.1. Another way of using the FPU is by assembler instructions, however converting all operations involving floating point calculations to assembler would be very time consuming and a rather cumbersome task. It is therefore chosen to use the Debian distribution provided by Technologic Systems.

To ease the implementation of controllers and filters Real-Time Workshop (RTW)for MATLAB Simulink is used. RTW discretizes a Simulink model and generates C-code that encapsulates the model. In order for RTW to support linux the soft real-time linux target LNX [Bhanderi, 2006] is used which executes the model at the frequency specified in the Simulink model.

Figure 6.1 shows the robot and the Simulink computer connected through the wireless connection. The idea is that a set of drivers are developed as S-functions in a Simulink environment. These drivers provide a simple interface to the actuators on the robot as well as all the sensor signals. The code, along with a automatically generated make-file, is then uploaded to the robot through the wireless connection. In order to compile the uploaded code an ssh connection is established to the robot and the make-file is executed using *make*. After the code has been compiled, a stand-alone program is available on the robot, that is executed using the ssh connection and the task specified in the Simulink model is carried out. After the code has been generated, only the make-file and the generated code is needed on the on-board computer. The work flow for executing a command specified by Simulink is seen from Figure 6.2. This figure also shows that it is possible to download the data measured on the sensors.

To exploit the full potential of the 200 MHz ARM9 processor located on the on-board computer, a high level interface has been developed which implements a multi-threading system. So the software layers consists of a Debian Linux distribution, the LNX RTW Target controlling the soft real-time requirements, the developed Simulink model, the high level interface drivers ensuring multi-threading and the low level input/output drivers where the communication protocol to interface sensors and servo motors is implemented. The layering of the implemented software is depicted in Figure 6.3.

Next a description of the low level I/O drivers is given, after which a description of the high level interface drivers is given. All the code that the drivers consist of will not be shown here. Instead the reader is encouraged to read the source code located on the enclosed CD-ROM under `software/drivers`.

6.2 I/O Drivers

The drivers can be divided into input drivers and output drivers. The drivers will in the following be described as such. Technologic System supplies code that shows how the interfacing of the I/O is done. This code has been utilized in the development of the drivers.

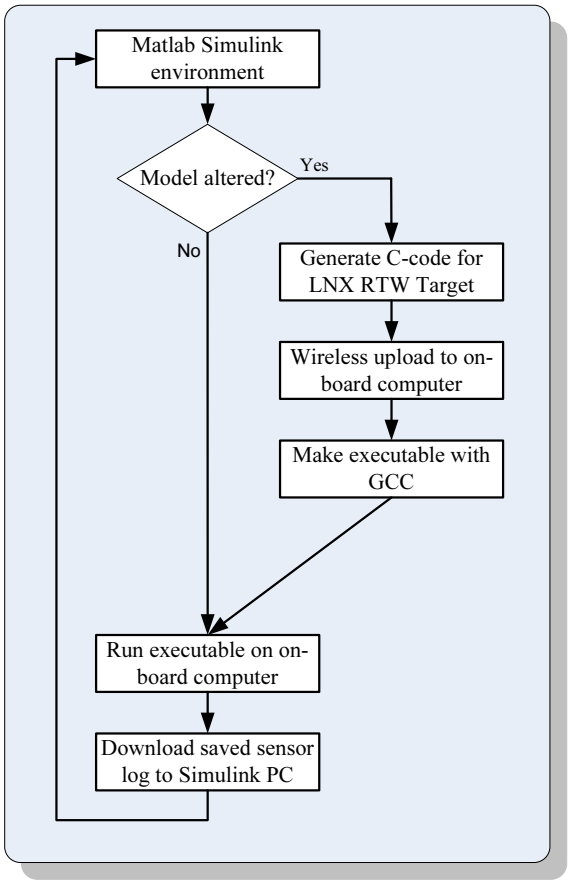


Figure 6.2: *The routines that run when using the system.*

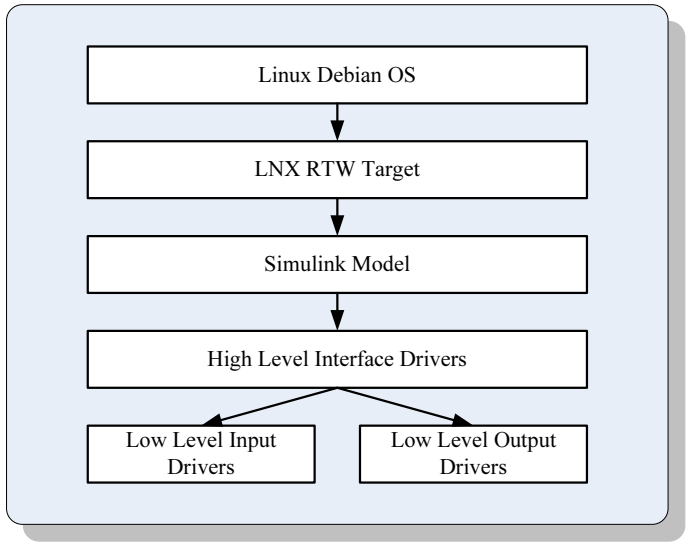


Figure 6.3: *The different parts of the software system layer.*

6.2.1 Low Level Input Drivers

As described in Section 5.5.1 on page 42, 32 sensor input signals have to be read through two external multiplexers which are each connected to another multiplexer located internally in the on-board computer. Furthermore the voltage levels on the batteries have to be read as well. Figure 6.4 shows how the multiplexers and the batteries are connected to the on-board computer. As can be seen on the figure the four address pins (AD0-3) on the multiplexers share the same four general purpose input output (GPIO) pins on the on-board computer. This means that changing the levels of the GPIO pins sets a new address on both multiplexers. ADC0-3 on Figure 6.4 refers to the analogue-to-digital converters on the on-board computer.

The address pins are not connected to the GPIO's in an incrementing order because this would have complicated the layout of the interface print. This has to be handled in the driver so the correct sensors are addressed. Changing the levels of the GPIO's are controlled by a single 8 bit register where a high bit corresponds to a high level. Table 6.1 shows which input on the multiplexer is connected to the ADC at the given register values.

The task of reading a sensor is thereby twofold, the correct value has to be written to the register and afterwards the ADC0 and ADC1 has to be read. An intuitive approach would be to read both ADC0 and ADC1 before changing the address on the external multiplexer. However, because the on-board computer does not have four dedicated ADC channels and instead uses a multiplexer that connects four inputs to a single ADC, it is faster to read all 16 channels on one of the external multiplexers and then change to the next ADC. This is because the multiplexer on the on-board computer is slower than the one used on the interface print. The sequence for reading all the sensors is seen from Figure 6.5.

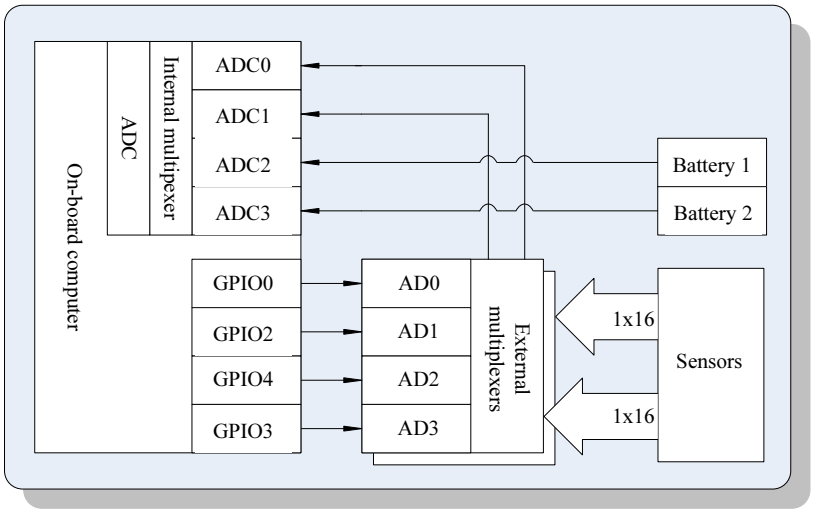


Figure 6.4: *The figure shows how the multiplexers and the batteries are connected to the on-board computer.*

Input	Register value	ADC0	ADC1
1	0	θ_7	IMU gyro X
2	1	θ_9	IMU gyro Y
3	4	θ_8	IMU reference voltage
4	5	θ_{10}	IMU acc. X
5	16	θ_{11}	IMU acc. Y
6	17	θ_{12}	IMU acc. Z
7	20	θ_6	Right foot force 1
8	21	θ_4	Ground
9	8	DIP switch 3	Right foot force 2
10	9	DIP switch 4	Right foot force 3
11	12	DIP switch 1	Left foot force 1
12	13	DIP switch 2	Left foot force 2
13	24	θ_5	Left foot force 3
14	25	θ_3	Push button 1
15	28	θ_2	Push button 2
16	29	θ_1	Push button 3

Table 6.1: *The register value corresponds to the input that is connected to the ADC.*

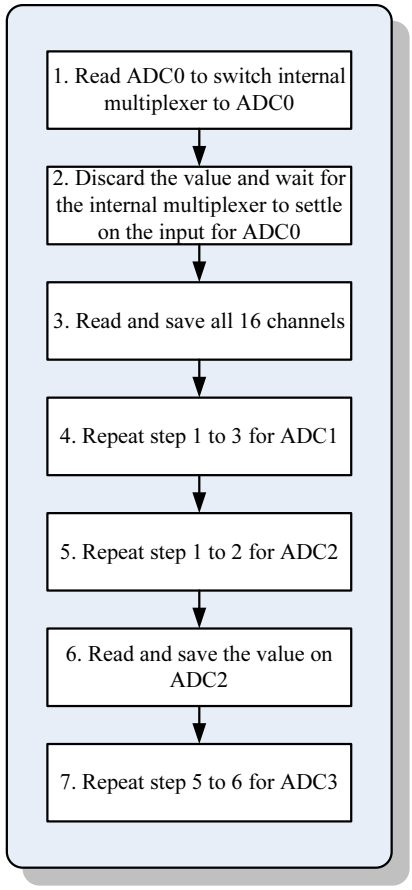


Figure 6.5: *The sequence called for reading all the sensors.*

The ADC is read using the function `read_channel` which was supplied by Technologic Systems.

`read_channel:`
Arguments: *unsigned long adc_page, unsigned short channel*
Returns: `int`
Explanation: The function returns the value recorded by the ADC channel specified by the argument *channel*. The argument *adc_page* is the memory page of the ADC.

The virtual address space is divided into two areas. One area is the kernel space which is restricted to access from the kernel only. The second area is the user space where in the compiled Simulink model runs. As the ADC stores the read values in kernel space, these can not be accessed from the compiled code in user space by simply assigning the address of the register to a pointer. Doing so would lead to a segmentation fault. Instead the system call `mmap` is used to access kernel space from user space.

`mmap:`
Arguments: *void *start, size_t length, int prot, int flags, int fd, off_t offset*
Returns: *void **
Explanation: Is a function that can map devices in to memory. `mmap` asks to map *length* in to memory from the file descriptor *fd* starting at *offset* preferably mapped to *start*. *prot* tells what kind of protection should be on the mapping, eg. read only or write only. *flags* specifies if the map should be private to the process or shared.

The value returned from the ADC is not a linear representation of the voltage which is explained in the data sheet [Cirrus Logic, 2004, pp. 34]. If the voltage is between $0\,\mathrm{V}$ and $1.65\,\mathrm{V}$ the ADC will return a value from $40,536$ to $65,535$ where $40,536$ represents $0\,\mathrm{V}$. If the voltage is between $1.65\,\mathrm{V}$ and $3.3\,\mathrm{V}$ the ADC will return a value from 0 to $25,000$. The returned value from the ADC is normalized by the function `adcNorm` to a value between 0 and $50,000$.

`adcNorm:`
Arguments: *int adcVal*
Returns: *int*
Explanation: The function converts *adcVal*, which is the value returned by `read_channel`, to a value between 0 and $50,000$.

Conve¡rting the normalized ADC value to a voltage can be done using Equation (6.1)

$$V_{\mathrm{ADC}} = \frac{3.3 n_{\mathrm{ADC}}}{50,000} \tag{6.1}$$

where n_{ADC} is the normalized value from the function `adcNorm`. The complete driver can be found in `Software/Drivers/ADC/sensor_thread.c` on the enclosed CD-ROM.

Known driver issues

As described in the previous section `mmap` is used to access kernel space where the ADC registers are situated. The LNX RTW Target uses the function `mlockall` to avoid that the address space, used by the process, is paged to the swap area, i.e., the memory card which is slower to read from than RAM.

`mlockall:`
Arguments: *const void *addr, size_t len*
Returns: *int*
Explanation: The function locks the virtual address space starting from *addr* to *len* bytes for the calling process.

This is done to increase the performance, however, in kernel 2.4 which is supported by Technologic System the same process can not use both `mmap` and `mlockall`. Doing

so makes the kernel crash and the system needs to be rebooted. `mlockall` is therefore not used. As a consequence the model stops with an alarm clock until it has been called 4-5 times. The reason why the program then starts after some attempts is most likely because the kernel recognizes the model as a frequently called program and keeps the memory page in RAM which results in faster startup of the model.

6.2.2 Low Level Output Drivers

The task of the output driver is to control the servo motors. To do this the driver must support the communication protocol specified in Section A. However not the entire protocol will be implemented as it is only necessary to control the position and the velocity. The servo controllers uses a standard UART for the communication. Because of the Debian Linux running on the on-board computer the UARTs have to be utilized through the kernel which is done using `open`. `open` returns a file descriptor that can be used with `write`, which writes a string on the UART. However before `write` is used the UART has to be configured with the correct settings which is done using a struct named *adc*. The entries in *termios* dictates the baud rate, synchronous or non-synchronous operation etc. These settings are given in Appendix A on page 173 concerning the setup of the protocol used for communication with the Pololu boards. `open` and *termios* are used in `initSerPort`.

`initSerPort`:
Arguments: *char *device*
Returns: *int*
Explanation: Returns a file descriptor to the UART pointed to by **device*. **device* is a string which gives the path to the serial port, eg. `/dev/ttyAM0`.

After the serial port has been initialized by `initSerPort` the Pololu boards can be accessed. Controlling the position of the servo motors can be done using command 4 of the Pololu protocol which is explained in Appendix A on page 173. This is implemented in the function `sendPos`.

`sendPos`:
Arguments: *unsigned short int pos, unsigned char servoNumber, int fd*
Returns: *int*
Explanation: Implements command 4 of the Pololu protocol. *pos* is a value between 500 and 5500 specifying the desired position of the servo motor connected to output *servoNumber* on the serial port pointed to by the file descriptor *fd*. In order to update the position of a servo motor it is necessary to send a packet of 6 bytes.

Controlling the velocity of the servo motors can be done using command 1 of the Pololu protocol which is implemented in the function `sendVel`.

`sendVel`:
Arguments:*unsigned var vel, unsigned char servoNumber, int fd)*
Returns: *int*
Explanation: Implements command 1 of the Pololu protocol. *vel* is the velocity at which the servo motor should move. This can be between 0 and 127. 0 makes the servo motor change to the desired position as fast as possible. *servoNumber* is the output port on the Pololu board pointed to by the file descriptor *fd*. In order to set the velocity of a servo motor it is necessary to send a packet of 5 bytes.

From the functions `sendPos` and `sendVel` it can be seen that a total of $\Delta d = 11\,\mathrm{B}$ has to be sent to each servo motor if position and velocity are to be controlled. Each servo controller board has a dedicated UART as described in Section 5.4 on page 40 where it is possible to connect 8 servo motors. The servo controllers can not operate at a baud rate higher than $38,400$ baud. To translate this to bytes per second it has to be determined how many bits that are needed to transmit a byte. The servo controller

board specifies that $8\,\mathrm{bit}$ should be sent at a time, including no parity and one stop bit. If one byte has to be transmitted a start bit and stop bit have to be sent along with the $8\,\mathrm{bit}$ giving a total of $10\,\mathrm{bit}$. However using 10 bits assumes that there is no delay between sending two bytes. The data sheet [Cirrus Logic, 2004] for the on-board computer does not mention how long this delay is. Therefore it has been chosen to assume that a byte, that is to be transmitted requires 11 bits. Converting the baud rate to a data rate can thereby be done in the following way:

$$s_{\max} = \frac{38400\,\mathrm{baud}}{11\,\mathrm{bit}} = 3490\frac{\mathrm{B}}{\mathrm{s}} \tag{6.2}$$

As the maximum data rate has been determined it is now possible to determine the maximum update frequency. This is done in a worst case sense meaning that it is assumed that 8 servo motors are connected to the servo controller board:

$$f_{\max} = \frac{s_{\max}}{n_{\text{servos}}\Delta d} \tag{6.3}$$

$$= \frac{3490\frac{\mathrm{B}}{\mathrm{s}}}{8 \cdot 11\,\mathrm{B}} \tag{6.4}$$

$$= 39.7\,\mathrm{Hz} \tag{6.5}$$

However a better approach would be to connect 7 servo motors to each servo controller board reducing n_{servos} to 7, and thereby increasing $f_{\max}$ to $45.3\,\mathrm{Hz}$. It can also be mentioned that if the velocity input is omitted then the maximum update frequency, $f_{\max}$, can be increased to $83.0\,\mathrm{Hz}$.

The source code for the low level output functions can be found on the enclosed CD-ROM in `Software/Drivers/Serial driver/serial_driver.c`.

In the next section the high level interface of the drivers is described.

6.3 High Level Interfacing

The following section describes the interface between the Simulink model and the low level drivers. This interface isolates the low level drivers from Simulink such that only two functions are needed by Simulink for each task; an initialization function and a function for the task. The tasks here are to update the servo motors and collect sensor data.

6.3.1 Interfacing the Servo Motors

The flow of the functions used to update the servo motors is depicted in Figure 6.6.

Updating the servo motors is a rather slow process due to the low baud rate that can be communicated with the Pololu boards. In Equation (6.5) the maximum update frequency, $f_{\max}$, was determined to $39.7\,\mathrm{Hz}$. This frequency is without any execution of code on the robot, which means that the frequency has to be lowered to give time to executing code. A better approach would be to update the servo motors in parallel with

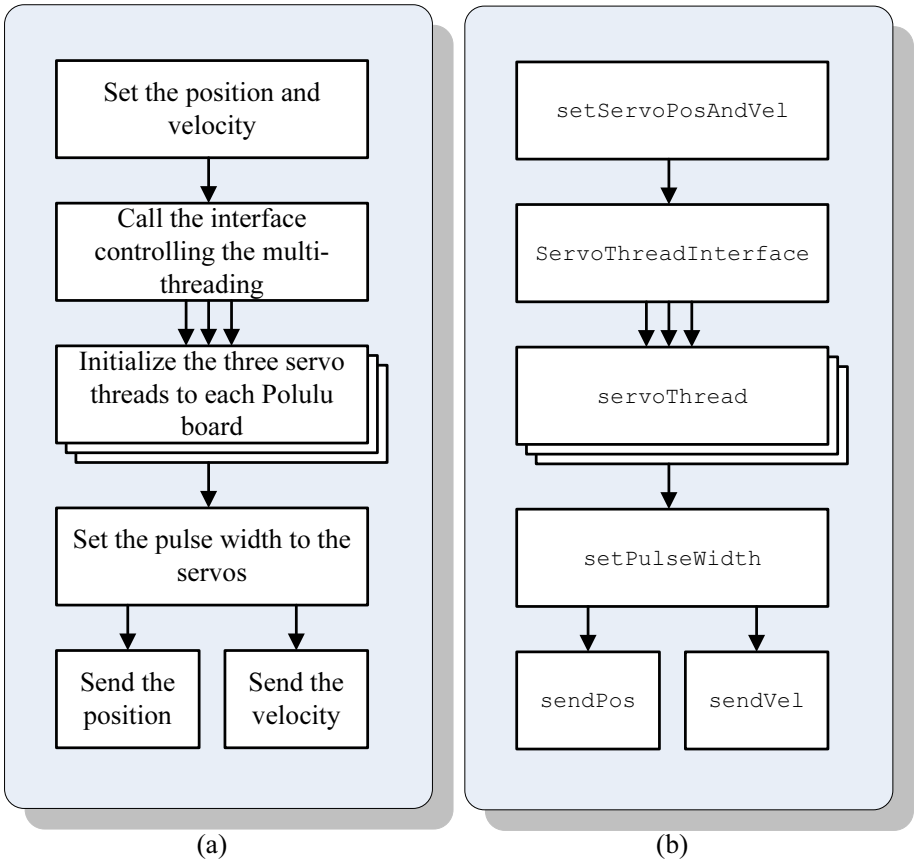


Figure 6.6: *The flow of the functions used to update the servos. (a) The boxes show the layering of the software and what is done by each layer. (b) The name of the functions that implement the layers.*

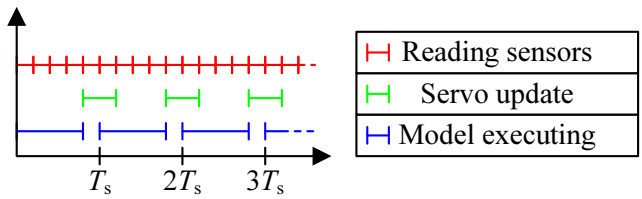


Figure 6.7: *The update of the servo motors is allowed to run in the succeding time slot.*

the executing code.

In Figure 6.7 the principle of parallel execution is depicted, where T_s is the sample time. Here the update sequence of the servo motors can continue in the next timeslot as long as $f_{max} > 1/T_s$. To accomplish this each Pololu board is controlled by a thread. A shared memory area, called a critical section, is created between the Simulink model and the threads. This critical section is protected by a mutual exclusion (mutex) to avoid concurrent access. A library exists in Linux which implements the POSIX thread standard called pthreads. Mutex support is directly implemented in pthreads. But before the thread can be designed it is necesarry to have a way of representing the Pololu boards and the servo motors. In order to describe each servo motor the following struct is used:

```
struct _servoMotor{
  short int servoNumber;
  short int outputNumber;
  short int servoType;
  double upperLimit;
  double lowerLimit
  double pos;
  double vel;
}
```

In the struct *_servoMotor* the variable *servoNumber* is the number of the servo motor as described by Figure C.3 on page 188. *outputNumber* is the connector on the Pololu board which the servo motor is connected to. *servoType* is the type of servo motor, either a HS645 or a 5995TG. *upperLimit* and *lowerLimit* are the limits, in degress, in which the servo motor is allowed to operate. *pos* is the postion of the servo motor shaft in degrees and *vel* is the velocity at which the servo motor should move to a new position. Each Pololu board can have up to 8 servo motors connected. This is encapsulated in the following struct:

```
struct _servoBoard{
  short int boardNumber;
  short int numberOfServos;
  short int data;
  pthread_mutex_t *newData;
  char serialPort[25];
  struct _servoMotor servoMotors[8];
}
```

In the struct *_servoBoard* the variable *boardNumber* is the number given to the Pololu board as described in Table C.3 on page 187. *numberOfServos* is the number of servo motors that are connected to the Pololu board. The mutex *newData* is used to avoid concurrent access to the *_servoBoard* struct. *serialPort[25]* is a string which gives the path to the serial port, e.g., `/dev/ttyAM0`, which the Pololu board is connected to. In order to update the positions contained in the *servoBoard* struct the function *setPulseWidth* is used.

`setPulseWidth:`
Arguments: `struct _servoBoard *servoBoard, int fd`
Returns: `void`
Explanation: The function traverses the array *servoMotors* in **servoBoard* and updates the position and velocity of each `_servoMotor` struct. `fd` is a file descriptor of the serial port which the Pololu board is connected to. The upper and lower limits of the servo motors are checked and the output is limited if necessary. If the limit is exceeded a warning is printed in the console telling which servo motor was limited. `setPulseWidth` uses `sendPos` and `sendVel`, described in Section 6.2.2 on page 55, to update position and velocity of the servo motors.

The task of the thread, called `servoThread`, is now to take a pointer to a *_servoBoard* struct and call `setPulseWidth` with this struct.

`servoThread`:
Arguments: *void *servoBoard*
Returns: *void **
Explanation: When the thread is called it first calls `initSerPort` with the path to the serial port given in **servoBoard-¿serialPort*. The thread waits for **servoBoard-¿data* to become true indicating that there are new positions and velocities that should be updated. To avoid that the thread uses its complete time slot, checking to see if *data* becomes true, `sleep(0)` is called each time **servoBoard-¿data* is checked and it is false. `sleep(0)` releases the CPU so other processes or threads may run instead.

Since there are three Pololu boards three `servoThread` threads have to be started each with their own *_servoBoard* struct. The three *_servoBoard* structs have to be stored some where in the memory which also allow the Simulink model to access them. This is done using the function `servoThreadInterface`.

`servoThreadInterface`:
Arguments: *int task, double *positions, char *velocities*
Returns: *void*
Explanation: The argument *task* can either be SEND or INIT. If *task* is INIT the function initializes three static *_servoBoard* structs with the appropriate limits, serial port, servo number etc. If *task* is SEND then the values pointed to by *posistions* and *velocities* are inserted in the *_servoMotor* structs and the threads sends the new values to the servo motors.

To enable the Simulink model to update the positions and velocities of the servo motors the function `setServoPosAndVel` is used.

`setServoPosAndVel`:
Arguments: *double *servoPos, signed char *vel*
Returns: *int*
Explanation: The function updates the positions and the velocities of the servo motors given in **servoPos* and **vel*.

The positions **servoPos* and velocities **vel*, in the function `setServoPosAndVel`, must be ordered incrementally. That is, the position and velocity of servo motor 1, abbrivated as S1 in Figure C.3 on page 188, must be in *posistions[0]* and *velocities[0]*, position and velocity of servo motor 2 must be in *posistions[1]* and *velocities[1]* etc. `servoThreadInterface` is called with *task* set to SEND, the pointer **positions* set to **servoPos*, and the pointer **velocities* set to **vel*. Before `setServoPosAndVel` can be called by the Simulink model the threads have to initialized. This is done by calling the function `initServos`.

`initServos`:
Arguments: *void*
Returns: *void*
Explanation: The function initializes the threads that update the servo motors. `initServos` must be called before `setServoPosAndVel` can be used.

The source code for the low level output functions can be found on the enclosed CD-ROM in `software/drivers/serial_driver.c`.

6.3.2 Interfacing the Sensors

Using the same approach as described in Section 6.3.1, and depicted in Figure 6.7, the sensors are read by a thread. The readings are stored in the struct *_acq*.

```
struct _acq{
  pthread_mutex_t *readingsMutex;
  double *readings;
}
```

The thread, `readSensorsThread`, implements the approach described in Section 6.2.1 for reading the sensors.

`readSensorsThread:`
Arguments: *void *p*
Returns: *void*
Explanation: The thread continously reads the sensors and stores them in **_acq-¿readings*. The mutex *readingsMutex* is locked with every update of **_acq-¿readings*.

The thread continously reads the sensors as fast as possible. Everytime a sensor is read the mutex *readingsMutex* is locked and when the sensor value has been aqcuired the mutex is unlocked and the thread pauses until the multiplexer has settled on a new sensor. In this pause the sensor readings can be extracted and used in Simulink. This requires some shared memory that can accessed both by the thread and Simulink. The interface between the thread and Simulink is implemented by the function `readSensorsInterface`.

`readSensorsInterface:`
Arguments: *struct _acq *acq, double *readings, int task*
Returns: *void*
Explanation: The function either copies the contents of **readings* to **acq-¿readings* or copies **acq-¿readings* to **readings* depending on the value of *task*.

When the thread is started it calls `readSensorsInterface` with task set to INIT and a pointer to a `_acq` struct where the sensor readings are stored. The thread can then put readings in the shared memory by writing to the pointer `readings`. When Simulink needs the readings it calls `readSensors` from an S-function with a pointer to a local double array that can hold 34 values.

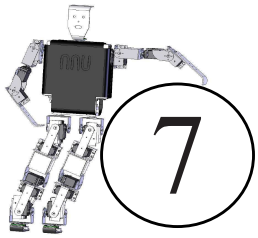
`readSensors:`
Arguments: *double *readings*
Returns: *void*
Explanation: The function copies the sensor readings to **readings*.

`readSensors` calls `readSensorsInterface` with *task* set to GETDATA and the pointer *readings*, which lock the mutex and copies the data to the pointer. Before `readSensors` can be used the function `initSensors` has to called which initializes the thread `readSensorsThread`.

6.4 Partial Conclusion to Software

This chapter gave a description of the implemented software. The solution given enabled the robot to be completely autonomous. A Debian distribution was installed on the SD card on the on-board computer. A real-time target for RTW was installed enabling the use of Simulink for the design of the controllers. The developed drivers were designed to run in a multi-treaded environment in order to optimize the acquisition of sensor readings. Futhermore Simulink was isolated from the low level drivers by only having to initialize the threads by calling `initServos` and `initSensors` and afterwards using `setServoPosAndVel` to update the servo motors and `readSensors` to read the sensors.

MODELING



Chapter contents

IN THIS CHAPTER *a model for biped robot system i developed. The purpose of model is to give an insight into the system and thereby obtain a model which can be used for testing and simulation of controllers. In the following an introduction is given to the general system of biped robots, afterward a look at the approaches in the existing literature will be given. Then, as a last point before describing the development of the actual model, an explanation of the approach taken in this project is given.*

7.1 Introduction to Modeling

When dealing with modeling of a biped robot, there are two basic types of models; a kinematic and a dynamical model. The kinematic model describes the motion of the biped robot in terms of positions, velocities, and accelerations based on the joint angles. Exterior forces that cause the motion are not considered, nor is friction or system dynamics. The dynamical model is used to calculate the torques that act on each joint. Hence, all exterior forces are included in the dynamical model.

In this project, where the processing power on the biped robot is limited to 200 MHz, it is important that one considers the purpose of the model; should the model be implemented on the biped for model-based control or should the model be used for simulating the behavior of the robot before implementing control algorithms on the biped robot? If the accuracy of the model is of great importance, the loss of energy and dynamics in the gears should be taken into considerations, as well as the friction in each joints. Dynamic coupling between the joints could also be taken into considerations to achieve the highest accuracy. Therefore a review of different literature and methods is given later, and following a brief summery of the selected methods are given.

7.1.1 System Description

The biped robot under study is a 21 DoF biped robot. It consists of a torso, two identical legs with six joints each, two identical arms with four joints each, and a head with one joint. All 21 joints are revolute and actuated by servo motors. The friction between the feet and the ground is considered sufficient to prevent slippage.

The system to be modeled contains discrete states, and can therefore be considered a hybrid system. The two main systems are represented: first SSP, where the system can be considered as an open kinematic chain, and second DSP, where the system can be considered as a closed kinematic chain. In Figure 7.1 and 7.2 the two different systems are illustrated. The problem in modeling such a system arises in the transition between SSP and DSP. Also the motion in DSP can be difficult to describe since it is subjected to some mechanicals constraint when having both feet are on the ground.

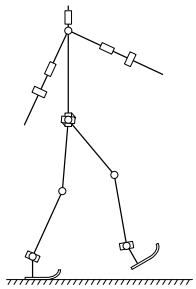


Figure 7.1: *A view of the biped robot in the sagittal plan. In SSP the system can be considered as an open kinematic chain.*

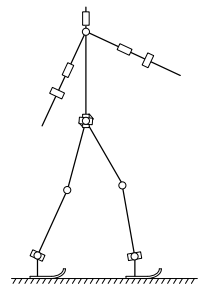


Figure 7.2: *A view of the biped robot in the sagittal plan. In DSP the system can be considered as a closed kinematic chain.*

7.1.2 Previous Work

Different attempts have been made to set up models for biped locomotion. In some of the many available papers the approaches are varying. A large number of prevailing papers only cover the SSP [Shih, 1996], and some only the DSP [Zhao et al., 1997]. Some neglects the impact phase [Mu and Wu, 2004] while others claim that this phase is the most important phase in the description of biped locomotion [Bruneau et al., 2001]. Some include a very thorough actuator model, while other completely disregard the actuators.

Many of the papers also have some similarities, they rarely includes the arms in the model, many consider robots with low degree of freedom, usually 4-6 DoF's and in most papers the model only describes the biped robot in the sagittal plane. Furthermore rotation about the z-axis in the hip is almost never included. All these points simplify the modeling much.

In this project, it is desired to develop a complete model, describing a biped robot with 21 DoF's and human proportions. Furthermore it is desired to describe movement in both the sagittal and frontal plane, while switching between SSP and DSP. To develop this model, it is necessary to combine ideas presented in many different papers. The following will be a description of some of the approaches taken in the available literature. The description is divided into some of the main topics within modeling of biped robots.

Kinematic Representation To be able to describe the position and orientation of the links on the biped robot, a frame is attached to each of them. The kinematic model developed in [Mu and Wu, 2004] considers the biped robot as an open-loop serial-chain from the supporting foot to the free ends. This frame attachment is intuitive, since all positions are found relative to a frame fixed to the ground. However, some problems can arise during the DSP, where both feet are placed on the ground. Furthermore some consideration should be given toward the fact that the biped robot will shift the supporting foot during a walking cycle. In [Yamaguchi et al., 1999] the position of all links are given relative to a frame attached in the hip. Using this configuration there is no need for moving the base frame around. A consequence of having the frame fixed to the hip, is however that all positions are found relative to a frame that moves with respect to the ground. In [Li et al., 1993] the model is developed with respect to a ground fixed frame and a moving frame placed in the waist of the robot.

Dynamic Representation The dynamics of the biped robot can be described using two different approaches; the Newtonian and the Lagrangian. The Newtonian approach is based on Newtons laws while the Lagrangian approach builds on energy considerations.

In [Craig, 1989] a very thorough description of the Newtonian approach is given. Though the application is manipulators, the ideas can be applied on a biped robot. In [Christensen et al., 2006] a Newtonian model, based on [Craig, 1989], describing a 21 DoF biped robot in SSP has been implemented. Furthermore [Christensen et al., 2006] has implemented a model of a 10 DoF biped robot in SSP based on the lagrangian approach, and compared the two approaches. The

main result of this comparison was that the Newtonian approach was more intuitive, but had shortcomings when the DSP should be modeled. However [Christensen et al., 2006] gives a proposal for a DSP model using the Lagrangian approach. The Newtonian model is recursively defined and therefore the calculation time increases significantly with the number of DoF. One way of optimizing the Newton Euler algorithm is by reducing the DoF. To do that, the reduced DoF must be taken into account by adding constraints to the kinematic, dynamical and inverse kinematic model. In the swing phase, it is possible to lock some of the joints, and thereby merging two or more joints to a single joint. Similar, in DSP it is also possible to merge some of the joints to achieve a reduced model, which has been used in [Hardt et al., 1999]. It is clear that the effect of the optimization is limited due to the fact that even though the number of DoF is reduced, the algorithm is still time consuming and with a high number of DoF, the possibilities for implementing such model on a small-scale computer is impossible [Chaillet et al., 1994].

The Lagrangian approach is very time demanding in the derivation phase, but when the model is derived first time, the simulation time is low compared to the Newtonian approach, for large DoF. This is due to the fact that the Lagrangian approach will result in a number of analytic equations compared with the recursive algorithms of the Newton Euler approach.

Constraints in DSP Many papers treat the difficult issues that arise when the biped robot is in DSP and the system is imposed to different constraints. In [Mu and Wu, 2003] and [Mu and Wu, 2004] a complete dynamical model for a five link biped robot is outlined, the model is based on D'Alembert's principle (Lagrangian). The constraints imposed on the system are implemented as Lagrangian multipliers, which is also described in [Wisniewski, 2006]. In [Udwadia and Kalaba, 2001] and [Udwadia and Kalaba, 2002] they obtain the equations of motion for systems where the principle of D'Alembert's does not hold, which is the case when sliding friction is present. In [Christensen et al., 2006] an approach is suggested where the DSP model is based on a combination of SSP models. The latter seems optimal since the SSP models can be used for both SSP and DSP. No specific model for the DSP model needs to be derived.

Impact phase The switch phase in a normal gait-cycle has a very strong influence on the system dynamics. In [Buschmann et al., 2006] the importance of deriving a model for the foot if this contains damping or any other shock absorber is mentioned. In the case of the biped robot for this project, the foot could be modeled as a spring and a damper in parallel. This model can be used as a transition between the SSP and the DSP model. The same method is used in [Buschmann et al., 2006] where the feet on Johnnie are modeled as a spring and a damper in parallel. Other articles mention that the need for deriving a model of the feet is limited due to the fact that the duration of the impact is very short. So by saying that the time is infinite short, there is no need to model the impact phase. These, however, do not use any shock-absorber and often model their robots as having single-point contact with the surface [Doi et al.,]. A model describing the system in the impact phase rather than a model of the foot, is given in [Hurmuzlu, 2001]. Here it is found that if the impact phase is included

in the model, it will result in smooth walk, as the biped robot is able to minimize to impact.

7.1.3 Modeling Approach

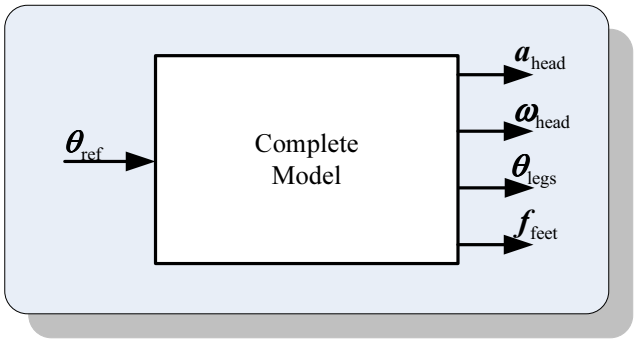


Figure 7.3: *The model of the biped robot, with the same input and output as the physical system.*

The final model of the biped robot should describe the relationship between the input applied to the physical system and the output measured at the physical system. This is illustrated in Figure 7.3.

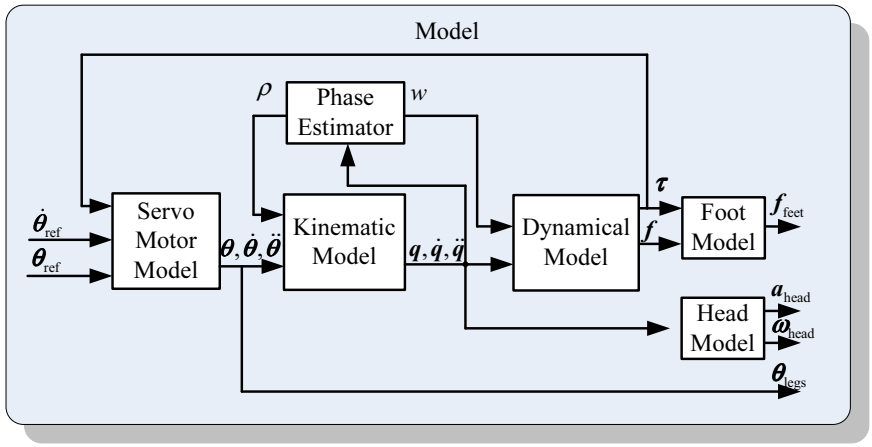


Figure 7.4: *An extended illustration of the model, describing the interfaces and division between/of sub models.*

In order to clarify the modeling of the entire biped robot, it is divided into several sub models. These sub models can be seen in Figure 7.4, where the interface between the models is also illustrated. The model is divided into five sub models, which are:

Servo motor model As seen in Figure 7.4 the input to the model are the angular position references, θ_{ref}. In the physical system these references are given to each servo motor, whose internal controller ensures that the motor shaft obtains this

position. Therefore the first sub model is a model of the servo motors. This model should describe the relationship between the input reference angle and the actual position, $\boldsymbol{\theta}$, velocity, $\dot{\boldsymbol{\theta}}$, and acceleration, $\ddot{\boldsymbol{\theta}}$, of the motor shaft. This motion is however also dependent on the load of the servo motor, therefore the servo model is extended with an additional input, providing the torque on the motor shaft. This torque is calculated in the dynamical model, and provided as feedback to the servo motor. From the output of the servo motor model, it is possible to retrieve the actual position of the twelve servos in the legs, $\boldsymbol{\theta}_{\text{legs}}$. The deviation of the servo motor model is explained in Section 7.2.

Kinematic model At this point the exact motion of the biped robot is given in joint space using $\boldsymbol{\theta}$, $\dot{\boldsymbol{\theta}}$, and $\ddot{\boldsymbol{\theta}}$. A kinematic model transforms from joint coordinates into generalized coordinates, $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$. This makes the motion of the biped robot be given in both joint coordinates and Cartesian coordinates. The deviation of the kinematic model is explained in Section 7.3.

Dynamical model With the generalized coordinates as input, a dynamical model is found. This dynamical model describes the torque that each servo motor is subjected to, given the motion. This torque is fed back to the servo motors. The dynamical model describes both the SSP-R, SSP-L and the DSP. As described in Section 7.1.1 the SSP and DSP are two different systems, therefore the dynamical model will be a hybrid model, capable of describing both types of system. The dynamical model will be developed using Lagrangian formulation. The deviation of the dynamical model is explained in Section 7.4.

Phase estimator Both the dynamical and the kinematic model are dependent of the phase of the hybrid system, that is if the biped robot is in SSP-R, SSP-L or DSP. A phase estimator is designed in order to determine the phase. Further the phase estimator calculates the weight distribution between the feet, since the dynamics of the system is dependent on this distribution.

Foot model In order to determine the six forces exerted on the feet it is necessary to have a model of the foot. This model takes the forces and torques exerted in the ankle joint, and returns the forces exerted on the left and right foot. The deviation of the foot model is explained in Section 7.6.

Head model The output from the kinematic model can be transformed using the head model, to give the output of the IMU, namely the linear acceleration and angular velocity. The deviation of the head model is explained in Section 7.7.

In the following the development of each of the five models will be described, starting with the servo motor model.

7.2 Servo Motor Model

The servo motor can be controlled by a position reference,$\boldsymbol{\theta}_{\text{ref}}$, and according to this reference, and the load, $\boldsymbol{\tau}$, the shaft is moved with a certain velocity and acceleration. The servo motor model is illustrated in Figure 7.5. The block contains 21 servo motor models, each having τ_i and θ_{ref_i} as input, and θ_i, $\dot{\theta}_i$ and $\ddot{\theta}_i$ as output. Note that there are two different servo motors 12 of the type HSR-5995TG and 9 of the type HS-645MG. The specifications of the two servo motors can be found in Table 5.1 on page 38.

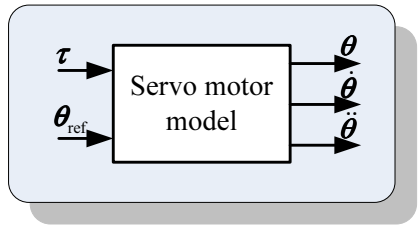


Figure 7.5: *The I/O of the servo motor model.*

7.2.1 Implementation of the Servo Motor Model

The model is based on a discrete state space system and is build up using the structure seen from Figure .

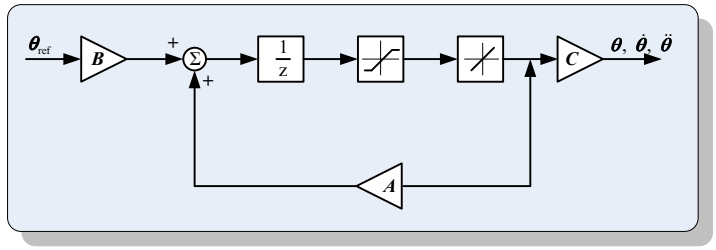


Figure 7.6: *State space structure of the servo motor model.*

The velocity is used to control the other states, and can be calculated as:

$$\dot{\theta}_{k+1} = c(\theta_{\text{ref}} - \theta_k) = c\theta_{\text{ref}} - c\theta_k \quad , \quad \dot{\theta}_{\text{min}}(\tau) \leq \dot{\theta}_{k+1} \leq \dot{\theta}_{\text{max}}(\tau) \tag{7.1}$$

where c is a constant controlling the first order behavior when $\boldsymbol{\theta}$ is close to $\boldsymbol{\theta}_{\text{ref}}$. As seen, the maximum and minimum velocity depends on $\boldsymbol{\tau}$. This relationship has to be found from measurements, which is done in Section 7.2.2. The velocity is implemented as a dynamic saturation which depend on the load $\boldsymbol{\tau}$.

The position is calculated from the velocity:

$$\theta_{k+1} = \theta_k + T_{\text{s}}\dot{\theta}_k \tag{7.2}$$

where T_{s} is the sampling time. The acceleration can be calculated by saving the velocity state one step back in time and calculate the difference between the new and old velocity divided by the sampling time:

$$\ddot{\theta}_{k+1} = \frac{\dot{\theta}_k - \dot{\theta}_{k-1}}{T_s} \quad , \quad -\ddot{\theta}_{\text{max}} \leq \ddot{\theta}_{k+1} \leq \ddot{\theta}_{\text{max}} \tag{7.3}$$

Since it is the velocity state that is being controlled, the maximum acceleration is imposed on the velocity as a rate limiter. Thus Equation (7.1) is also limited by:

$$-\ddot{\theta}_{\text{max}} \leq \frac{\dot{\theta}_k - \dot{\theta}_{k-1}}{T_{\text{s}}} \leq \ddot{\theta}_{\text{max}} \tag{7.4}$$

Equations 7.1-7.3 can be written in state space form with one extra state added to save the velocity:

$$\begin{array}{ccccccc} \boldsymbol{x_{k+1}} & = & \boldsymbol{A} & \boldsymbol{x_k} & + & \boldsymbol{B} & u_k \end{array}$$

$$\begin{bmatrix} \theta_{k+1} \\ \dot{\theta}_{k+1} \\ \dot{\theta}_k \\ \ddot{\theta}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T_s & 0 & 0 \\ -c & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \frac{1}{T_s} & -\frac{1}{T_s} & 0 \end{bmatrix} \begin{bmatrix} \theta_k \\ \dot{\theta}_k \\ \dot{\theta}_{k-1} \\ \ddot{\theta}_k \end{bmatrix} + \begin{bmatrix} 0 \\ c \\ 0 \\ 0 \end{bmatrix} \theta_{ref} \tag{7.5}$$

Because the outputs are position, velocity, and acceleration, the output equation can be written as:

$$\begin{array}{ccccc} \boldsymbol{y_k} & = & \boldsymbol{C} & \boldsymbol{x_k} \end{array}$$

$$\begin{bmatrix} \theta_{k+1} \\ \dot{\theta}_{k+1} \\ \ddot{\theta}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_{k+1} \\ \dot{\theta}_{k+1} \\ \dot{\theta}_k \\ \ddot{\theta}_{k+1} \end{bmatrix} \tag{7.6}$$

7.2.2 Determination of Servo Motor Parameters

The parameters to be determined are loads impact on the maximum angular velocity, $\dot{\theta}_{\text{max}}(\tau)$, and minimum angular velocity, $\dot{\theta}_{\text{min}}(\tau)$, velocity, the maximum acceleration, $\ddot{\theta}_{\text{max}}$, and the constant c. The maximum velocity with no load, specified by the manufacturer, are given Table 5.1 on page 38 to be:

$$\dot{\theta}_{\text{max,HSR-5995TG}} = 8.73\,\text{rad/s} \tag{7.7}$$

$$\dot{\theta}_{\text{max,HS-645MG}} = 5.22\,\text{rad/s} \tag{7.8}$$

In order to determine torque dependency of velocity, a number of measurements are taken, where a step input is given to the servo motor, while imposing it to a known load. In Figure 7.7(a) the result from one of these measurements is shown. In this test no torque is applied. The HSR-5995TG is given a step from $0\,\text{rad}$ to $2.1\,\text{rad}$. Note that the result shown is filtered by the filter explained in Section B.1 on page 175.

Having filtered the position measurement, it is also possible to differentiate it in order to obtain the angular velocity and angular acceleration, which are shown in Figure 7.7(b) and Figure 7.7(c). The maximum velocity for the unloaded HSR-5995TG servo motor, $\dot{\theta}_{\text{max,HSR-5995TG}}$, reached in Figure 7.7(b) is $8.12\,\text{rad/s}$, which is marked with a dashed line. Thus there is accordance between the measurements and the specifications provided in the data sheet, Equation (7.7).

Several measurements are performed with different loads applied to the HSR-5995TG servo motor, the maximum velocity is found for the given torque. These can be seen in Figure 7.8, together with a first order polynomial fit.

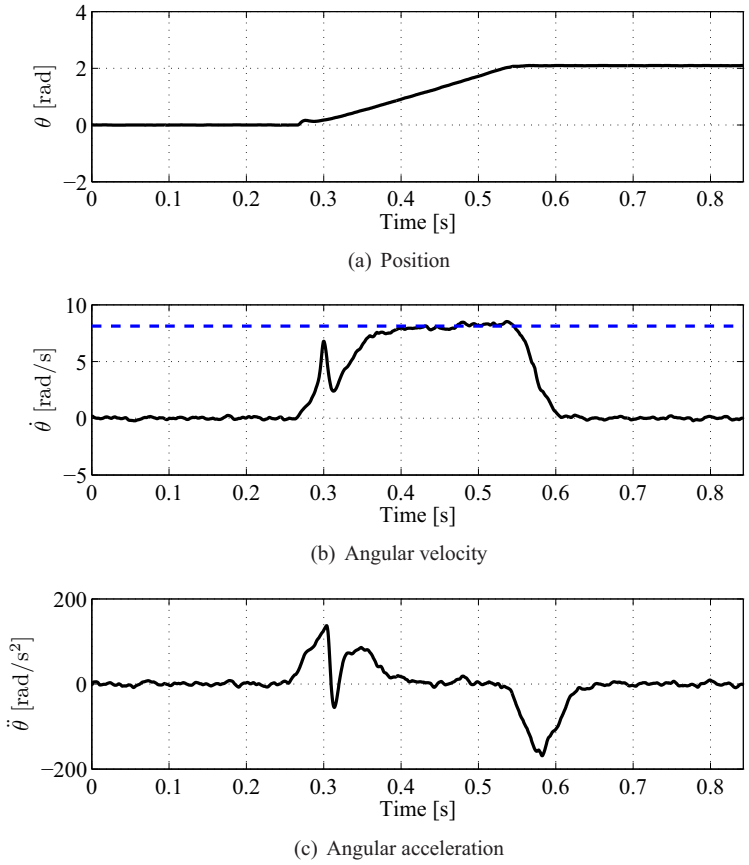


Figure 7.7: *Measurement taken from the servo potentiometer and differentiated to obtain velocity and acceleration. No load is applied.*

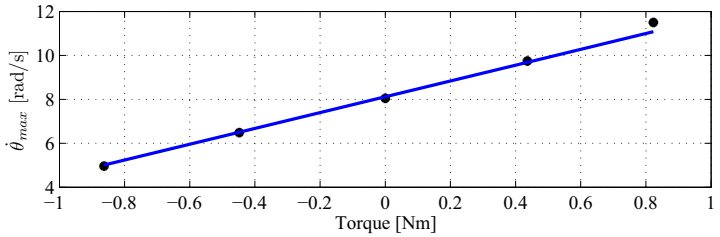


Figure 7.8: *The relationship between the torque and the maximum velocity. * marks the point found from the measurements, and the straight line is the fit.*

The first order polynomial fit is found to be:

$$\dot{\theta}_{\max}(\tau) = 3.6\tau + 8.12[\,\mathrm{rad/s}] \tag{7.9}$$

Which is implemented as a dynamical saturation. The parameters for the two servo motors are summarized in Table 5.1 on page 38.

7.2.3 Verification of Servo Motor Model

The developed servo motor model has been verified through a comparison with the real motor. A test has been performed for each type of servo motor. A complete description of the tests performed and the obtained results can be found in Appendix F in Section F.2. The main results were that the model of the HSR-5995TG fits the real motor measurements with an average R^2-value of 0.998 and the model of the HS-645MG fits with an average of 0.932. A plot showing the output of the model and the measured output of the servo, when given the same input can be seen in Figure 7.9.

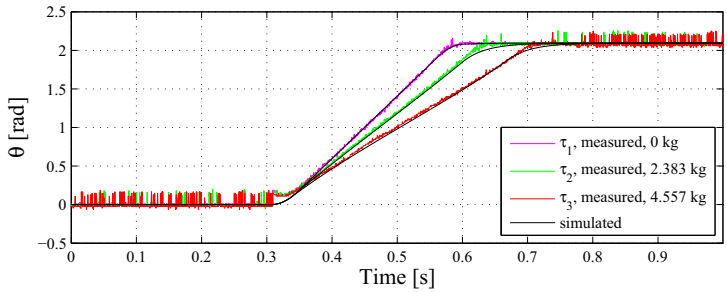


Figure 7.9: *Comparison between measurement and simulated result for three different loads, for the HSR-5995TG*

The servo motor model is therefore considered verified and correct.

7.3 Kinematic Model

The kinematic model is a transformation from joint space to Cartesian space. This transformation is necessary to calculate the positions of the CoM of the links, given the rotation of the joints. That is; the angular positions of the servos, $\boldsymbol{\theta}$, are transformed into positions of the CoM of each link, $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$, given in some global Cartesian frame. Link CoM are later used for dynamical modeling. In this project the kinematic model returns the generalized coordinate vector, where the generalized coordinate vector is defined as: $\boldsymbol{q} = [\boldsymbol{x}\ \boldsymbol{y}\ \boldsymbol{z}\ \boldsymbol{\theta}]^{\mathrm{T}}$. Thus, having 21 joints and 22 links, $\boldsymbol{q}$ becomes an 87x1 vector. The kinematic model also returns the generalized velocity vector $\dot{\boldsymbol{q}}$ and the generalized acceleration vector $\ddot{\boldsymbol{q}}$. The input and output of the kinematic model is illustrated in Figure 7.10.

However, before the kinematic model can be obtained, it is necessary to present the mechanical structure of the biped robot in a convenient mathematical form. The following section will deal with this representation.

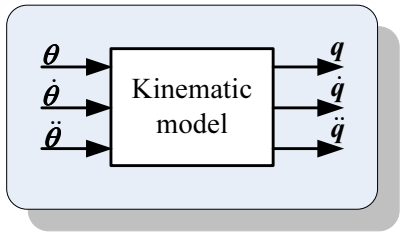


Figure 7.10: *The kinematic model is a transformation from joint coordinates to generalized coordinates.*

7.3.1 Representation of the Mechanical Structure

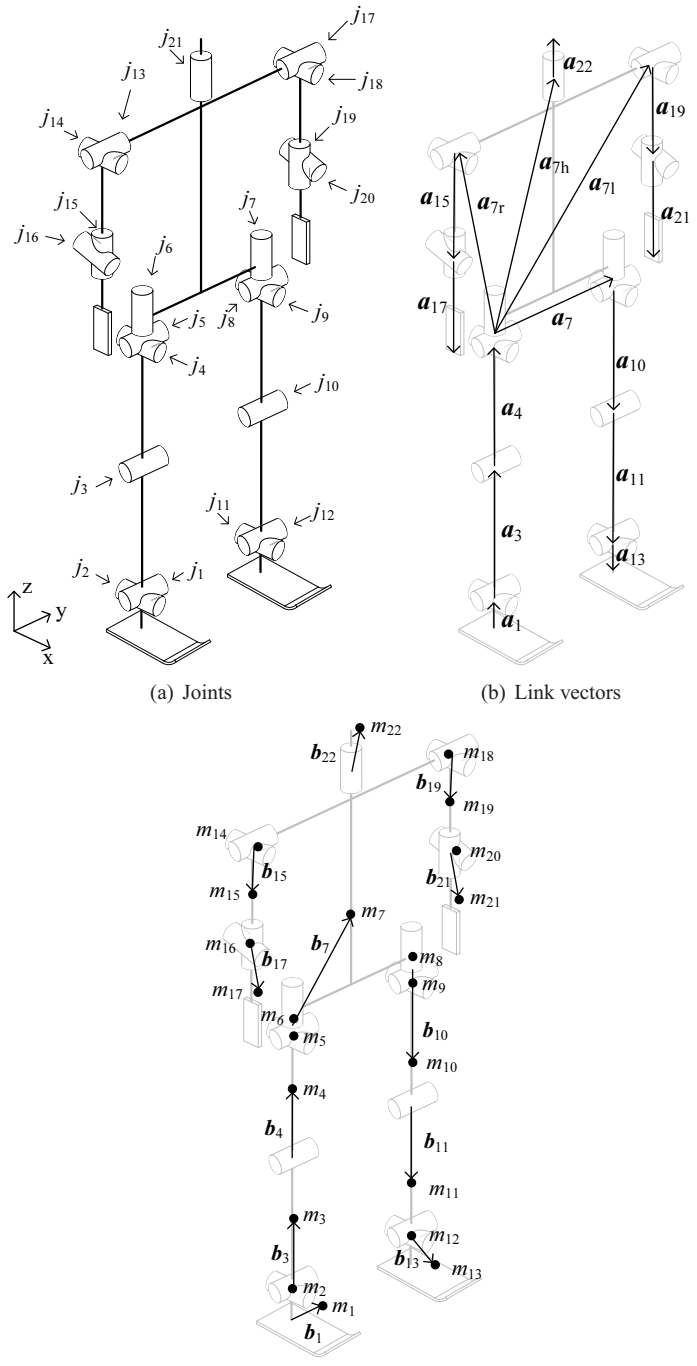
The biped robot can be considered a collection of links and joints collected in a unique structure. All the names are assigned with basis in a situation where the biped robot is in SSP-R, thus the mechanical structure is seen as a kinematic chain starting from a reference frame placed on the sole of the right foot. The kinematic model developed in this section will be a model describing the kinematics when in SSP-R. With few changes the model can be extended to include the DSP and SSP-L. This will be explained at the end of this section.

In Figure 7.11(a) the names of all the joints are illustrated. The joints are named in accordance with the name of the actuator in the specific joint. The name of the actuators can be seen in Figure C.3 on page 188. Now joint i has a unique reference name, j_i. All joints are actuated revolute joints, controlled by a servo motor. In Figure 7.11(b) the link vectors are illustrated, the link vector a_i is the vector j_{i-1} to j_i. Thus link i is the link connecting joint j_{i-1} and j_i. A joint with n DoF, will in this model be threated as n joints with 1 DoF, placed in the same point, [Craig, 1989]. Therefore not all link vectors are illustrated in Figure 7.11(b), since the lengths are zero. Following joints have multiple DoF; the ankle joints ($j_1 - j_2$ and $j_{11} - j_{12}$); the hip joints ($j_4 - j_5 - j_6$) and $j_7 - j_8 - j_9$); the shoulder joints ($j_{13} - j_{14}$ and $j_{17} - j_{18}$) and the elbow joints ($j_{15} - j_{16}$ and $j_{19} - j_{20}$).

Link vectors [mm]	**CoM vectors [mm]**	**Masses [g]**	**Moments of inertia [kg/mm^2]**
$\boldsymbol{a}_1 = [0\ 0\ 45]^T$	$\boldsymbol{b}_1 = [9.2\ 0\ 13]^T$	$m_1 = 43.4$	$\boldsymbol{I}_1 = [0\ 0\ 0]^T$
$\boldsymbol{a}_2 = [0\ 0\ 0]^T$	$\boldsymbol{b}_2 = [3.5\ 0\ 6.6]^T$	$m_2 = 137$	$\boldsymbol{I}_2 = [47\ 51\ 34]^T$
$\boldsymbol{a}_3 = [-10.6\ 0\ 86.4]^T$	$\boldsymbol{b}_3 = [0\ 0\ 61.7]^T$	$m_3 = 195.3$	$\boldsymbol{I}_3 = [872\ 876\ 56]^T$
$\boldsymbol{a}_4 = [10.6\ 0\ 139.1]^T$	$\boldsymbol{b}_4 = [14.9\ 0\ 71.4]^T$	$m_4 = 209.6$	$\boldsymbol{I}_4 = [1299\ 1320\ 124]^T$
$\boldsymbol{a}_5 = [0\ 0\ 0]^T$	$\boldsymbol{b}_5 = [3.5\ 0\ -6.6]^T$	$m_5 = 137$	$\boldsymbol{I}_5 = [47\ 52\ 34]^T$
$\boldsymbol{a}_6 = [0\ 0\ 0]^T$	$\boldsymbol{b}_6 = [0\ 0\ 27.4]^T$	$m_6 = 18.2$	$\boldsymbol{I}_6 = [25\ 20\ 6]^T$
$\boldsymbol{a}_8 = [0\ 0\ 0]^T$	$\boldsymbol{b}_8 = [0\ 0\ 27.4]^T$	$m_8 = 18.2$	$\boldsymbol{I}_8 = [25\ 20\ 6]^T$
$\boldsymbol{a}_9 = [0\ 0\ 0]^T$	$\boldsymbol{b}_9 = [3.5\ 0\ -6.6]^T$	$m_9 = 137$	$\boldsymbol{I}_9 = [47\ 52\ 34]^T$
$\boldsymbol{a}_{10} = [-10.6\ 0\ 139.1]^T$	$\boldsymbol{b}_{10} = [0\ 0\ -67.3]^T$	$m_{10} = 209.6$	$\boldsymbol{I}_{10} = [1182\ 1158\ 79]^T$
$\boldsymbol{a}_{11} = [10.6\ 0\ -86.4]^T$	$\boldsymbol{b}_{11} = [11.4\ 0\ -24.2]^T$	$m_{11} = 195.3$	$\boldsymbol{I}_{11} = [243\ 270\ 80]^T$
$\boldsymbol{a}_{12} = [0\ 0\ 0]^T$	$\boldsymbol{b}_{12} = [3.5\ 0\ 6.6]^T$	$m_{12} = 137$	$\boldsymbol{I}_{12} = [48\ 52\ 35]^T$
$\boldsymbol{a}_{13} = [0\ -63\ 192.7]^T$	$\boldsymbol{b}_{13} = [9.2\ 0\ -32]^T$	$m_{13} = 43.4$	$\boldsymbol{I}_{13} = [14\ 21\ 22]^T$

Table 7.1: *Parameters for the legs of the biped robot. These have been calculated in SolidWorks.*

In Figure 7.11(c) the CoM vectors are illustrated. A CoM vector is a vector, $\boldsymbol{b}_i$, specifying the CoM of link i relative to j_{i-1}. In Figure 7.11(c) the mass of the link i, called m_i, is also specified. In Table 7.1, Table 7.2 and Table 7.3 the sizes of all the parame-



(a) Joints (b) Link vectors

(c) Mass and CoM vectors

Figure 7.11: *Definitions of joints, link vectors and CoM vectors.*

ters illustrated in Figure 7.11 are given along with the principal moment of inertia for each link. All parameters are found from measurement of the robot, while moments of inertia is found from a model of the biped robot drawn in SolidWorks.

Link vectors [mm]	CoM vectors [mm]	Masses [g]	Moments of inertia [kg/mm^2]
$\boldsymbol{a}_7 = [0\ 82\ 0]^{\mathrm{T}}$	$\boldsymbol{b}_7 = [1.5\ 41\ 130.8]^{\mathrm{T}}$	$m_7 = 1213.6$	$\boldsymbol{I}_7 = [6799\ 2832\ 4526]^{\mathrm{T}}$
$\boldsymbol{a}_{7\mathrm{r}} = [0\ -63\ 192.7]^{\mathrm{T}}$			
$\boldsymbol{a}_{7\mathrm{l}} = [0\ 145\ 192.7]^{\mathrm{T}}$			
$\boldsymbol{a}_{7\mathrm{h}} = [0\ 41\ 219.4]^{\mathrm{T}}$			
$\boldsymbol{a}_{22} = [0\ 0\ 84.5]^{\mathrm{T}}$	$\boldsymbol{b}_{22} = [15.2\ 0\ 48]^{\mathrm{T}}$	$m_{22} = 15.6$	$\boldsymbol{I}_{22} = [14\ 14\ 4]^{\mathrm{T}}$

Table 7.2: *Parameters for the torso and head of the robot.*

It should be noted that the masses and principal moments of inertia given in this section are of no interest in relation to the kinematic model, since it only describes the motion of the system, with no consideration of forces that cause this motion. These parameters will be used in Section 7.4 where the derivation of the dynamical model of the biped robot is described.

Link vectors [mm]	CoM vectors [mm]	Masses [g]	Moments of inertia [kg/mm^2]
$\boldsymbol{a}_{14} = [0\ 0\ 0]^{\mathrm{T}}$	$\boldsymbol{b}_{14} = [-3\ 0\ -6.4]^{\mathrm{T}}$	$m_{14} = 62.7$	$\boldsymbol{I}_{14} = [13\ 15\ 6]^{\mathrm{T}}$
$\boldsymbol{a}_{15} = [0\ 0\ -114.8]^{\mathrm{T}}$	$\boldsymbol{b}_{15} = [0\ 0\ -17]^{\mathrm{T}}$	$m_{15} = 11.3$	$\boldsymbol{I}_{15} = [6\ 12\ 8]^{\mathrm{T}}$
$\boldsymbol{a}_{16} = [0\ 0\ 0]^{\mathrm{T}}$	$\boldsymbol{b}_{16} = [8.5\ 0\ 43]^{\mathrm{T}}$	$m_{16} = 78.2$	$\boldsymbol{I}_{16} = [17\ 29\ 18]^{\mathrm{T}}$
$\boldsymbol{a}_{17} = [0\ 0\ 107.1]^{\mathrm{T}}$	$\boldsymbol{b}_{17} = [7\ 0\ -21.2]^{\mathrm{T}}$	$m_{17} = 83.3$	$\boldsymbol{I}_{17} = [76\ 77\ 11]^{\mathrm{T}}$
$\boldsymbol{a}_{18} = [0\ 0\ 0]^{\mathrm{T}}$	$\boldsymbol{b}_{18} = [-3\ 0\ -6.4]^{\mathrm{T}}$	$m_{18} = 62.7$	$\boldsymbol{I}_{18} = [13\ 15\ 6]^{\mathrm{T}}$
$\boldsymbol{a}_{19} = [0\ 0\ -114.8]^{\mathrm{T}}$	$\boldsymbol{b}_{19} = [0\ 0\ -17]^{\mathrm{T}}$	$m_{19} = 11.3$	$\boldsymbol{I}_{19} = [6\ 12\ 8]^{\mathrm{T}}$
$\boldsymbol{a}_{20} = [0\ 0\ 0]^{\mathrm{T}}$	$\boldsymbol{b}_{20} = [8.5\ 0\ 43]^{\mathrm{T}}$	$m_{20} = 78.2$	$\boldsymbol{I}_{20} = [17\ 29\ 18]^{\mathrm{T}}$
$\boldsymbol{a}_{21} = [0\ 0\ 107.1]^{\mathrm{T}}$	$\boldsymbol{b}_{21} = [7\ 0\ -21.2]^{\mathrm{T}}$	$m_{21} = 83.3$	$\boldsymbol{I}_{21} = [76\ 77\ 11]^{\mathrm{T}}$

Table 7.3: *Parameters for the arms of the robot.*

In Figure 7.12(a) the zero-position and the direction of rotation is defined for all rotations around the x-axis, in Figure 7.12(b) for the rotations around the y-axis, and in Figure 7.12(c) for the yaw movements (joints rotating around the z-axis). The dashed lines indicates the ankle each link should be positioned in, in order to assume zero-position.

7.3.2 Attacing Frames

In Figure 7.11 the constant parameters of the biped robot are defined and in Figure 7.12 the variable parameters of the biped robot are defined. Next a representation of the positions of all links relative to a global reference frame is given. As mentioned in Section 7.1.2, the choice of reference frame varies in the literature. It is chosen to consider the biped robot as an open loop serial chain from the supporting foot as in [Mu and Wu, 2004]. In DSP both right and left foot are individually chosen as the supporting foot, and a combination of these two makes the DSP model. This means that whenever the right foot is on the ground (in SSP-R) the generalized positions, $\boldsymbol{q}$, are given with respect to frame $\{0\}$, and when in SSP-L, $\boldsymbol{q}$ are given in frame $\{13\}$. When in DSP, both models are used. In Figure 7.13 the biped robot is shown in SSP-R and frame $\{0\}$ is attached to the sole of the right foot.

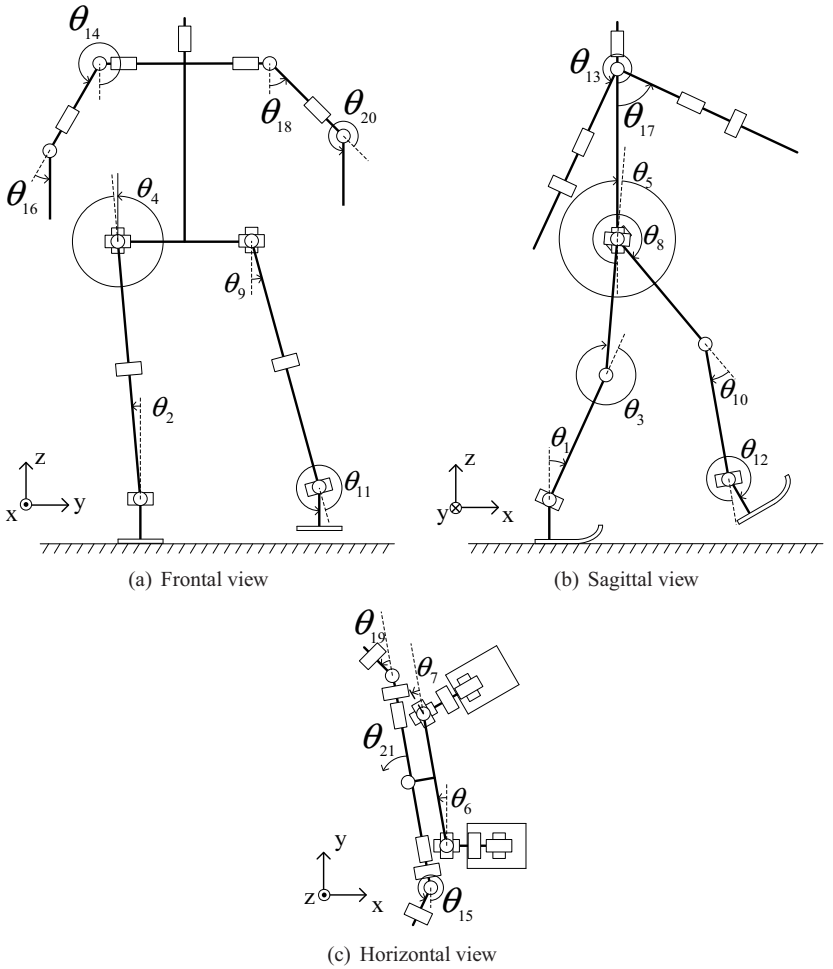


Figure 7.12: *Definition of the angles in the three planes.*

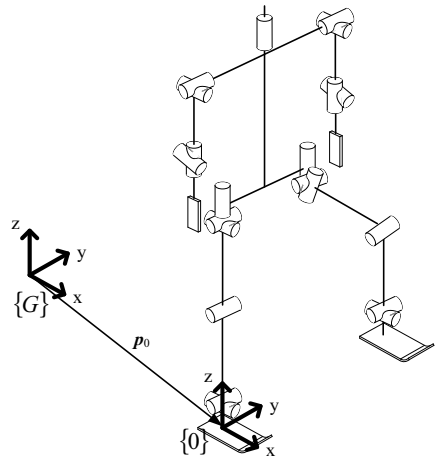


Figure 7.13: *In SSP-R the biped robot is represented in frame* $\{0\}$*. The position of frame* $\{0\}$ *is given in a floor fixed global reference frame* $\{G\}$*.*

To keep track of the global position a floor fixed global reference frame, frame $\{G\}$, is introduced. The position of frame $\{G\}$ is assigned as the initial position of frame $\{0\}$. To describe the orientation of the links, a link-frame is attached to each link, with its origin placed in the joints. Thus frame $\{i\}$ is placed with its origin in j_i and attached to link $i+1$. In [Craig, 1989] and in [Christensen et al., 2006] the link-frames are oriented such that they rotate around the z-axis. In this project it is chosen to orient the link-frames, such that they are aligned with the reference frame when all angles are zero ($\boldsymbol{\theta} = \mathbf{0}$), meaning that the robot is in zero-position. The advantage is, that the rotation between two succeeding link-frames is solely described by the joint angle. In Table 7.4 the axis of rotation is stated for each link-frame.

Frame	**Axis of rotation**
$\{2\},\{4\},\{9\},\{11\},\{14\},\{16\},\{18\},\{20\}$	x
$\{1\},\{3\},\{5\},\{8\},\{10\},\{12\},\{13\},\{17\}$	y
$\{6\},\{7\},\{15\},\{19\},\{21\}$	z

Table 7.4: *Axis of rotation for the link-frames.*

The rotation matrix:

$$ {}^{i-1}_{i}\boldsymbol{R} = \left[{}^{i-1}\hat{\boldsymbol{x}}_i \; {}^{i-1}\hat{\boldsymbol{y}}_i \; {}^{i-1}\hat{\boldsymbol{z}}_i \right] \tag{7.10} $$

describes the orientation of the axis of frame $\{i\}$ relative to frame $\{i-1\}$. ${}^{i-1}_{i}\boldsymbol{R}$ is an Euler rotation of θ_i and for rotation around the x-axis it is defined as [Craig, 1989]:

$$ {}^{i-1}_{i}\boldsymbol{R}_{\mathrm{x}}(\theta_i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_i & -\sin\theta_i \\ 0 & \sin\theta_i & \cos\theta_i \end{bmatrix} \tag{7.11} $$

about the y-axis:

$$ {}^{i-1}_{i}\boldsymbol{R}_{\text{y}}(\theta_i) = \begin{bmatrix} \cos\theta_i & 0 & \sin\theta_i \\ 0 & 1 & 0 \\ -\sin\theta_i & 0 & \cos\theta_i \end{bmatrix} \tag{7.12} $$

and around the z-axis:

$$ {}^{i-1}_{i}\boldsymbol{R}_{\text{z}}(\theta_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.13} $$

From the above definitions of rotation matrices, it is also possible to find the orientation of frame $\{i+n\}$ relative to frame $\{i\}$. This is given as:

$$ {}^{i}_{i+n}\boldsymbol{R}(\theta_{i+1}, \theta_{i+2}, ..., \theta_{i+n}) = {}^{i}_{i+1}\boldsymbol{R}(\theta_{i+1})\, {}^{i+1}_{i+2}\boldsymbol{R}(\theta_{i+2}) \, ... \, {}^{i+n-1}_{i+n}\boldsymbol{R}(\theta_{i+n}) \tag{7.14} $$

The kinematics of the robot is dependent on the phase of the system, if the phase is SSP-R, θ_1 is the first joint angle in the chain, but if the phase is SSP-L, then θ_{12} is the first joint angle in the chain. As a consequence, when the notation ${}^{i-1}_{i}\boldsymbol{R}$ is used, i does not refer to the name of the link, but the location in the kinematic chain. When in SSP-R there are the following four kinematic chains:

$$ \begin{cases} [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12] & \text{, for} \quad i \leq 12 \\ [1\ 2\ 3\ 4\ 5\ 6\ 13\ 14\ 15\ 16] & \text{, for} \quad 13 \leq i \leq 16 \\ [1\ 2\ 3\ 4\ 5\ 6\ 17\ 18\ 19\ 20] & \text{, for} \quad 17 \leq i \leq 20 \\ [1\ 2\ 3\ 4\ 5\ 6\ 21] & \text{, for} \quad i = 21 \end{cases} \tag{7.15} $$

and when the phase is SSP-L the following four chains are represented:

$$ \begin{cases} [12\ 11\ 10\ 9\ 8\ 7\ 6\ 5\ 4\ 3\ 2\ 1] & \text{, for} \quad i \leq 12 \\ [12\ 11\ 10\ 9\ 8\ 7\ 13\ 14\ 15\ 16] & \text{, for} \quad 13 \leq i \leq 16 \\ [12\ 11\ 10\ 9\ 8\ 7\ 17\ 18\ 19\ 20] & \text{, for} \quad 17 \leq i \leq 20 \\ [12\ 11\ 10\ 9\ 8\ 7\ 21] & \text{, for} \quad i = 21 \end{cases} \tag{7.16} $$

Thus if the phase is SSP-R and $i = 13$, the ${}^{i-1}_{i}\boldsymbol{R}$ means ${}^{6}_{13}\boldsymbol{R}$ (the rotation matrix from the right hip to the right shoulder).

To get the right output of the kinematic model, it is necessary to know the phase of the system, and thereby know if the supporting foot is the right foot or the left foot. The will be handled in Section 7.5, where a phase estimator is designed.

7.3.3 The Generalized Coordinates

The output of the kinematic model is the generalized coordinates, which is chosen to be the positions of all CoM's and all joint angles. The generalized position vector, $\boldsymbol{q}$, is defined to:

$$ \boldsymbol{q}_{\,87\times1} \equiv \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \boldsymbol{\theta} \end{bmatrix} \tag{7.17} $$

where

$$\boldsymbol{x} = [x_1\ x_2\ ...\ x_{22}]^{\mathrm{T}}\ ,\ \boldsymbol{y} = [y_1\ y_2\ ...\ y_{22}]^{\mathrm{T}}\ ,\ \boldsymbol{z} = [z_1\ z_2\ ...\ z_{22}]^{\mathrm{T}} \tag{7.18}$$

contains the x-, y- and z-coordinates of all CoM's, and:

$$\boldsymbol{\theta} = [\theta_1\ \theta_2\ ...\ \theta_{21}]^{\mathrm{T}} \tag{7.19}$$

contains all the joint angles.

With reference to Figure 7.11 on page 73 the generalized coordinate vector can now be determined. The iterative equation for $\boldsymbol{p}_{i+1}$ (the position of CoM of link $i+1$) is defined as:

$$\boldsymbol{p}_{i+1} = \begin{bmatrix} x_{i+1} \\ y_{i+1} \\ z_{i+1} \end{bmatrix} = {}^{0}_{i}\boldsymbol{R}\boldsymbol{b}_i + \boldsymbol{p}_{\mathrm{j}_i} \tag{7.20}$$

where $\boldsymbol{p}_{\mathrm{j}_i}$ is the position of joint i defined as:

$$\boldsymbol{p}_{\mathrm{j}_i} = \begin{bmatrix} x_{\mathrm{j}_i} \\ y_{\mathrm{j}_i} \\ z_{\mathrm{j}_i} \end{bmatrix} = {}^{0}_{i-1}\boldsymbol{R}\boldsymbol{a}_i + \boldsymbol{p}_{\mathrm{j}_{i-1}} \tag{7.21}$$

From Equation (7.20) and Equation (7.21) the generalized position vector defined in Equation (7.17) can now be found.

From Figure 7.10 it can be seen that the kinematic model also returns the generalized velocity vector, $\dot{\boldsymbol{q}}$, and the generalized acceleration vector, $\ddot{\boldsymbol{q}}$. These are defined as follows:

$$\dot{\boldsymbol{q}} \equiv \frac{\partial \boldsymbol{q}}{\partial t}\ ,\ \ddot{\boldsymbol{q}} \equiv \frac{\partial^2 \boldsymbol{q}}{\partial t^2} \tag{7.22}$$

Now the theoretical basis for the kinematic model is established. In Appendix E on page 195 the kinematic model will be derived for a simple 4 DoF biped robot with 3 links. Its purpose is to clarify the derivation of the complete model.

7.3.4 Verification of the Kinematic Model

A test have been performed in order to verify the kinematic model developed above. The complete description of the test and a review of the results can be found in Appendix F in Section F.4 on page 212. To verify the model four different tests have been performed, and sensors have been mounted temporary on the robot for test purpose only. The result from one of these tests can be seen from Figure 7.14.

The main result is that the kinematic model fits the measured output of the real system with a average R^2-value of 0.702, where 1 is the best. Considering the noise on the measurements along with all the uncertainties in a test like this, the kinematic model is considered verified.

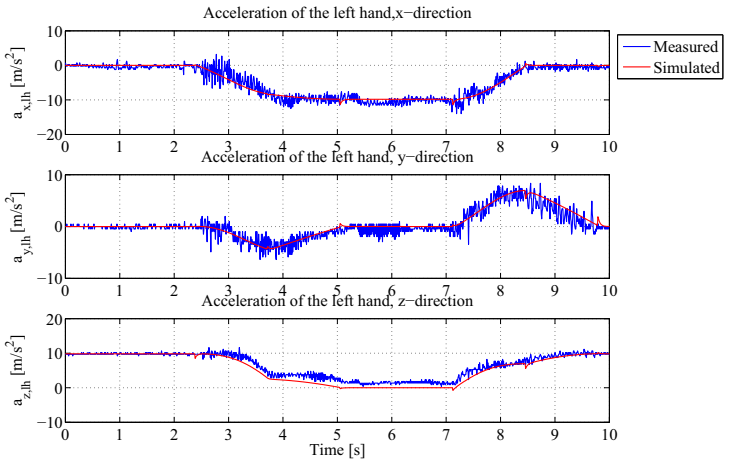


Figure 7.14: *Result of the kinematic test of the left hand. The blue line is the measured output of the accelerometer and the red line is the output from the model.*

7.4 Dynamical Model

This section explains the dynamical model of the biped robot. First a general introduction to the method is given. Then the theory behind Lagrangian modeling is covered and following the approach for SSP and DSP are explained. As the biped robot has 21 DoF, the calculated equations expand to such a degree that it has been chosen to make a motivating example for a smaller, four joint robot, to further emphasize the chosen approach. This example is a continuation of the example given in Section E.1 on page 195. At the end, the full 21 joints dynamical model is verified.

Dynamical modeling is to calculate the motion of the system when an external force is applied. This has been a research topic for several decades [Chaillet et al., 1994], [McGeer, 1990]. The most common approach is to investigate the response of a biped robot when the actuators exert a certain force. For this, both Newtonian and Lagrangian approaches have been taken. In [Christensen et al., 2006] and [Mu and Wu, 2004] a Lagrangian approach was taken and it was found that the system dynamics could be described by Equation (7.23):

$$\boldsymbol{\tau} = \boldsymbol{A}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{B}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \boldsymbol{C}(\boldsymbol{\theta}) \tag{7.23}$$

Where $\boldsymbol{A}$ is a matrix describing the system inertia, $\boldsymbol{B}$ is related to the centrifugal and Coriolis terms, $\boldsymbol{C}$ is a vector related to the gravity and $\boldsymbol{\tau}$ is the external forces applied to the system. In [Mu and Wu, 2004] it was found that the motion of the system could be described by Equation (7.24), which is found by rewriting Equation (7.23):

$$\ddot{\boldsymbol{\theta}} = \boldsymbol{A}(\boldsymbol{\theta})^{-1}(\boldsymbol{\tau} - \boldsymbol{B}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} - \boldsymbol{C}(\boldsymbol{\theta})) \tag{7.24}$$

Equation (7.24) describes the acceleration $\ddot{\boldsymbol{\theta}}$ of the biped robot under the actuation of the external torque $\boldsymbol{\tau}$. This equation is therefore considered as a dynamical model, where Equation (7.23) is an inverse dynamical model describing the torques needed to make the biped robot undergo a certain motion.

As the purpose of the model is to give a description of the robot system, usable for off-line testing of controllers and off-line trajectory generation, it is chosen to develop an inverse dynamic model of the biped robot and the input/output relation of the dynamical system can then be seen from Figure 7.15.

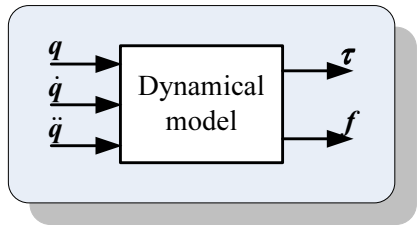


Figure 7.15: *Input/output relation of the dynamical model*

But as the biped robot system can be seen as a hybrid system with three phases (SSP-L, SSP-R and DSP) and the inverse dynamical model only describes the biped robot system when in SSP, another solution has to be found to described the forces acting on the system when in DSP. If a dynamic model was to be found, a common approach is to introduce a constraint on the system resulting in an altered motion because of the constraint force acting on the system. In [Christensen et al., 2006] and [Mu and Wu, 2004] this was investigated and it was found that by using Lagrangian multipliers the motion of the system could be described by:

$$\ddot{\boldsymbol{\theta}} = \boldsymbol{A}(\boldsymbol{\theta})^{-1}((\boldsymbol{\tau} + \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{\theta})\boldsymbol{\lambda}) - \boldsymbol{B}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} - \boldsymbol{C}(\boldsymbol{\theta})) \tag{7.25}$$

Where $\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{\theta})\boldsymbol{\lambda}$ is the contribution from the constraint force, $\boldsymbol{\lambda}$ is the Lagrangian multiplier and $\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{\theta})$ is the Jacobian matrix of the constraints with respect to $\boldsymbol{\theta}$. The problem with using the Lagrangian multipliers is that this approach can be rather cumbersome, as the multiplier has to be solved for and then inserted into the Equation (7.25). Another approach is to use Udwadia-Kalaba equations to describing motion of a dynamic system under constrains, which follows the same approach as with the Lagrangian multipliers by introducing a force of constraints [Udwadia and Kalaba, 2001].

But with this approach it is not necessary to compute the Lagrangian multiplier, and the approach is therefore more straight forward. But the problem with both of the approaches is that they can only be used for the dynamical model, and are non-invertible. An approach valid for estimating the external forces in the DSP has not been found in the literature and therefore a solution, to solve this problem, has to be found.

7.4.1 Dynamical Modeling Approach

As earlier explained it is chosen to use Lagrangian modeling to describe the dynamical system. In this section an introduction to the Lagrangian approach is given based on [Wisniewski, 2006]. Lagrange modeling is based on Hamilton's principle. The principle states that the integrated difference between the kinetic energy, K, minus potential energy, P, is always minimal. The motion of a mechanical system from time a to b can be expressed by:

$$I(t, \boldsymbol{q}, \dot{\boldsymbol{q}}) = \int_{\mathrm{a}}^{\mathrm{b}} L(t, \boldsymbol{q}, \dot{q}) dt \tag{7.26}$$

L is called the Lagrangian and is defined as the energy difference of the mechanical system (Kinetic energy (K) minus potential energy (P)) and q is the generalized co-ordinate. Hamilton's principle describes the path used to come from a to b, such that the difference in energy is minimal. Thus, $I(t, \boldsymbol{q}, \dot{\boldsymbol{q}})$ has to be at a critical point (a point where the derivative is zero), and the derivative, i.e. a small change of path, is zero for the true path (the path the system undergoes). A small change of path can be described as:

$$I(t, \boldsymbol{q} + \epsilon\eta, \dot{\boldsymbol{q}} + \epsilon\dot{\eta}) \quad , \quad \epsilon \to 0 \tag{7.27}$$

for any arbitrary path η.

The derivative of Equation (7.27) is then given as:

$$\frac{d}{d\epsilon} I(t, \boldsymbol{q} + \epsilon\eta, \dot{\boldsymbol{q}} + \epsilon\dot{\eta}) = 0 \quad , \quad \epsilon \to 0 \tag{7.28}$$

Substituting Equation (7.26) into Equation (7.28) and performing the differentiation, the following expression, which applies to all the generalized co-ordinates q_i, can be obtained:

$$\int_{\mathrm{a}}^{\mathrm{b}} \left(\frac{\partial \boldsymbol{L}}{\partial q_i} - \frac{d}{dt} \frac{\partial \boldsymbol{L}}{\partial \dot{q}_i} \right) \eta dt = 0 \tag{7.29}$$

Now the *Fundamental Lemma of Calculus of Variations* is applied to Equation (7.29). This lemma basically states that if the expression is true, then the expression in the brackets has to be zero as well:

$$\frac{\partial \boldsymbol{L}}{\partial q_i} - \frac{d}{dt} \frac{\partial \boldsymbol{L}}{\partial \dot{q}_i} = 0 \tag{7.30}$$

This applies to all the generalized co-ordinates. The expression can be rewritten as a vector equation describing the motion of a mechanical system with n generalized coordinates under influence of no external force.

$$\begin{bmatrix} \frac{\partial \boldsymbol{L}}{\partial q_1} - \frac{d}{dt} \frac{\partial \boldsymbol{L}}{\partial \dot{q}_1} \\ \frac{\partial \boldsymbol{L}}{\partial q_2} - \frac{d}{dt} \frac{\partial \boldsymbol{L}}{\partial \dot{q}_2} \\ \vdots \\ \frac{\partial \boldsymbol{L}}{\partial q_{\mathrm{n}}} - \frac{d}{dt} \frac{\partial \boldsymbol{L}}{\partial \dot{q}_{\mathrm{n}}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{7.31}$$

If the mechanical system is influenced by external forces the Lagrange-d'Alembert principle can be applied. The principle states that the external force field, $\boldsymbol{Q}$, can be taken into account as follows:

$$I(t, \boldsymbol{q}, \dot{\boldsymbol{q}}) = \int_{\mathrm{a}}^{\mathrm{b}} \boldsymbol{L}(t, \boldsymbol{q}, \dot{\boldsymbol{q}}) dt + \int_{a}^{b} \boldsymbol{Q}(t, \boldsymbol{q}, \dot{\boldsymbol{q}}) dt \tag{7.32}$$

The force field in Equation (7.32) can be interpreted as the displacement from the minimal energy state in the motion of the system, done by the work of the external force $\boldsymbol{Q}$. Using the same method as for the system under influence of no external force,

Equation (7.32) leads to the following equation describing the motion of a mechanical system under influence of an external force:

$$\frac{\partial \boldsymbol{L}}{\partial q_i} - \frac{d}{dt}\frac{\partial \boldsymbol{L}}{\partial \dot{q}_i} = Q_i \tag{7.33}$$

In the case of a robot with joint angles as generalized co-ordinates, the external force is the torque applied to each servo, and hence Equation (7.33) can be rewritten into vector equation describing the motion of the robot:

$$\begin{bmatrix} \frac{\partial \boldsymbol{L}}{\partial q_1} - \frac{d}{dt}\frac{\partial \boldsymbol{L}}{\partial \dot{q}_1} \\ \frac{\partial \boldsymbol{L}}{\partial q_2} - \frac{d}{dt}\frac{\partial \boldsymbol{L}}{\partial \dot{q}_2} \\ \vdots \\ \frac{\partial \boldsymbol{L}}{\partial q_\mathrm{n}} - \frac{d}{dt}\frac{\partial \boldsymbol{L}}{\partial \dot{q}_\mathrm{n}} \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_\mathrm{n} \end{bmatrix} \tag{7.34}$$

where τ_i is the torque provided by the servo motor controlling the i'th link.

As the model of the biped robot in DSP is a combination of SSP-L and SSP-R, a model of the biped robot in SSP has to be derived first. Hence the following section will use Equation (7.34) to develop a model of the biped robot in single support phase.

7.4.2 Single Support Phase Modeling

In this section an overview of the approach used in the SSP modeling will be given. This model has two different phases, namely when in SSP-L and SSP-R. Only the approach for the SSP-R phase will be given, as the approach for each phase is similar.

When using Lagrangian modeling it is the energy in the system that is considered, and thereby the model is derived from energy equations. As a biped robot is seen as a multi linked body connected by joints, the energy has to be considered for each link. From Figure 7.11 it is possible to see the different joints which connect the links of the robot. As earlier explained the Lagrangian can be found as the kinetic energy minus the potential energy. The kinetic energy of the i'th link can be calculated as in [Craig, 1989, p. 207], where all coordinates are given in Cartesian coordinates:

$$k_i = \frac{1}{2}\,\mathrm{m}_i\,\boldsymbol{v}_i^{\mathrm{T}}\boldsymbol{v}_i + \frac{1}{2}\,\boldsymbol{\omega}_i^{\mathrm{T}}\boldsymbol{I}_i\,\boldsymbol{\omega}_i = \frac{1}{2}\,\mathrm{m}_i\,(\dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2) + \frac{1}{2}\,\boldsymbol{\omega}_i^{\mathrm{T}}\boldsymbol{I}_i\,\boldsymbol{\omega}_i \tag{7.35}$$

The matrix, $\boldsymbol{I}_i$, is the inertia tensor of the i'th link, $\boldsymbol{v}_i$ is the linear velocity and $\boldsymbol{\omega}_i$ is the angular velocity vector of the link. The angular velocity can be found as [Craig, 1989, p. 207]:

$$\boldsymbol{\omega}_{i+1} = \boldsymbol{\omega}_i + {}^{i+1}_{0}\boldsymbol{R}\dot{\theta}_i\boldsymbol{\zeta} \tag{7.36}$$

Where ${}^{i+1}_{0}\boldsymbol{R}$ is the rotation from the base frame to frame $i+1$, found from Equation 7.14 on page 77 and $\boldsymbol{\zeta}$ is a vector denoting the axis which $\dot{\theta}_i$ rotates about. The kinetic energy of all links can be found as $K = \sum_{i=1}^{22} k_i$

The potential energy of link i is calculated as:

$$p_i = \quad \mathrm{m}_i \; \mathrm{g} \; z_i \tag{7.37}$$

The potential energy of all links is $P = \sum_{i=1}^{22} p_i$, and the Lagrangian is calculated as $L = K - P$. To calculate the external forces given by Equation (7.33), the Lagrangian is first differentiated with respect to the generalized coordinates as:

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{q}}_{\,1\times 87} = \left[\begin{array}{cccc} \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{z}} & \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{\theta}} \end{array} \right] \tag{7.38}$$

Notice that the Lagrangian differentiated with respect to the generalized coordinates becomes a 1×87. Now the derivate of the Lagrangian with respect to the velocity of the generalized coordinates are found. This is then differentiated with respect to time and yields:

$$\frac{\partial}{\partial t}\frac{\partial \boldsymbol{L}}{\partial \dot{\boldsymbol{q}}}_{\,1\times 87} = \left[\begin{array}{cccc} \frac{\partial}{\partial t}\frac{\partial \boldsymbol{L}}{\partial \dot{\boldsymbol{x}}} & \frac{\partial}{\partial t}\frac{\partial \boldsymbol{L}}{\partial \dot{\boldsymbol{y}}} & \frac{\partial}{\partial t}\frac{\partial \boldsymbol{L}}{\partial \dot{\boldsymbol{z}}} & \frac{\partial}{\partial t}\frac{\partial \boldsymbol{L}}{\partial \dot{\boldsymbol{\theta}}} \end{array} \right] \tag{7.39}$$

Substracting Equation (7.38) with Equation (7.39) yields Equation (7.33). Now the external forces acting on the system in SSP are found. As this is given in the Cartesian coordinate system it corresponds to the linear force, $\boldsymbol{f}$, the robot needs to be influences by to move the CoM's for each link in the desired motion. But as the external forces acting on the biped robot is the torque actuated by the servo motor, given in the joint space, a transformation from linear forces in Cartesian coordinates to torques in joint coordinates has to be performed. For this mapping the Jacobian can be used and the actuator torque can be found as [Wisniewski, 2006]:

$$\boldsymbol{Q} = \boldsymbol{\tau}_\theta = \left(\frac{d\boldsymbol{q}}{d\boldsymbol{\theta}}\right)^{\mathrm{T}} \boldsymbol{f} \tag{7.40}$$

where $\boldsymbol{q}$ is the Cartesian coordinates and $\boldsymbol{\theta}$ is the joint coordinates. $\left(\frac{d\boldsymbol{q}}{d\boldsymbol{\theta}}\right)$ is the Jacobian of $\boldsymbol{q}$ denoted $\mathrm{J}(\boldsymbol{q})$, and describes the transformation from Cartesian coordinates to joint coordinates given by the calculations described in Section 7.3.3 on page 77:

$$\frac{d\mathbf{q}}{d\boldsymbol{\theta}} = \mathrm{J}(\mathbf{q}) = \left[\begin{array}{ccc} \frac{\partial q_1}{\partial \theta_1} & \cdots & \frac{\partial q_1}{\partial \theta_{21}} \\ \vdots & \ddots & \vdots \\ \frac{\partial q_{87}}{\partial \theta_1} & \cdots & \frac{\partial q_{87}}{\partial \theta_{21}} \end{array} \right] \tag{7.41}$$

$\mathbf{f}$ is the Cartesian force field and $\boldsymbol{Q}$ is the joint space force field. This yields the model for the biped robot in the SSP-R phase. To construct the model for the SSP-L phase, all the previous steps need to be repeated with the left foot as base frame. This has been performed and implemented for the final model but will not be presented here.

7.4.3 Double Support Phase Modeling

As explained in the introduction to Section 7.4, obtaining a model for the torque given a motion of the biped robot system in DSP, has not been described in the searched

literature. But it has, however, been described for robot manipulator, where there is two or more arms cooperating to move a object in space. This is also known as the Gough-Stewart platform and has been covered throughout the literature [Zheng and Luh, 1988] [Lee et al., 2003]. But the principles used to model the Gough-Stewart platform does not directly apply to biped robots, as the Gough-Stewart platform at all times has the same base frame. But the legs of a biped robot can change their position from phase to phase, as the robot moves in space, and thereby not having a fixed base frame.

Instead a new approach is suggested, which will be presented here. The basic idea about the approach is to realize that when i DSP the biped robot is partly in SSP-R and partly in SSP-L. This means that the actuated torque in the DSP can be calculated as in 7.42:

$$\boldsymbol{\tau}_{\text{DSP}} = w\boldsymbol{\tau}_{\text{L}} + (1-w)\boldsymbol{\tau}_{\text{R}} \tag{7.42}$$

Where $\boldsymbol{\tau}_{\text{L}}$ is the torque calculated in the DSP seen from the left foot, $\boldsymbol{\tau}_{\text{R}}$ is the torque calculated in the DSP seen from the right foot and w is some constant. When only in SSP the sum of the linear forces acting on each link should equal the ground reaction force, acting on the sole of the supporting foot. The same is the case in DSP, except that the sum of linear forces should equal the sum of the ground reaction forces working on both supporting feet. This can be written as:

$$\sum_{i=1}^{22} f_i = f_{\text{N-L}} + f_{\text{N-R}} \tag{7.43}$$

This is also illustrated in Figure 7.16(a), where the black arrows denote the gravity force acting on each link, the red arrow is the ground reaction force working on the right foot and the blue arrow is the ground reaction force working on the left foot.

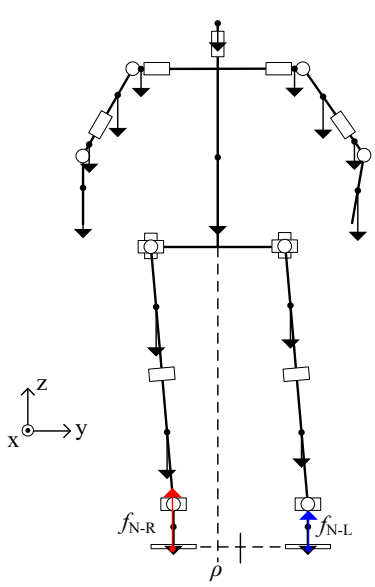
Now to calculate the linear force acting on the biped robot in DSP seen from the left foot the ground reaction force from the right foot has to be considered. As the ground reaction force acts on the foot, it has to be added to that link and thereby the equation for the DSP, seen from the left foot, becomes:

$$\boldsymbol{f}_{\text{DSP-L}} = \boldsymbol{f}_{\text{SSP-L}} + \boldsymbol{\zeta}_{\text{L}}(1-w)m_{\text{tot}}g \tag{7.44}$$

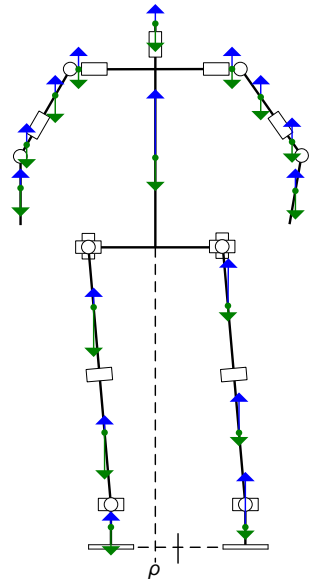
Where w is come constant, m_{tot} is the total mass of the robot, g is the force of gravity and $\boldsymbol{\zeta}_{\text{L}}$ is a vector with zeros in all positions except at one, being position 45, which is the vertical component of the linear force acting on the right foot. The same procedure is used for the right side. By inserting Equation (7.40) into Equation (7.42) and substituting $\boldsymbol{f}$ with Equation 7.44, Equation (7.42) can be written as:

$$\boldsymbol{\tau}_{\text{DSP}} = w\left(\boldsymbol{J}_{\text{L}}\right)^{\text{T}}\left(\boldsymbol{f}_{\text{SSP-L}} + \boldsymbol{\zeta}_{\text{L}}(1-w)m_{\text{tot}}g\right) + (1-w)\left(\boldsymbol{J}_{\text{R}}\right)^{\text{T}}\left(\boldsymbol{f}_{\text{SSP-R}} + \boldsymbol{\zeta}_{\text{R}}m_{\text{tot}}gw\right) \tag{7.45}$$

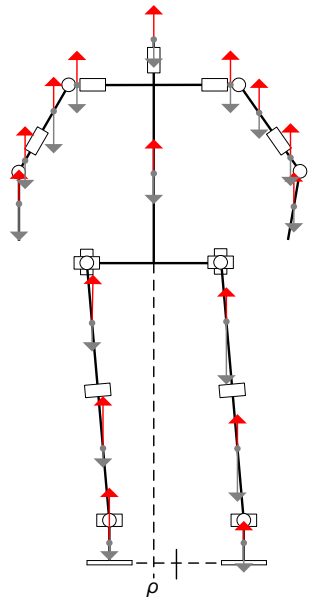
Where $\boldsymbol{J}_{\text{R}}$ is the Jacobian with the right foot as base frame and $\boldsymbol{\zeta}_{\text{R}}$ is a vector with zeros in all positions except a one at position 57 which is the vertical component of the linear force acting on the left foot. When the Jacobian is multiplied by the $\boldsymbol{f}_{\text{DSP-R}}$, the



(a) Forces in the DSP



(b) Distribution of forces in DSP seen from the right foot



(c) Distribution of forces in DSP seen from the left foot

Figure 7.16: *Forces acting on the biped robot when in DSP.* ***a)*** *Red arrow is the normal force acting on the right foot. Blue arrow is the normal force acting on the left foot. Black arrows is the gravity force acting on each link CoM.* ***b)*** *Green is the forces from gravity acting on the right foot. Blue is the normal force acting on the right foot and which has been distributed to the rest of the links.* ***c)*** *Gray is the forces from the gravity acting on the left foot. Red is the normal force acting on the left foot and which has been distributed to the rest of the links.*

result is that the linear force, which has been added on the right foot, is distributed to all of the joints on the robot. This situation is depicted in Figure 7.16(b) seen from the right foot, where the green arrows are the forces from the gravity and the blue arrows are the distributed force which originates from the ground reaction force on the right foot. Figure 7.16(c) is seen from the left foot, where the gray arrows are the forces from gravity and the red arrows are the distributed ground reaction force from the left foot.

The constant w gives an indication of where in the DSP the robot is currently located. In Figure 7.16 it is seen that the biped robot is more in SSP-R than in SSP-L and therefore the ground reaction force from the right foot is largest. This is indicated by making w small. The explanation of how these factors are found, will be done in Section 7.5, since this is a part of the phase estimation. The purpose of the phase estimator is to estimate if the system is in SSP-L, SSP-R or DSP, and if in DSP, estimate the weight distribution on each foot.

To further illustrate the derivation of the dynamic model, a motivating example for a four DoF robot given in Appendix E on page 195.

7.4.4 Verification of the Dynamical Model

In order to verify the dynamical model, two test have been performed. The complete description of these tests and the gained results can be seen in Appendix F in Section F.6 on page 221. The output of the tests were measurements from the strain gauges, which were compared with the output of the model. The results were not in complete accordance, and due to an offset error it was not possible to calculate an error percentage of this test. One of the result from the tests can be seen from Figure 7.17, where the mentioned offset is clearly seen.

Looking at the results, it is however clear, that there is consistency to a high degree between the model and the real system. The most outspoken difference is as mentioned the offset, but with a system like this one, a backlash making the robot tilt i.e. 5 degree backward, would make huge rearrangement in pressure distribution. It is seen from Figure 7.17 that the robot in the real system is tilted further backwards than the robot is in the model, which is caused by backlash. Another reason for the offset error is that in the real system the strain plates located in the feet consists of spring steel, which makes the real system bend more than the model. Since it can be seen that the model captures the trends of the system, and only the offsets are wrong, it is valued that the model is correct.

7.5 Phase Estimator

As explained in Section 7.3 it is necessary to know which leg that is the supporting leg, if the kinematic model are to give the correct output. Besides from knowing which leg is the supporting one, it is further necessary to know how much weight the biped robot distribute on each foot. As explained in Section 7.4 this weight as used to calculate the ground reaction force on each foot in DSP.

The model of the biped robot can be classified as a hybrid system, since there exist an interaction between the continuous states and a set of discrete events. To find

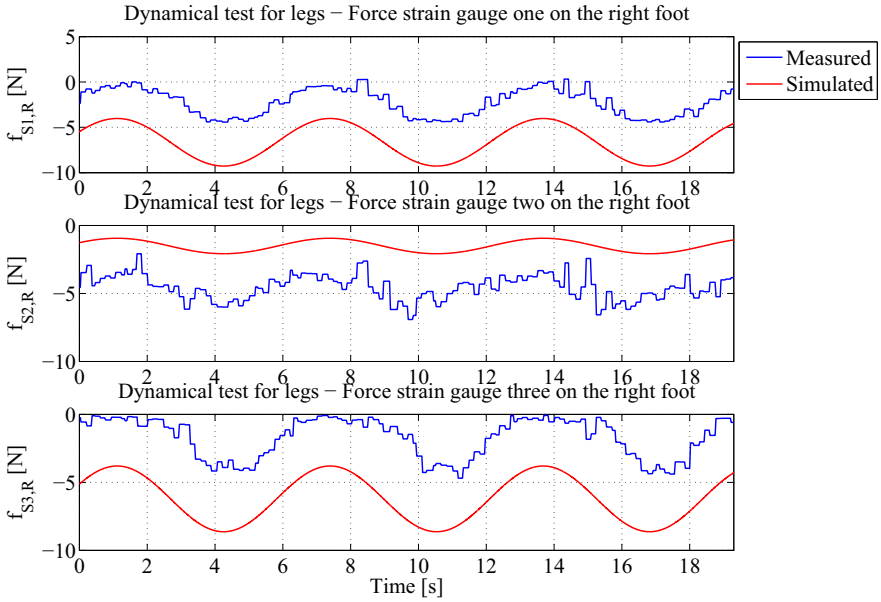


Figure 7.17: *Output of the strain gauges on the right foot for the leg test of the dynamical model.*

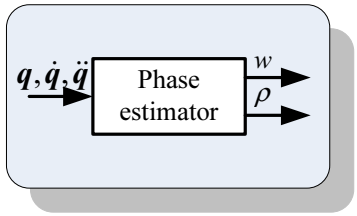
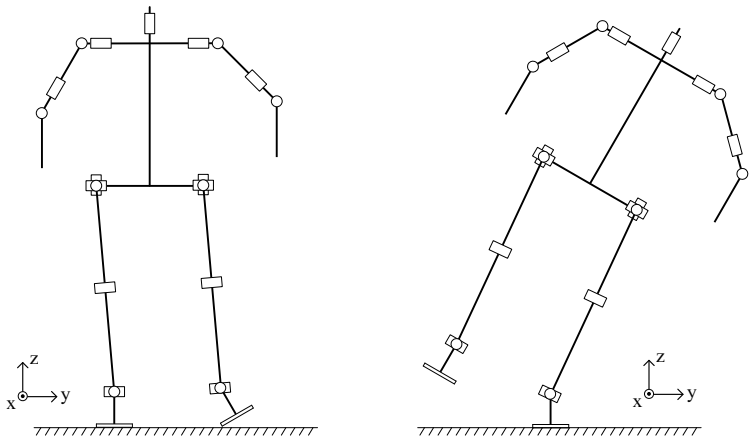


Figure 7.18: *The phase estimator calculates the current state of the system, ρ, and the weight distribution on the feet, w.*

these two values a phase estimator is used. The phase estimator is illustrated in Figure 7.18, where it can be seen that it takes the generalized coordinate vector as an input, and returns the current state of the system, ρ, and the weight distribution on the feet, w.

The necessity of the phase estimator can easily be shown by the example in Figure 7.19, where the biped robot is shown in two completely different situations. But in the two cases the exact same input vector, $\boldsymbol{\theta}$, is given. This illustrates that from the input solely it is not possible to determine the state of the system.



(a) SSP-R (b) SSP-L

Figure 7.19: *The biped robot seen in SSP-R and SSP-L. In both cases the the same input vector, $\boldsymbol{\theta}$, is given.*

7.5.1 The Hybrid Model

The dynamics of a biped robot can be divided into three phases, only two blocks are dependent on the phase, the dynamical model and the kinematic model. The dynamical model is found using Lagrangian formulation, from which the force vector, $\boldsymbol{f}$, is found. The torque on the motor shaft is calculated as follows:

$$\boldsymbol{\tau} = \begin{cases} \boldsymbol{J}_1^{\mathrm{T}} \boldsymbol{f}_1 & \text{if} \quad \mathcal{P} = \rho_1 \\ (1-w)\boldsymbol{J}_1^{\mathrm{T}} \boldsymbol{f}_1 + w\boldsymbol{J}_2^{\mathrm{T}} \boldsymbol{f}_2 & \text{if} \quad \mathcal{P} = \rho_2 \\ \boldsymbol{J}_2^{\mathrm{T}} \boldsymbol{f}_2 & \text{if} \quad \mathcal{P} = \rho_3 \end{cases} \tag{7.46}$$

where $\boldsymbol{f}_1 = \boldsymbol{f}_{\text{SSP-L}} + \boldsymbol{\zeta}_{\text{L}} (1-w)\, m_{\text{tot}} g$ and $\boldsymbol{f}_2 = \boldsymbol{f}_{\text{SSP-R}} + \boldsymbol{\zeta}_{\text{R}} m_{\text{tot}} g w$, $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ are the Jacobians of SSP-R and SSP-L (size 21×87), $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are the force vectors of SSP-R and SSP-L (size 1×87), and w is a weight factor denoting how much the biped robot is in SSP-L. Where the following holds for w:

$$\begin{aligned} \text{DSP:} \quad & 0 < w < 1 \\ \text{SSP-R:} \quad & w = 0 \\ \text{SSP-L:} \quad & w = 1 \end{aligned} \tag{7.47}$$

$\mathcal{P}$ and ρ_i links to the current state of the system, and will be described more thorough in the follow, where the whole system is formulated as a hybrid automaton, which is the common way to express a hybrid system.

Hybrid Automaton

In [Bak and Izadi-Zamanabadi, 2006, pp. 11] a hybrid automaton, H, is given as:

$$H = (\mathcal{P}, \boldsymbol{X}, \text{Init}, f, \text{Dom}, E, G, R)$$

Note that in [Bak and Izadi-Zamanabadi, 2006] the phase of the system, $\mathcal{P}$, is denoted with $\boldsymbol{Q}$, but since $\boldsymbol{q}$ is the generalized coordinate vector, in this project, the phase is denoted $\mathcal{P}$.

In this project the elements of H is given as follows:

$\mathcal{P}$ is the discrete state/phase of the system.

$$\mathcal{P} = \{\rho_1, \rho_2, \rho_3\} = \{\text{SSP-R}, \text{DSP}, \text{SSP-L}\}$$

$\boldsymbol{X}$ is the continuous variables, is this case the joint angles.

$$\boldsymbol{X} = \{\theta_1, \theta_2, ..., \theta_{21}\} \quad , \quad \boldsymbol{X} \in \Re^{21}$$

Init is the initial conditions, the biped robot always starts up in zero position, and DSP.

$$\textbf{Init} = \{\theta_1 = 0, \theta_2 = 0, ..., \theta_{21} = 0, \mathcal{P} = \text{DSP}\}$$

$\boldsymbol{f}$ is a vector field.

$$\begin{aligned} \boldsymbol{f}(\rho_1, \boldsymbol{X}) &= \boldsymbol{f}_{\text{SSP-R}}(\boldsymbol{X}) \\ \boldsymbol{f}(\rho_2, \boldsymbol{X}) &= \boldsymbol{f}_{\text{DSP}}(\boldsymbol{X}) \\ \boldsymbol{f}(\rho_3, \boldsymbol{X}) &= \boldsymbol{f}_{\text{SSP-L}}(\boldsymbol{X}) \end{aligned}$$

Dom is the domain, assigned to each discrete state.

$$\begin{aligned} \text{dom}(\rho_1) &= \{x \in \boldsymbol{X} : z_{\text{left,foot}}(\boldsymbol{X}) > z_{\text{right,foot}}(\boldsymbol{X})\} \\ \text{dom}(\rho_2) &= \{x \in \boldsymbol{X} : z_{\text{left,foot}}(\boldsymbol{X}) = z_{\text{right,foot}}(\boldsymbol{X})\} \\ \text{dom}(\rho_3) &= \{x \in \boldsymbol{X} : z_{\text{left,foot}}(\boldsymbol{X}) < z_{\text{right,foot}}(\boldsymbol{X})\} \end{aligned}$$

$\boldsymbol{E}$ is a collection of discrete transitions.

$$\boldsymbol{E} = \{(\rho_1, \rho_2), (\rho_2, \rho_1), (\rho_3, \rho_2), (\rho_2, \rho_3)\}$$

$\boldsymbol{G}$ are the guards, assigned to each discrete transition.

$$\begin{aligned} G(\rho_1, \rho_2) &= \{x \in \boldsymbol{X} : z_{\text{left,foot}}(\boldsymbol{X}) = z_{\text{right,foot}}(\boldsymbol{X})\} \\ G(\rho_2, \rho_1) &= \{x \in \boldsymbol{X} : z_{\text{left,foot}}(\boldsymbol{X}) > z_{\text{right,foot}}(\boldsymbol{X})\} \\ G(\rho_3, \rho_2) &= \{x \in \boldsymbol{X} : z_{\text{left,foot}}(\boldsymbol{X}) = z_{\text{right,foot}}(\boldsymbol{X})\} \\ G(\rho_2, \rho_3) &= \{x \in \boldsymbol{X} : z_{\text{left,foot}}(\boldsymbol{X}) < z_{\text{right,foot}}(\boldsymbol{X})\} \end{aligned}$$

$\boldsymbol{R}$ are the reset relations.

$$R(\rho_1, \rho_2, x) = R(\rho_2, \rho_1, x) = R(\rho_3, \rho_2, x) = R(\rho_2, \rho_3, x) = \{x\}$$

7.5.2 Finding the Phase of the System

As mentioned, the dynamics of a biped robot can be divided into three phases, $\mathcal{P}$. The switch between the different phases is called a transition, $\boldsymbol{E}$, and are governed by the guards, $\boldsymbol{G}$.

Transitions are evoked when the height of the feet are changing. When the height is equal both feet are on the ground and the biped must be in DSP. In SSP the foot that is on the ground determines which phase the biped robot is in. These constraints are therefore used as guards to determine the transitions between the phases, and the design of the phase location estimation will depend on these transitions.

In practice it is not possible to implement the phase estimator precisely as described above, since the feet will never have exactly the same height, and the system will never reach DSP. It is therefore chosen to insert a small limit, δ_{DSP}, and the guards are then given as:

$$\begin{aligned} G(\rho_1, \rho_2) &= \{x \in \boldsymbol{X} : z_{\text{left,foot}}(\boldsymbol{X}) < z_{\text{right,foot}}(\boldsymbol{X}) + \delta_{\text{DSP}}\} \\ G(\rho_2, \rho_1) &= \{x \in \boldsymbol{X} : z_{\text{left,foot}}(\boldsymbol{X}) > z_{\text{right,foot}}(\boldsymbol{X}) + \delta_{\text{DSP}}\} \\ G(\rho_3, \rho_2) &= \{x \in \boldsymbol{X} : z_{\text{left,foot}}(\boldsymbol{X}) > z_{\text{right,foot}}(\boldsymbol{X}) - \delta_{\text{DSP}}\} \\ G(\rho_2, \rho_3) &= \{x \in \boldsymbol{X} : z_{\text{left,foot}}(\boldsymbol{X}) < z_{\text{right,foot}}(\boldsymbol{X}) - \delta_{\text{DSP}}\} \end{aligned}$$

The value of this limit is set to $\delta_{\text{DSP}} = 5\,\text{mm}$.

7.5.3 Calculation of the Weight Distribution

The phase estimator should also return the weight distribution on the feet w. In the three phases w is given in Equation (7.47). As seen w is easily found in SSP, where all the weight is on one foot. In DSP it is however a little more difficult. In DSP w is calculated from the ZMP, which was introduced in Section 2.3 on page 7 and given as:

$$x_{\text{ZMP}} = \frac{\sum_{i=1}^{22} m_i \left((\ddot{z}_i - g_z) x_i \ddot{x}_i z_i \right)}{\sum_{i=1}^{22} m_i \left(\ddot{z}_i - g_z \right)} \tag{7.48}$$

The ZMP in the frontal plane is given by:

$$y_{\text{ZMP}} = \frac{\sum_{i=1}^{22} m_i \left((\ddot{z}_i - g_z) y_i - \ddot{y}_i z_i \right)}{\sum_{i=1}^{22} m_i \left(\ddot{z}_i - g_z \right)} \tag{7.49}$$

The problem with the calculation of the ZMP from Equation (7.48) and Equation (7.49) is that if the denominator equals zero the result will go to infinity. Instead a new way of calculation the ZMP is given in Equation (7.50) and Equation (7.51), where the denominator consists of a constants factor. This ZMP calculation is given in [You et al., 2004]:

$$x_{\text{ZMP,m}} = \frac{\sum_{i=1}^{22} m_i \left(g_z x_i - \ddot{x}_i z_i\right)}{\sum_{i=1}^{22} m_i g_z} \tag{7.50}$$

Where the added subscript ",m" is short for "modified". For the y-direction, the ZMP is calculated as follows:

$$y_{\text{ZMP,m}} = \frac{\sum_{i=1}^{22} m_i \left(g_z y_i - \ddot{y}_i z_i\right)}{\sum_{i=1}^{22} m_i g_z} \tag{7.51}$$

Now to calculate w the ZMP and the distance between the feet are used. This is illustrated in Figure 7.20:

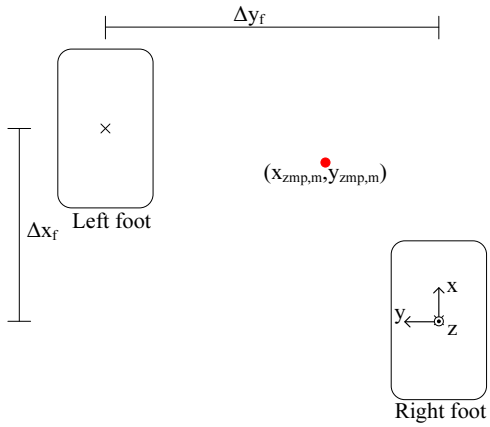


Figure 7.20: *The ZMP and distances Δx_f and Δy_f used for calculation of the weight distribution, w.*

Now w can be calculated as:

$$w = \frac{x_{\text{ZMP,m}}}{\Delta x_{\text{f}} + \Delta y_{\text{f}}} + \frac{y_{\text{ZMP,m}}}{\Delta x_{\text{f}} + \Delta y_{\text{f}}} \tag{7.52}$$

where only ZMP values between the feet are used, thus:

$$x_{\text{ZMP,m}} \in [0; \Delta x_{\text{f}}] \tag{7.53}$$

$$y_{\text{ZMP,m}} \in [0; \Delta y_{\text{f}}] \tag{7.54}$$

w now serves as a weight factors that indicates where in the DSP, the dynamical system currently is located in. The closer the ZMP is to the left foot, the more the model is in SSP-L and the model for this phase gets the most influence. Now the collected torque in the DSP can be calculated using Equation (7.45).

7.6 Foot Model

In Section 4.5.3 on page 30 the foot is designed, and it can be seen that it consists of an upper part, which is placed on three bendable levers, called strain plates. On the three plates a strain gauge is placed on each side. These three levers are mounted on the lower part of the foot which consist of the sole.

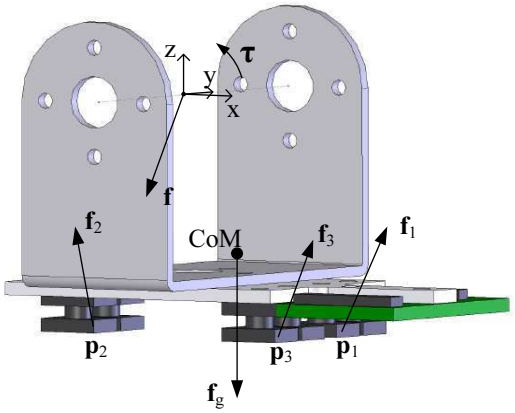


Figure 7.21: *Isometric view of the foot. Showing the forces acting on the upper part of the foot.*

In Figure 7.21 the upper part of the right foot is illustrated. The model of the foot will yield the three forces, $\boldsymbol{f}_{\text{foot}} = [f_{1_z}, f_{2_z}, f_{3_z}]^{\text{T}}$ (the vertical components of $\boldsymbol{f}_1$, $\boldsymbol{f}_2$ and $\boldsymbol{f}_3$), acting on the connection point of each of these plates. The measurement model of the strain gauge sensors, described in Appendix D on page 191, also yields this value. The foot model describes the three forces imposed on the strain gauges plates, given the torque in the ankle. This is illustrated in Figure 7.22.

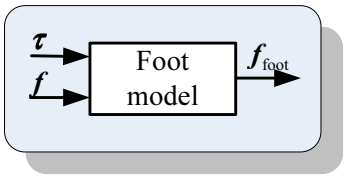


Figure 7.22: *The foot model is a transformation from the torques and forces in the ankle joint to the three forces measured by the strain gauges.*

The developed model, described in this section, builds on principals presented in [Ito and Kawasaki, 2005] and [Ito et al., 2004], where similar models are derived, though only for two dimensions. All general equations and laws of physics are found in [Serway and Beichner, 2000]. Also the approach taken in [Christensen et al., 2006] is considered, where a similar model for three dimensions is developed.

7.6.1 Forces Acting on the Foot

In order to make an expression of the ground reaction force, it is necessary to identify all the forces acting on the foot. In Figure 7.21 it can be seen that the foot is influenced by the gravity force, $\boldsymbol{f}_{\mathrm{g}}$, acting on the CoM of the upper part of the foot, a linear force, $\boldsymbol{f}$, imposed by the next link, a torque, $\boldsymbol{\tau}$, imposed by the next link and the ground reaction forces, $\boldsymbol{f}_1$, $\boldsymbol{f}_2$ and $\boldsymbol{f}_3$. Since it is only wanted to find the vertical components of the ground reaction force it is only necessary to consider the vertical component of $\boldsymbol{f}$, called f_{z}, and the two components of the torque situated on the x-axis and the y-axis, called τ_{x} and τ_{y}, see Figure 7.25 and Figure 7.26.

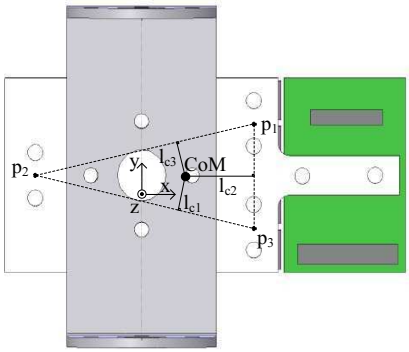


Figure 7.23: *Horizontal view of the foot seen from above. Showing placement of CoM.*

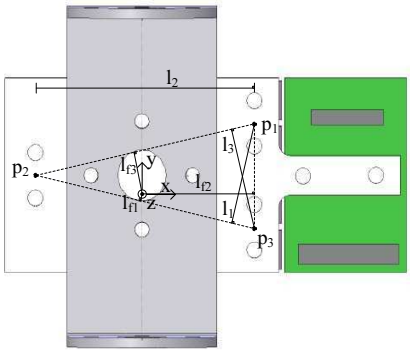


Figure 7.24: *Horizontal view of the foot seen from above. Showing placement of frame* $\{1\}$.

Gravity, $\boldsymbol{f}_{\mathrm{g}}$

First the effect from the gravity is considered. The total effect from the gravity is $m\mathrm{g}$, where m is the mass of the upper part of the foot. The mass m should now be "distributed" among the three contact points, $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$. Figure 7.23 is an illustration of the foot in the horizontal plane, illustrating the placement of CoM.

The part of the mass m introduce the following force at $\boldsymbol{p}_i$:

$$f_{\mathrm{g}i} = c_{\mathrm{m}} \frac{l_{\mathrm{c}i}}{l_i} m\mathrm{g} \tag{7.55}$$

where c_{m} is a constant ensuring that the resulting mass on the three contact points equals m, $\mathrm{l}_{\mathrm{c}i}$ is the distance from the CoM to the sides of the triangle and l_i is the heights of the triangle, illustrated in Figure 7.24. c_{m} is given as:

$$c_{\mathrm{m}} = \left(\sum_{i=1}^{3} \frac{l_{\mathrm{c}i}}{l_i} \right)^{-1} \tag{7.56}$$

Linear force, f_{z}

Figure 7.24 shows the placement of frame $\{1\}$, which is the point where the force f_{z} is applied. The same considerations as for the gravity, can be done for the force f_{z}. This force has the following influence on the contact point p_i:

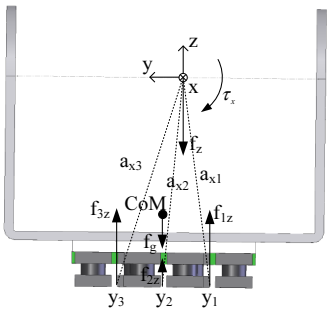


Figure 7.25: *Frontal view of the foot seen from the back.*

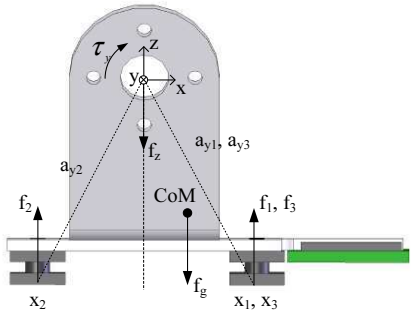


Figure 7.26: *Sagittal view of the foot seen from the right.*

$$f_{zi} = c_f \frac{l_{fi}}{l_i} f_z \tag{7.57}$$

where c_f is given as:

$$c_f = \left(\sum_{i=1}^{3} \frac{l_{fi}}{l_i} \right)^{-1} \tag{7.58}$$

Torque, τ_y

A torque, τ_y, applied at joint 1, will introduce a force, $\boldsymbol{f}_{\tau_y i}$, at contact point p_i. The torque τ_y, can be divided into three components, τ_{y1}, τ_{y2} and τ_{y3}. Where τ_{yi} is related to the force at p_i as:

$$\tau_{yi} = \boldsymbol{f}_{\tau_y i} a_{yi} \tag{7.59}$$

where a_{yi} is the moment arm. Following relation must be fulfilled:

$$\tau_y = \tau_{y1} + \tau_{y2} + \tau_{y3} \tag{7.60}$$

The relationship between the torques is given as:

$$\tau_{yi} = \frac{a_{yj}}{a_{yi}} \tau_{yj} \tag{7.61}$$

From Equation (7.61), Equation (7.60) can be re-written as:

$$\tau_{yi} = \tau_y \sum_j \frac{a_{yj}}{a_{yi}} \tag{7.62}$$

Having divided the torque into these three components, the force at contact point p_i can now be found from Equation (7.59). It is however, only the vertical part of $\boldsymbol{f}_{\tau_y i}$, called $f_{\tau_y z}$ that is of interest. This force can be found as:

$$\begin{aligned} f_{\tau_y z} &= \sin\theta_{yi} \boldsymbol{f}_{\tau_y i} = \frac{x_i}{a_{yi}} \boldsymbol{f}_{\tau_y i} \\ &= \frac{x_i}{a_{yi}^2 \sum_j \frac{a_{yj}}{a_{yi}}} \tau_y \end{aligned} \tag{7.63}$$

Note that p_1 and p_3 in the sagittal plane are placed equal, thus $f_{\tau_y 1} = f_{\tau_y 3}$.

Torque, τ_x

With reference to Figure 7.25, the same calculations can be performed in the frontal plane, and the force at contact point p_i is given as:

$$f_{\tau_x i} = -\frac{y_i}{a_{xi}^2 \sum_j \frac{a_{xj}}{a_{xi}}} \tau_x \tag{7.64}$$

7.6.2 Summing the Forces

Gathering the contribution from the gravity force, the linear force, and the two torques, given in Equation (7.55), (7.57), (7.63) and (7.64) respectively, the resulting force applied at the three contact points can be found as:

$$f_i = f_{gi} + f_{zi} + f_{\tau_y i} + f_{\tau_x i} \tag{7.65}$$

which can be written as:

$$\boldsymbol{f}_{\text{foot}} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} 1 \\ f_z(t) \\ \tau_y(t) \\ \tau_x(t) \end{bmatrix} \tag{7.66}$$

where $\boldsymbol{A}$ is found, by inserting all know values, to be:

$$\boldsymbol{A} = \begin{bmatrix} c_m \frac{l_{c1}}{l_1} mg & c_f \frac{l_{f1}}{l_1} & \frac{x_1}{a_{y1}^2 + a_{y1}a_{y2} + a_{y1}a_{y3}} & -\frac{y_1}{a_{x1}^2 + a_{x1}a_{x2} + a_{x1}a_{x3}} \\ c_m \frac{l_{c2}}{l_2} mg & c_f \frac{l_{f2}}{l_2} & \frac{x_2}{a_{y1}a_{y2} + a_{y2}^2 + a_{y2}a_{y3}} & -\frac{y_2}{a_{x1}a_{x2} + a_{x2}^2 + a_{x2}a_{x3}} \\ c_m \frac{l_{c3}}{l_3} mg & c_f \frac{l_{f3}}{l_3} & \frac{x_1}{a_{y1}a_{y3} + a_{y2}a_{y3} + a_{y3}^2} & -\frac{y_1}{a_{x1}a_{x3} + a_{x2}a_{x3} + a_{x3}^2} \end{bmatrix} \tag{7.67}$$

Note that the foot model described in this section is for the right foot, however the same approach holds for the left foot. The foot shown in Figure 7.21 is the right foot, the left foot is exactly the same, only the location of the contact point changes. Thus if all the constants are changed according to the coordinate system placed in the joint 12, the same model can be used for the left foot.

7.6.3 Verification of the Foot Model

In order to verify the model of the foot, one of the feet has been dismounted from the robot, such that known torques and forces could be applied. The complete description of the performed tests and the obtained results is placed in Appendix F in Section F.3 on page 207. The main result is, that the foot model fits the real foot with a average R^2-value of 0.622. The relatively low fit percentage is mostly due to noise in the measurements. A plot of the results can be seen in Figure 7.27, where it can be seen that the model estimates the trend in the output. Therefore it is concluded that the foot model is correct.

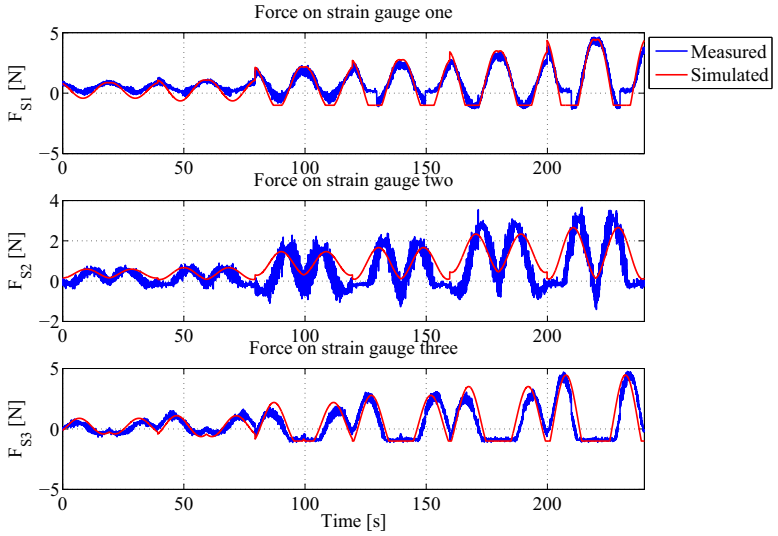


Figure 7.27: *Results from the verification of the foot model. Blue is the measured output of the strain gauges on the real foot. Red is the output of the model. The uppermost figure is output from strain gauge one, middle is output from strain gauge two and lowest is the output from strain gauge three.*

7.7 Head Model

The head model is a transformation from the generalized coordinates, $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$, onto the linear acceleration, $\boldsymbol{a}_{\text{head}}$, and the angular velocity, $\boldsymbol{\omega}_{\text{head}}$, measured by the IMU. The IMU consist of an accelerometer and a gyroscope and is explained in Section 5.2.3 on page 36.

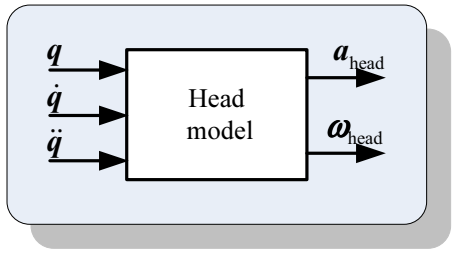


Figure 7.28: *The head model returns the linear acceleration and angular velocity of the head, when given the generalized positions, velocities and accelerations as input.*

The placement of the IMU can be seen in Figure 7.29.

7.7.1 Angular Velocity of the Head

The gyroscope returns the angular velocity around the x- and y-axis of link frame $\{21\}$, illustrated in Figure 7.29. The angular velocity can simply be found from last part of $\dot{\boldsymbol{q}}$ containing the joint angle velocities, $\dot{\boldsymbol{\theta}}$. $\boldsymbol{\omega}_{\text{head}}$ can be found from the following iterative equation:

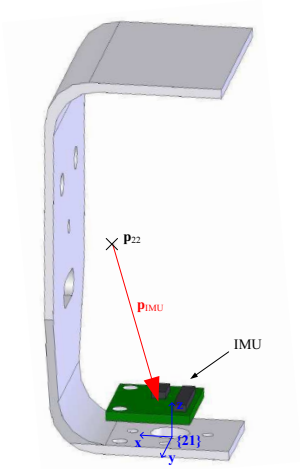


Figure 7.29: *The placement of the IMU. $\boldsymbol{p}_{22}$ is the CoM of the head, frame $\{21\}$ is the link frame attached to the head.*

$$ {}^{i}\boldsymbol{\omega}_i = {}^{i}_{i-1}\boldsymbol{R}\,{}^{i-1}\boldsymbol{\omega}_{i-1} + \boldsymbol{\zeta}_i\dot{\theta}_i \tag{7.68} $$

where $\boldsymbol{\zeta}_i$ is a vector denoting the axis which θ_i rotates about. From Equation (7.68) the angular velocity of the head can be found:

$$ \boldsymbol{\omega}_{\text{head}} = {}^{21}\boldsymbol{\omega}_{21} \tag{7.69} $$

7.7.2 Linear Acceleration of the Head

From $\ddot{\boldsymbol{q}}$ the linear acceleration of the CoM of the head can be found. The head is link 22, thus $\boldsymbol{a}_{22} = [\ddot{x}_{22}, \ddot{y}_{22}, \ddot{z}_{22}]^{\mathrm{T}}$. $\boldsymbol{a}_{22}$ is the actual acceleration of link 22, the accelerometer however also measures the acceleration due to the gravity force. As the model should give the same output as sensors, the contribution from the force of gravity is added:

$$ {}^{G}\boldsymbol{a}_{22} = \begin{bmatrix} \ddot{x}_{22} \\ \ddot{y}_{22} \\ \ddot{z}_{22} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{7.70} $$

As indicated by the leading superscript, G, the acceleration is given in the global frame, the accelerometer however measures the acceleration in its own frame, which is aligned with frame $\{21\}$. Therefore ${}^{G}\boldsymbol{a}_{22}$ has to be transformed to this frame, which is done by premultiplying with the rotation matrix defined in Equation (7.14) on page 77:

$$ {}^{21}\boldsymbol{a}_{22} = {}^{21}_{0}\boldsymbol{R}\,{}^{G}\boldsymbol{a}_{22} \tag{7.71} $$

Equation (7.71) gives the acceleration of the CoM of the head, this should instead be the acceleration of the IMU. Having the linear acceleration of one point in a rigid body, it can be moved to another point of the body with Equation (7.72) [Craig, 1989]:

$$ \boldsymbol{a}_{\text{head}} = \dot{\boldsymbol{\omega}}_{21} \times \boldsymbol{p}_{\text{IMU}} + \boldsymbol{\omega}_{\text{head}} \times (\boldsymbol{\omega}_{\text{head}} \times \boldsymbol{p}_{\text{IMU}}) + {}^{21}\boldsymbol{a}_{22} \tag{7.72} $$

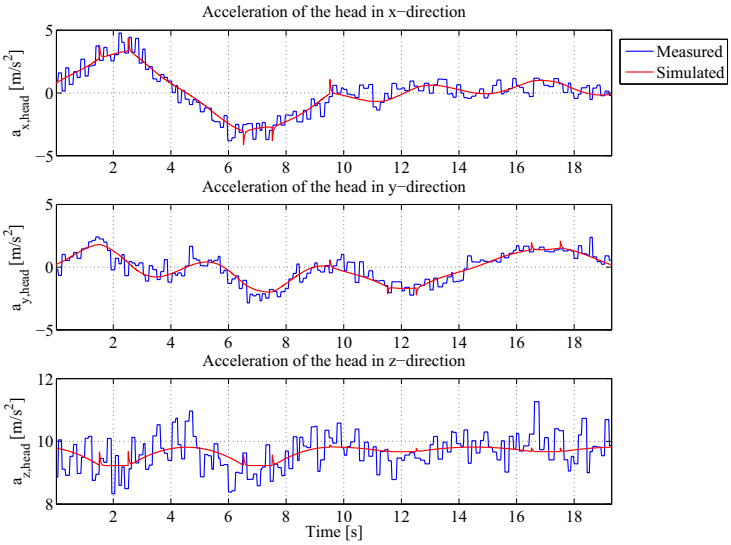


Figure 7.30: *The measured and simulated acceleration of the head.*

where $\dot{\boldsymbol{\omega}}_{21}$ is angular acceleration of the head, given in Equation (7.72), $\boldsymbol{\omega}_{\text{head}}$ is the angular velocity given in Equation (7.69) and $\boldsymbol{p}_{\text{IMU}}$ is the position of the IMU, given with respect to $\boldsymbol{p}_{22}$. $\boldsymbol{p}_{\text{IMU}}$ is illustrated in Figure 7.29. From [Craig, 1989] the angular acceleration is defined as:

$$ {}^{i}\dot{\boldsymbol{\omega}}_i = {}^{i}_{i-1}\boldsymbol{R}\,{}^{i-1}\dot{\boldsymbol{\omega}}_{i-1} + {}^{i}_{i-1}\boldsymbol{R}\,{}^{i-1}\boldsymbol{\omega}_{i-1} \times \boldsymbol{\zeta}_i\dot{\theta}_i + \boldsymbol{\zeta}_i\ddot{\theta}_i \tag{7.73} $$

7.7.3 Verification of the Head Model

The verify that the head model is correct, a test has been performed, where the head model and the real system are given the same input, and afterward the outputs are compared. A description of the complete test of the head model and a review of the obtained results can be found in Appendix F in Section F.5 on page 218.

The main result of the head test is that the acceleration output of the model fits the measured out with an average R^2-value of 0.631. In Figure 7.30 the acceleration output can be seen of both the model and the real system. And it can clearly be seen that the model fits the measurements, which however contains a large level of noise thus the relatively low fit percentage.

For the angular velocity the results were not that good, which is caused by the fact that it is impossible to apply an input to the system where a sufficient angular velocity can be obtained, and still maintain stability of the robot. Since the angular velocity and the acceleration are calculated from the same rotation matrices, it is concluded that both output are correct.

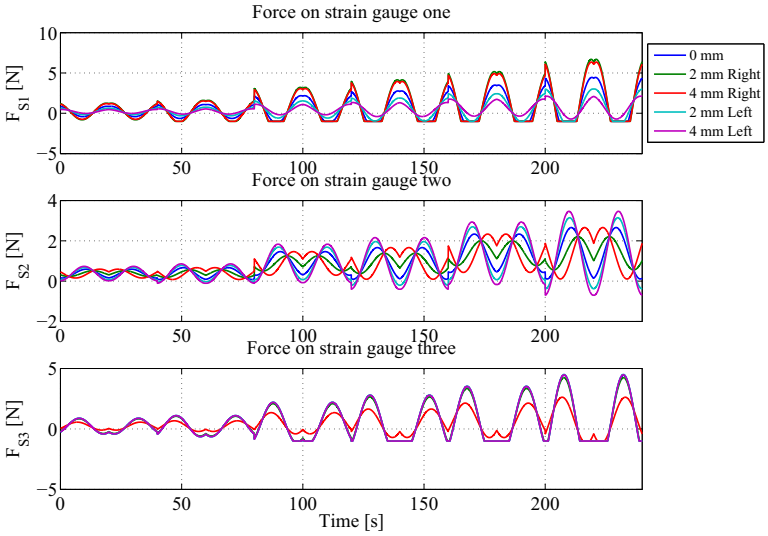


Figure 7.31: *The coordinate system located in the foot was moved to the left and right in the y-axis. It is seen that his has a large impact on the model, specially on strain gauge two which is located right above the origin.*

7.8 Partial Conclusion to Modeling

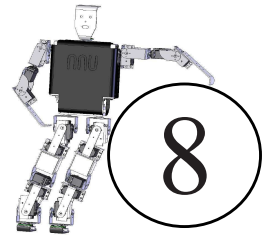
In this chapter the model of the robot was developed. The model was designed to be an input to output description of the real system, enabling the possibility to used the model for simulation purpose. Both kinetics and dynamics were included in the model. The model was constructed for both SSP and DSP which thereby made the model a hybrid system. The dynamics during the DSP was calculated using a novel solution, combining the two SSP in one. A phase estimator was designed and implemented to switch between models.

A verification of the complete model is conducted, where the model and the robot are given the same input and afterward the output is compared. The complete description of the performed test can be seen in Appendix F in Section F.7. The main result of the test was, that the developed model fits the real system to a certain degree. That is, the same fluctuations changes in signals can be observed, but several of the outputs have an offset error, which was also observed in the verification of the dynamical model. The main reason for the divergens between the model output and the system is the bendable spring steel plates in the feet. Because of these plates the robot starts wobbling, which have a huge impact on the output. Furthermore the backlash in the system makes the robot tilt, which also influences the measurements greatly.

A test was conducted to see how much an error in the location of the origin in the foot, had on the foot model. This is seen from Figure 7.31, where the origin was moved in the y-axis. From this figure it is seen that moving the origin has a large impact on the model, which would be the same in the real system. If the real origin is located different than that of the model, this would lead to a difference in the output of the strain gauges.

Based on the fact that all sub-models have been verified individually and based on the fact that the model clearly estimates the trends of the measure output, it is concluded that the complete model is correct. However a more precise model can be obtained if the plates in the feet and the backlash in the system are included in the model.

INVERSE KINEMATICS



8

Chapter contents

IN SECTION *7.3 on page 71, a kinematic model for the biped robot was derived. This model can be used to calculate the position and orientation of the CoM of all links, given the joint angles. In this chapter, the inverse kinematic model is derived, which can be used to calculate the joint angles, if the orientation and position of the torso, swing foot and the hands are given. Two solutions are given to solve the inverse kinematic problem of the robot; a closed form solution and a numerical solution. The former, which is utilized in this project, is explained in this chapter, whereas the latter, which is not utilized, is explained in Appendix G on page 233.*

In the previous chapter the derivation of the kinematic model was shown. The kinematic model describes the position of all the limbs in cartesian space given the joint angles:

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{\theta}) \tag{8.1}$$

where $\boldsymbol{f}$ is a nonlinear function, that maps $\boldsymbol{x}$ to $\boldsymbol{\theta}$. An equally important model is the inverse kinematic model where the joint angles are found given the desired position and orientation of the limbs in Cartesian space:

$$\boldsymbol{\theta} = \boldsymbol{f}^{-1}\boldsymbol{x}) \tag{8.2}$$

The problem regarding the inverse kinematic model in Equation (8.2), compared to the kinematic model, is that there exists no single solution, but often infinitely many. Within the topic of inverse kinematics, there are two major subjects, namely numerical solutions and closed form solutions. The closed form solution seeks a single solution by writing explicit equations for each joint, using either a geometric or analytic approach. The downside of the closed form solution is that no solutions can be found for kinematic chain containing more than six joints. Even for kinematic chains containing six joints the closed form solution can only be found for special cases [Craig, 1989]. The numerical solution approximates the solution via an iterative procedure, which minimizes a performance function. When using the numerical solution there is no limit on the number of joints the kinematic chain can contain. But the downside is that the numerical solution is slower than the closed form solution, and can therefore be disadvantageous to implement on the robot. To investigate the best solution both the closed form solution and the numerical solution are constructed and compared. In Appendix G on page 233 the numerical solution to the inverse kinematic problem is derived.

Figure 8.1 shows the inputs that the inverse kinematic model needs to map to joint space.

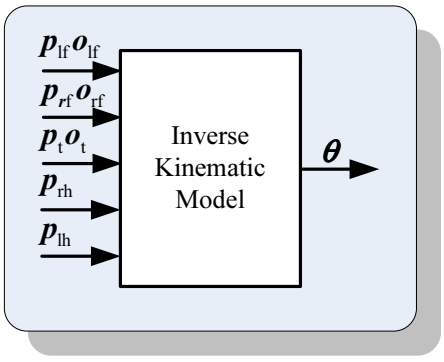


Figure 8.1: *Inputs and output relations of the inverse kinematic model. Inputs are positions and orientation, given in Cartesian space, of the limbs and torso. Output is the joint space coordinates.*

The inputs in Figure 8.1 are as follows:

$\boldsymbol{p}_{\text{lf}}$ The position vector of the left foot

$\boldsymbol{o}_{\text{lf}}$ The orientation vector of the left foot

$\boldsymbol{p}_{\text{rf}}$ The position vector of the right foot

$\boldsymbol{o}_{\text{rf}}$ The orientation vector of the right foot

$\boldsymbol{p}_{\text{t}}$ The position vector of the torso

$\boldsymbol{o}_{\text{t}}$ The orientation vector of the torso

$\boldsymbol{p}_{\text{rh}}$ The position vector of the right hand

$\boldsymbol{p}_{\text{lh}}$ The position vector of the left hand

Using the above inputs it is possible to place and orient the limbs of the robot. Figure 8.2 shows how the above position vectors are placed on the robot with regards to a common frame.

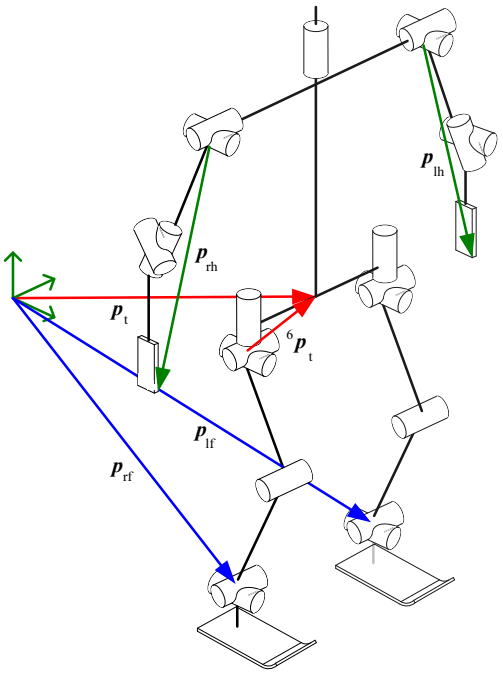


Figure 8.2: *Position vectors in the inverse kinemeatics.*

The following will describe the derivation of the closed form inverse kinematic model.

8.1 Closed Form Solution to the Inverse Kinematic Problem

In this section a closed form solution for the inverse kinematic problem for legs and arms is given.

8.1.1 Inverse Kinematics for the Legs

The placement and orientation of the legs not only determines where the feet are placed but also the posture of the robot. Here, posture refers to the orientation of the torso which, as described in Chapter 3.1 on page 14, is actively controlled in humans. This makes it necessary for the inverse kinematics of the legs to include the orientation of the torso as well as the orientation of the feet.

Pieper's solution [Craig, 1989] is a special case of the closed form solution to kinematic chains having six joints. This can be either prismatic or revolute joints. The solution applies to chains where the revolution axes of the last three links intersect in one point. This is the case for the legs of the biped robot. Here the last three joints, located in the hip, have intersecting axes. This point is called point of intersection (PoI). That Pieper's solution is fulfilled can also be seen from Figure 8.3, showing the kinematics of the robot.

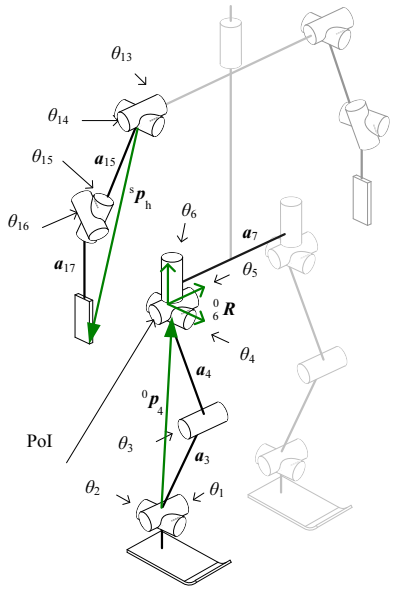


Figure 8.3: *The kinematics of the legs fulfill the requirements to Piepers's solution as the rotation axis of the last three joints intersect in one point, called PoI. Furthermore the target rotation and position of the inverse kinematics of the legs and hands are seen. The target position of the hip is ${}^{0}\boldsymbol{p}_4$ and the orientation is ${}^{0}_{6}\boldsymbol{R}$. The target position of the hand is ${}^{s}\boldsymbol{p}_h$.*

The idea behind Pieper's solution is to use the first three joints, ankle and knee, to position PoI and then the last three joints, the hip, to get the right orientation of the torso. In Figure 8.3 the vector ${}^{0}\boldsymbol{p}_4$ is the position of the PoI and also one of the input to the closed form inverse kinematic model of the leg. The other input to the inverse kinematic model is the rotation from the base frame to the torso, called ${}^{0}_{6}\boldsymbol{R}$. The positioning of PoI is given by:

$$ {}^{0}\boldsymbol{p}_4 = {}^{0}_{1}\boldsymbol{T}^{1}_{2}\boldsymbol{T}^{2}_{3}\boldsymbol{T}^{3}\boldsymbol{p}_4 \tag{8.3} $$

where ${}^{0}_{1}\boldsymbol{T}$ describes the position and orientation of frame $\{1\}$ relative to frame $\{0\}$. ${}^{0}_{1}\boldsymbol{T}$ is a transformation matrix, that maps a point ${}^{1}\boldsymbol{p} \mapsto {}^{0}\boldsymbol{p}$. If the mapping to be performed is ${}^{0}\boldsymbol{p} \mapsto {}^{1}\boldsymbol{p}$ this can be obtained by using ${}^{1}_{0}\boldsymbol{T} = {}^{0}_{1}\boldsymbol{T}^{-1}$. The transformation matrix can be computed as:

$$ {}^{0}_{1}\boldsymbol{T} = \left[\begin{array}{ccc|c} & {}^{0}_{1}\boldsymbol{R} & & {}^{0}\boldsymbol{p}_{1org} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{8.4} $$

where ${}^{0}_{1}\boldsymbol{R}$ is the rotation matrix that describes the orientation of frame $\{1\}$ relative to frame $\{0\}$ and ${}^{0}\boldsymbol{p}_{1org}$ is the position vector, that describes the origin of frame $\{1\}$ relative to $\{0\}$. A more thorough description of the rotation matrices and their properties can be found in Section 7.3 on page 71.

Pieper's solution states that Equation (8.3) can be written as:

$$ {}^{3}\boldsymbol{p}_4 = {}^{0}_{1}\boldsymbol{T}^{1}_{2}\boldsymbol{T} \begin{bmatrix} f_1(\theta_3) \\ f_2(\theta_3) \\ f_3(\theta_3) \\ 1 \end{bmatrix} \tag{8.5} $$

where f_i is a function describing PoI in frame $\{2\}$. It is noticed that the function f_i only contains θ_3 as a variable. Note that:

$$ {}^{0}\boldsymbol{p}_4 = \boldsymbol{a}_4 = \begin{bmatrix} a_{4_x} \\ a_{4_y} \\ a_{4_z} \\ 1 \end{bmatrix} \tag{8.6} $$

With ${}^{2}_{3}\boldsymbol{T}$ and ${}^{3}_{4}\boldsymbol{p}$ given, f_i can be written as:

$$ f_1 = c_3 a_{4_x} + s_3 a_{4_z} - a_{3_x} \tag{8.7} $$
$$ f_2 = 0 \tag{8.8} $$
$$ f_3 = -s_3 a_{4_x} + c_3 a_{4_z} + a_{3_z} \tag{8.9} $$

The magnitude squared of ${}^{0}_{4}\boldsymbol{p}$ is calculated as:

$$ r^2 = f_1^2 + f_2^2 + f_3^2 \tag{8.10} $$

For the type of kinematic composition used for the biped robot, the magnitude squared of ${}^{0}_{4}\boldsymbol{p}$ can also be written as:

$$ \begin{aligned} r^2 = {c_3}^2 {a_{4_x}}^2 - 2c_3 a_{3_x} a_{4_x} + {s_3}^2 {a_{4_z}}^2 - 2s_3 a_{4_z} a_{3_x} + {a_{3_x}}^2 + {s_3}^2 {a_{4_x}}^2 \\ -2s_3 a_{4_x} a_{3_z} + {c_3}^2 {a_{4_z}}^2 + 2c_3 a_{4_z} a_{3_z} + {a_{3_z}}^2 \end{aligned} \tag{8.11} $$

According to Pieper the solution of the magnitude squared of ${}^{0}\boldsymbol{p}_4$ can contain other variables than θ_3. Finding θ_3 from Equation (8.11) would then have been more cumbersome. But because of the construction of the biped robot, the magnitude squared of

${}^0\boldsymbol{p}_4$ is the same as the squared of the f_i's and thereby θ_3 is found. Equation (8.11) can be rewritten to:

$$\underbrace{r^2 - {a_{3_x}}^2 - {a_{4_x}}^2 - {a_{3_z}}^2 - {a_{4_z}}^2}_{C} = c_3 \underbrace{(2a_{4_z}a_{3_z} - 2a_{3_x}a_{4_x})}_{A} + s_3 \underbrace{(-2a_{4_z}a_{3_x} - 2a_{4_x}a_{3_z})}_{B} \tag{8.12}$$

The inverse kinematic model will have redundant solutions, however, because of the joint limitations only one solution for each joint will be correct. The joint limitations are shown in Table L.1 on page 271 in Appendix L. Equation (8.12) can now be solved for θ_3 as:

$$\theta_3 = \text{arctan2}(B, A) \pm \text{arctan2}(\sqrt{A^2 + B^2 - C^2}, C) \tag{8.13}$$

where it is noticed that there are two solutions, corresponding to flipping the knee angle θ_3. Only one of the solutions is correct at any time, and this solution corresponds to the leg bending backwards. The correct solution is determined using Table L.1 on page 271. As θ_3 is now known, Equation (8.3) can be used to solve for θ_2 and θ_1. Writing out Equation (8.3) yields:

$${}^0\boldsymbol{p}_4 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} c_1 \underbrace{(c_3a_{4_x} + s_3a_{4_z} - a_{3_x})}_{B_2} + s_1 \underbrace{c_2\,(-s_3a_{4_x} + c_3a_{4_z} + a_{3_z})}_{A_2} \\ -s_2\,(-s_3a_{4_x} + c_3a_{4_z} + a_{3_z}) \\ -s_1 \underbrace{(c_3a_{4_x} + s_3a_{4_z} - a_{3_x})}_{B_2} + c_1 \underbrace{c_2\,(-s_3a_{4_x} + c_3a_{4_z} + a_{3_z})}_{A_2} \end{bmatrix} \tag{8.14}$$

It is seen that the y-component of Equation (8.14) only contains variables of θ_2 and θ_3. With θ_3 known this may be used to solve for θ_2 as:

$$\theta_2 = \arcsin\left(\frac{-y}{-s_3a_{4_x} + c_3a_{4_z} + a_{3_z}}\right) \tag{8.15}$$

with θ_3 and θ_2 known, Equation (8.14) can be used to solve for θ_1. Using both the x and z-components of the equation will reduce the number of solutions for θ_1 to one. Both equations can be used as they are similar. It is noticed that A_2 and B_2 are represented in the equations. The solution to θ_1 is then:

$$\theta_1 = \text{arctan2}(A_2x - B_2z, A_2z + B_2x) \tag{8.16}$$

Now the solutions for the first three joints are found. As earlier explained these are used for positioning the PoI.

The last three joints are used for setting the right orientation of the torso. Input rotation to the inverse kinematic model is given as an X-Y-Z Euler rotation, defining ${}^0_6\boldsymbol{R}$.
The rotation which should be implied to the last three links can be calculated as:

$${}^3_6\boldsymbol{R} = {}^0_3\boldsymbol{R}^{-1}{}^0_6\boldsymbol{R} \tag{8.17}$$

It is known that the rotation of the last three links is an X-Y-Z rotation about θ_4, θ_5, and θ_6, and thereby can Equation (8.17) be re-written as:

$$\begin{bmatrix} c_5c_6 & -c_5s_6 & s_5 \\ s_4s_5c_6 + c_4s_6 & -s_4s_5s_6 + c_4c_6 & -s_4c_5 \\ -c_4s_5c_6 + s_4s_6 & c_4s_5s_6 + s_4c_6 & c_4c_5 \end{bmatrix} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} \tag{8.18}$$

The right side of Equation (8.18) can be found using $\theta_1 \dots \theta_3$ to calculate ${}^0_3\boldsymbol{R}$, and the target rotation is used to find ${}^0_6\boldsymbol{R}$. From Equation (8.18) it is seen that θ_5 can be found as:

$$\theta_5 = \arcsin(r_{1,3}) \tag{8.19}$$

There are two solutions to (8.19), and again the Table L.1 on page 271 used to determine the correct solution. With θ_5 known, Equation (8.18) may be solved for θ_6:

$$\theta_6 = \text{arctan2}\left(\frac{r_{1,2}}{-\cos(\theta_5)}, \frac{r_{1,1}}{\cos(\theta_5)}\right) \tag{8.20}$$

θ_4 is now found as:

$$\theta_4 = \text{arctan2}\left(\frac{r_{2,3}}{-\cos(\theta_5)}, \frac{r_{3,3}}{\cos(\theta_5)}\right) \tag{8.21}$$

The described solution determines $\theta_1 \dots \theta_6$ for the right leg. The approach for finding $\theta_7 \dots \theta_1 2$ is the same, differing only in signs. This completes the inverse kinematics for both the right and left leg. Now the inverse kinematics for the arms will be solved.

8.1.2 Inverse Kinematics for the Arms

Solving the inverse kinematics for the arms takes on the same problem as for the legs. As it is valued that the arms will only be used for stability purposes, it is chosen to solve the problem for three links only. Thereby the same method as the lower three links of the leg can be used, namely Pieper's method. Using only three joints have the advantage that the goal can be specified using only position and not orientation. It has been chosen not to include joint 15 and 19 in the inverse kinematic solution and this can then later be used for rotation purposes. The rotation of the arms is then given by an Y-X-X rotation and the target position is given from:

$${}^s\boldsymbol{p}_h = {}^s_{13}\boldsymbol{T}\,{}^{13}_{14}\boldsymbol{T}\,{}^{14}_{16}\boldsymbol{T}\,{}^{16}\boldsymbol{p}_h \tag{8.22}$$

where ${}^s\boldsymbol{p}_h$ is the position of the hand with respect to frame $\{\text{s}\}$ placed in the shoulder. ${}^s\boldsymbol{p}_h$ is the input to the inverse kinematic solution of the arms and is also seen from Figure 8.3. The f_i function is found from ${}^{16}\boldsymbol{p}_h$ and ${}^{14}_{16}\boldsymbol{T}$. Using the magnitude squared of the f_i functions θ_{16} is found to be:

$$\theta_{16} = \text{arctan2}\left(Q_{16}, \pm\sqrt{1 - Q_{16}}\right) \tag{8.23}$$

where $Q_{16} = \frac{r^2 - {a_{15_z}}^2 - {a_{17_z}}^2}{2a_{15_z}a_{17_z}}$. Writing out Equation (8.22) yields:

$$ {}^{s}\boldsymbol{p}_h = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} s_{13}\underbrace{\left(-a_{17_z}s_{14}s_{16} + c_{14}\left(a_{17_z}c_{16} + a_{15_z}\right)\right)}_{A_5} \\ s_{14}\underbrace{\left(-a_{17_z}c_{16} - a_{15_z}\right)}_{B_4} + c_{14}\underbrace{\left(-a_{17_z}s_{16}\right)}_{A_4} \\ c_{13}\underbrace{\left(-a_{17_z}s_{14}s_{16} + c_{14}\left(a_{17_z}c_{16} + a_{15_z}\right)\right)}_{B_5} \end{bmatrix} \tag{8.24} $$

Using the inverse kinematic formulas found in [Craig, 1989], Equation (8.24) can be used to find θ_{14} by using the x-component:

$$ \theta_{14} = \text{arctan2}\left(B_4, A_4\right) \pm \text{arctan2}\left(\sqrt{{A_4}^2 + {B_4}^2 - y^2}, y\right) \tag{8.25} $$

The x and z-component of Equation (8.24) can be used to solve for θ_{13}:

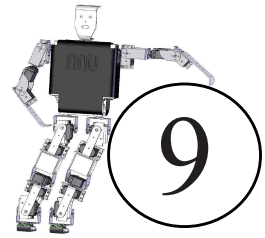
$$ \theta_{13} = \text{arctan2}\left(\frac{x}{A_5}, \frac{z}{B_5}\right) \tag{8.26} $$

The inverse kinematic solutions found for the right arm, can be directly applied to the left arm, since the construction of the biped robot is symmetric. This concludes the inverse kinematics for the arms and legs.

8.2 Partial Conclusion to Inverse Kinematics

Two solutions were proposed, one was the numeric solution, and the other was the closed form solution. In Appendix G on page 233 the numerical solution to the inverse kinematics was found. The solution was able to find the joint angles given rotations and positions. But it was found that the solution was very demanding in processing power and could therefore not be implemented on the real system. In [Craig, 1989] it is pointed out that the numerical solution to kinematics usually are so slow, that they are of no interest if the application should run real-time. The procedure described in this section could however be used for simulation or for off-line path generation. Instead another solution was investigated utilizing a closed-form solution. For this Pieper's solution was used. It was found that Pieper's solution gave a direct mapping from Cartesian space to joint space, and that the solution could be implemented on the real system. The closed-form solution is therefore used. In Appendix H on page 237 a verification of the closed from inverse kinematic solution is given. The verification is performed by applying a set of trajectories to the inverse kinematics, the output θ, is then applied to the kinematic model. The output of the kinematic model is compared with the input to the inverse kinematic model, to see if the limbs have been placed in the desired position and have the desired orientation. The result is a 100% match.

TRAJECTORY GENERATION



Chapter contents

THIS CHAPTER *describes the derivation of the walking trajectories. First some methods that are frequently used for generating walking trajectories, will be described. The prospects and considerations for each of the different types will be emphasized, which is done in Section 9.1. The chosen method will be given a detailed description. Both the steady-state walk, the start-up phase and the stop phase will be considered. This leads to estimation of the parameters that should be used to generate walking trajectories. All this is described in Section 9.2. Section 9.3 describes a simulation on the developed model, and the resulting trajectories are presented in Section 9.4.*

9.1 Trajectory Generation Methods

As stated in Section 3.4 on page 20, and as depicted in Figure 3.1 on page 14, a trajectory generator has to be developed. This section describes the methods often used for generating walking trajectories. The section will begin with a description of the two overall methods; on-line and off-line trajectory generation. Here after a number of different methods, which are most frequently used for generating on-line or off-line trajectories will be described.

9.1.1 On-line and Off-line Trajectory Generation

There are two overall methods used for trajectory generation, which is investigated in this thesis; on-line and off-line trajectory generation. An introduction to these two methods will be given in this section, and their prospects and considerations will be described.

On-line Trajectory Generation

When using on-line trajectory generation, walking trajectories are calculated on-the-fly as the robot walk. This makes the robot capable of changing its walking patterns if changes in the surroundings should occur. Changes in the surroundings could be presented as a tilted surface, uneven surface, obstacles in the walking path, etc. On-line trajectory generation can be enhanced by using self-learning algorithms. This is seen in [Capi et al., 2003]. A disadvantage of this type of trajectory generation is the fact that it is very demanding with regards to processing power. Most projects regarding on-line trajectory generation is limited to simulation due to the high demand of processing power and complex implementation, as in [Kondak and Hommel, 2003].

In [Kondak and Hommel, 2003] they propose a method where a biped robot is performing stable walk without any precomputed trajectories. The trajectory generation is made on-line, along with model-based control of the robot. The downside of the results in [Kondak and Hommel, 2003] is however, that the algorithms are only implemented on a model in simulation. This is the most widespread approach taken in the literature, when on-line trajectory generation is the subject.

Off-line Trajectory Generation

When using off-line trajectory generation, walking trajectories are calculated beforehand. Trajectories could be calculated using an external computer and then implemented on the robot as a look-up table or as functions describing the trajectories. The advantage of calculating the trajectories off-line is that the new reference positions are found faster, making it easier to fulfill the real-time demands. Hence, it is possible to calculate trajectories, using the dynamical model to ensure energy efficiency, if this is a matter of interest, and then implement the trajectories when these have been calculated. This method is very applicable in systems where the processing power is very limited.

Due to the limited processing power in this project, being a 200 MHz ARM9 processor, it has been chosen to use off-line trajectory generation. Different methods that can be used to generate the trajectories are described in the following section. A disadvantage of using off-line trajectory generation is that the possibility of changing the walking patterns on-the-fly is limited to the number of trajectories stored on the robot.

One could consider the situation where the surroundings are changing, i.e., the surface is tilted or in any other way changed. In this case the robot could have difficulties maintaining balance if it is not capable of changing the walking patterns [Buss et al., 2003].

9.1.2 Methods for Generating Walking Trajectories

This section gives a description of the methods which are most often used for trajectory generation. Only the basic concept of each method will be described, as there exists several derivatives of each method where the concept is applied with small alterations. The method chosen for this project will be given a detailed description in Section 9.2. Common for many of the methods is that they can both be implemented for on-line off-line trajectory generation, depending on the processing power at hand.

Inverted Pendulum

The complex mechanical structure of a biped robot is sometimes modeled as an inverted pendulum to reduce the complexity of the model. The basic principle is shown in Figure 9.1. This figure shows a walking cycle starting with the end of a DSP and then initiates a SSP-L. In SSP-L the swing leg, being green leg in the figure, swings with a parabolic shape until impact. After the impact there is a short DSP where the mass, being the CoM of the entire robot, is moved forward and the weight is shifted to the new supporting leg, being the right leg. In both SSP-L and SSP-R shown, the mass is moved in a path similar to an inverted pendulum. Previous studies try to model a biped robot as an inverted pendulum in the sagittal plane, as shown in Figure 9.1, as well as in the frontal plane. This can be seen as a 3-dimensional pendulum [Kajita et al., 2001]. By using an inverted pendulum to model the robot, the trajectories for the CoM of the robot will become a smooth sine-wave in both the sagittal plane as well as in the frontal plane. This method is mostly used as a way to simplify the model of the biped because this allows a more simple linear model. The method can only be used to dictate which path or trajectory for instance the CoM should take, and further trajectory planning has to be done for the legs, to ensure that they are in the right position at the right time. When using this method to generate trajectories, the trajectories could be verified by calculating the ZMP stability.

Walking Trajectories Based on ZMP-Stability

This method is probably the most used trajectory generating method. Hence, this method is discussed and applied in several articles, see [Choi et al., 2004], [Huang et al., 2000] and [Peng et al., 2005]. The basic concept of this method is to find a stable ZMP trajectory and then calculate the trajectories for the feet that follows that ZMP trajectory when implemented. After the trajectories have been calculated, a number of simulations are performed with smaller alterations for the trajectory of the hip. The trajectory that yields the largest stability is then chosen.

A deviation of this method has been proposed in [Huang et al., 2001] where the concept of this method is to first calculate the walking trajectories for the feet and then calculate a number of trajectories for the hip. When these calculations have been performed, the ZMP is calculated along the walking trajectory. To maintain a dynamical stable walk, the ZMP should be located inside the PoS. A simulation is made for each of the different hip-trajectories and the trajectory with the largest margin of stability is chosen.

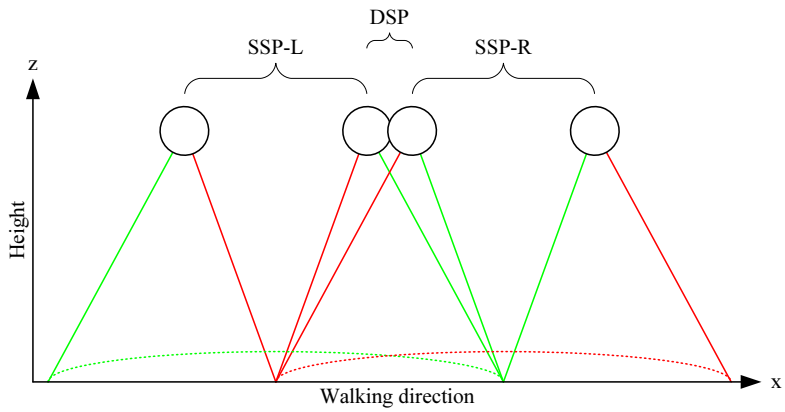


Figure 9.1: *The basic principle of using an inverted pendulum to generate trajectories. Two steps in the sagittal plane is illustrated. The green line is the right leg and the red line is the left leg. The circle illustrates the CoM of the entire robot.*

The appealing property of using this method is that human-like walking patterns could be realized by analyzing human walk and then set the desired step length, foot angle, etc. according to this. The trajectory of the hip should also be fitted to human walking trajectories but several hip trajectories should be generated with small alterations. After having simulated all the trajectories, the trajectory with the best stability properties should be chosen. A downside of this method is however, that the energy in the movements is not exploited in a sufficient degree, as for instance with the inverted pendulum method. This could result in an unnecessary load exposed on the actuators.

CoM Method

Another popular method is the centre of mass method, which has been studied and previously used in [Christensen et al., 2006]. This method relies on calculations of the CoM, and keeping this point within the PoS. It was discovered that this type of gait generation resulted in a stable walk of the robot. But the walk was not dynamical, only static. As the requirements to the CoM being within the PoS, given by the supporting foot/feet, results in shifting of the weight from foot to foot. This results in a ”duck-like” walk, which does not resemble human walk. The upside of this method, is that it is relatively simple, and the trajectories can easily be calculated off-line. It should be noted that the CoM method differs significantly from the ZMP method, as the dynamics of the movement is not included in the CoM method, but only the static location of the CoM is considered.

Cyclic Motion Generator

Cyclic motion generation is often mentioned as being the method that humans use to generate walking trajectories, see [Borghese et al., 1996], [Ogino et al., 2003] and [Nielsen, 2003]. When humans walk, the cyclic walking patterns are generated in a nerve center in the spinal cord. This is often referred to as the spinal central pattern generator, called central pattern generator (CPG). When CPG is used to achieve human-like motion, a cyclic motion generator is implemented, generating the cyclic trajectories that characterize human walk. The energy efficiency should be optimal when using CPG, see [Ogino et al., 2003], since CPG seeks to copy human walking patterns that are said to be optimal with regards to energy efficiency [Popovic, 2006].

This method is often implemented with adaptive, self-improving algorithms to realize human-like, energy efficient motion [Takahashi et al., 2005]. The adaptive property improves the stability of the robot, and enables the robot to walk on changing terrain, i.e., walking on a surface changing from being plain and solid to being uneven. Due to complex calculations, these adaptive properties can however impose a heavy load on the CPU.

Human Trajectories

Another way of designing trajectories is by using real human trajectories, recorded in a motion studio, as studied in [Bruneau et al., 2001] and in [Christensen et al., 2006]. In Section 3.2 trajectories for a human were analyzed with regards to posture. Using these trajectories as input trajectories would result in the same posture, and thereby the desired walk. The trajectories do, however belong to a person having a certain height and weight. A solution to this is to scale the trajectories, which is possible, since the biped robot has human proportion. Only an inverse kinematic model is needed to translate the trajectories into joint rotations. The disadvantages of this method is that calculation of trajectories from fitted polynomials, depending on the order, could result in heavy computation, which would be infeasible in an embedded system. Even if the human trajectories are used, and the human body always tries to emphasize low energy consumption, energy optimization can not be guaranteed using the fitted functions. This is caused by the fact that the design of the biped robot is not completely the same as the human body, e.g. the robot has servos in each joint working as muscles while the human body has muscles divided throughout the entire body. Because of this the robot is not able to completely repeat the human motion, and thereby a non-optimal energy solution would be obtained. Another reason why it can not be guaranteed that this method implements the most energy efficient gait, is that the functions are scaled. Some information might be lost in scaling, and no further calculations is done on this matter after the implementation.

Energy Optimal Dynamic Walk

As the previous method relies on fitted functions an energy efficient gait can not be ensured, another method is proposed. This method is called passive dynamic walk and was first proposed by Tad McGeer in [McGeer, 1990], where a 2D biped walker was set to walk downhill a slope utilizing only the energy given by the gravitational force. It was found that the biped robot walked in cyclic motions resembling human walk and emphasizing low energy consumption. [Collins et al., 2001] later implemented this approach on a 3D biped walker. But the passive dynamic walk was maintained without the use of actuated joints, which is difficult to implement in a biped robot that should be able to walk on surfaces not necessarily downhill.

Instead, a new method is proposed which emphasizes energy optimal dynamic walk, studied previously in [Capi et al., 2003] and [Djoudi et al., 2005]. This method calculates the most energy efficient trajectories for the robot, based on some given input set, e.g. velocity, turning curve and robot posture. Using a dynamic model of the robot, this method is able to calculate the next movements ahead in time and thereby ensuring the most energy optimal trajectories, for the task given ahead. The advantage of this method is that the energy efficient requirement is fulfilled and so is the requirement on scalability, as the trajectories are calculated from the dynamic model. A disadvantage is that the method could result in high computation requirement, as

the model of a biped robot can become complicated and thus require high computation time, see [Christensen et al., 2006]. The biped robot in this project consists of servo motors as muscles in the joints. It is not possible to release the servos while walking, and thus it is not possible utilize the gravity to guide the leg in the most optimal trajectory, which would require minimal energy. It is though possible to calculate the trajectories which are most energy efficient with regards to the torque. As the servos require more energy at higher torques, calculating the trajectories which minimize the torques would also result in the trajectories which are the most energy optimal.

9.2 Establishing the Trajectories

As stated in the previous section, it was chosen to calculate the reference trajectories off-line, in order to ease the workload placed on the on-board computer. The trajectories will be designed such that periodic symmetric gait is obtained. From the methods described in Section 9.1 it is chosen to design the walking trajectories based on the ZMP-stability. It is however chosen to include some of the properties from the cyclic motion generator, especially the energy efficiency will be considered when designing the trajectories. Furthermore it is chosen also to include some considerations toward the human trajectories method, since the goal is to obtain human-like walk. Human trajectories will however not be imitated exactly, but some of the key-parameters will be derived from the humans.

As the effect on stability gained by moving the arms is relatively little, compared to the effect of moving the torso and the lower body, it is chosen not to generate off-line trajectories for the arms. This decision is also based on the fact, that the model is not a precise description of the system, and it is therefore valued that a more feasible solution is to determine the movement of the arms on-line, based on the sensor feedback.

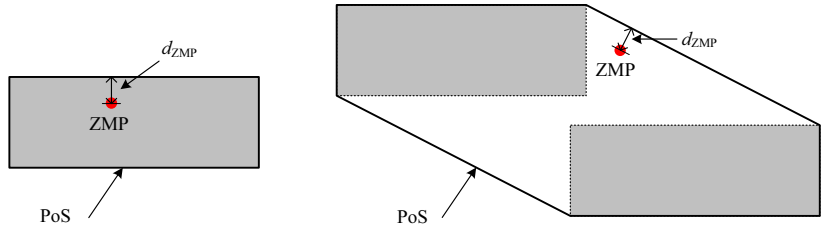
The following will contain a more thorough description of the chosen method.

9.2.1 ZMP-stability

It is desired to design the trajectories such that the robot is stable, ideally making it capable to walk if not exposed to any external disturbances. The ZMP will be used to ensure that the trajectories result in dynamical stability, in the ideal case where no disturbances are introduces. As stated i Section 3.4 on page 20 the goal is to obtain human-like walk, which ensures that the robot keeps walking.

Often the trajectories for biped locomotion are designed, to make the ZMP track a predetermined path. This is done by setting up the equations describing the movement of the ZMP, as a function of the movement of all links. These equations are inverted, and the movement of the links can be found when the movement of the ZMP is known. As stated in [Huang et al., 2001] and [Huang et al., 2000] this is however a very complex task, and not always an effective approach, since some paths can require a significant large acceleration in the hip joint, making it impossible to achieve the desired path of the ZMP. It is therefore chosen to use the approach proposed in [Huang et al., 2001], where a number of smooth hip motions are designed, by altering central parameters. Then the different trajectories are compared through simulation, to investigate their overall stability during locomotion. The comparison is based on the stability margin,

where the stability margin is the minimum distance, d_{ZMP}, from the boundary of the stable region to the ZMP. If the ZMP is outside the stable region, d_{ZMP} will be negative. The stable region is the convex hull of all contact points called PoS, see Figure 9.2.



(a) SSP (b) DSP

Figure 9.2: *The stable region is the convex hull of all contact points, called PoS. The stability margin, d_{ZMP}, is the minimum distance from the boundary of the stable region to the ZMP.*

If the stability margin is large, the moment refraining the biped robot from tipping over, is also large. The stability of a trajectory can be specified from the minimum value of d_{ZMP} during execution of that trajectory. In [Huang et al., 2001] it is simply chosen to use the trajectory with the largest stability margin. In this project it is however chosen, to define an acceptable region for the ZMP, see Figure 9.3(a). The trajectories are now considered stable enough if the ZMP is inside the acceptable region, thus:

$$d_{\text{ZMP}} \geq l_{\text{accept}} \tag{9.1}$$

However it is not possible to obtain a smooth walk and still meet the stability requirement of Equation (9.1) at the same time, since this would require infeasible large movement of the CoM during the DSP. Therefore the trajectories are compared by their stability index, i_{stab}, calculated as:

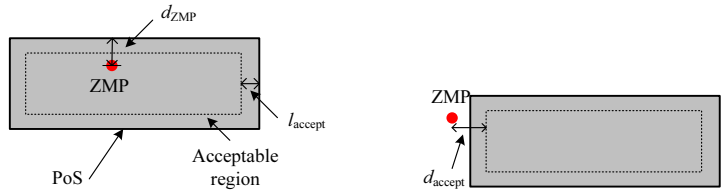
$$i_{\text{stab}} = \int_0^{T_{\text{sim}}} d_{\text{accept}}^2 (1 + c_{\text{PoS}}) dt \tag{9.2}$$

where d_{accept} is the distance from the acceptable region to the ZMP when $d_{\text{ZMP}} < l_{\text{accept}}$, which is illustrated in Figure 9.3(b). If the ZMP is inside the acceptable region, d_{accept} is zero. c_{PoS} is a penalty constant punishing the stability index of the trajectories with poor stability properties, thus:

$$\begin{aligned} c_{\text{PoS}} &= 0 \quad \text{for, } d_{\text{ZMP}} \geq l_{\text{accept}} \\ c_{\text{PoS}} &> 0 \quad \text{for, } d_{\text{ZMP}} < l_{\text{accept}} \end{aligned} \tag{9.3}$$

The stability index indicates how must the ZMP is outside the acceptable region, thus a smaller stability index relates to a more stable trajectory.

Besides from stability properties, the trajectories are further compared on energy consumption. Consideration of the energy is usually an approach taken when designing the trajectories with the cyclic motion generator as described in Section 9.1.2. There are two main reasons for including the energy considerations in the selection of a trajectory. First, the goal is to obtain human-like walk, and from [Azevedo et al., 2005]



(a) Stability regions (b) Stability measure

Figure 9.3: *If the ZMP is inside the acceptable region ($d_{ZMP} \geq l_{accept}$) as in 9.3(a) the stability index is not influenced. If the ZMP is outside the acceptable region, as in 9.3(b), the stability index is however influenced.*

it is known that humans optimize their walk with respect to energy consumption. Second, in order to lengthen the life-time of the batteries. This approach is similar to the one taken in [Arakawa and Fukuda, 1996], where the evaluation of the trajectories is based on the total energy consumption of the actuators, but also on a penalty vector that ensures that the constraints are met. This could e.g. be the d_{ZMP} requirement. Now the final trajectory is the one using the minimum energy and having the smallest stability index.

The energy, or rather the consumed power is calculated as:

$$P = \frac{1}{T_{\text{sim}}} \int_0^{T_{\text{sim}}} \tau \dot{\theta} \, dt \tag{9.4}$$

It is possible to derive the trajectories both in joint space as in Cartesian space. It is chosen to derive trajectories in Cartesian space for each foot and the torso.

Since the biped robot is symmetric, the same trajectories can be used for the left and right foot, if a time shift of half a walking cycle is applied. Having 6 DoF in the legs, all joint trajectories for the legs are specified by kinematic constraints, when trajectories for the feet and the torso are specified. As in [Huang et al., 2001], the trajectories in the sagittal and frontal plane are found independently of each other. This is possible, since the x-component of the ZMP is approximately independent of motion in the y-direction, which is perpendicular to the axis [Peng et al., 2005]. The same is valid for the y-component and movement in the x-direction. This is an advantageous feature of the ZMP, as the derivation of the trajectories is simplified, when the planes can be considered individually.

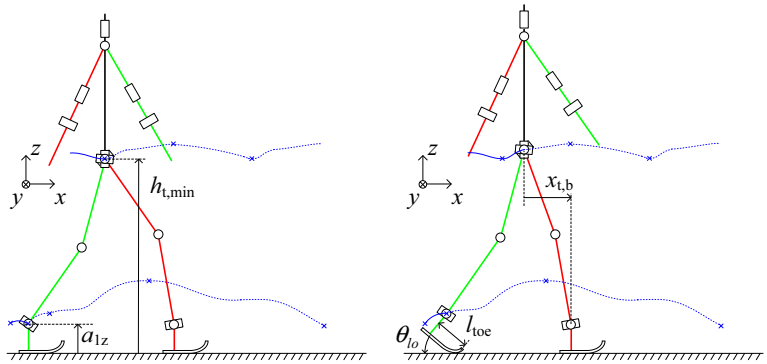
The motion of the feet specifies the overall movement of the biped robot. As the trajectory of the swing foot has little influence on the overall stability, compared to the trajectory of the torso, it is chosen first to design the trajectory for the swing foot, with no considerations to the resulting path of the ZMP. This is done in Section 9.2.2. In the following section, Section 9.2.3, the trajectory of the torso is designed, such that the stability index of Equation (9.2) is minimized, and the energy consumption is reduced. It is chosen to design the trajectories of the steady-state walk before designing the trajectories for the start-up phase, since the steady-state walk is the main focus in this project. The start-up and stop trajectories will then be designed to fit the trajectories of steady-state walk. These are mentioned in Section 9.2.4.

9.2.2 Foot Trajectories

The trajectories for the ankles will be designed in this section. Given these and the orientation of the feet, the trajectories of the feet can be found from kinematic constraints. It is the trajectories of the feet that determine the velocity and the direction of the biped robot across ground. Therefore it is necessary to specify the desired walking speed, v_{walking}, and the desired step length, l_{step}. Then the time period for one step can be found as:

$$T_{\text{step}} = \frac{l_{\text{step}}}{v_{\text{walking}}} \tag{9.5}$$

Note that the cycle time is $T_{\text{cycle}} = 2T_{\text{step}}$.



(a) DSP: $t_1 = (k-1)T_{\text{step}}$

(b) Lift off: $t_2 = kT_{\text{step}} - T_{\text{SSP}}$

(c) Middle of SSP: $t_3 = (k-1)T_{\text{step}} + T_{\text{max}}$

(d) Impact: $t_4 = kT_{\text{step}}$

Figure 9.4: *Illustration of the walking cycle in the sagittal plane, the green colour represents right-side limbs and the red represents the left-side. 9.4(a) shows the middle of the DSP, where the supporting leg is shifted, in 9.4(b) the SSP is initiated with the rear foot leaving the ground, in 9.4(c) the ankle reaches its maximum height and in 9.4(d) the SSP is terminated with the impact. The blue lines illustrates the trajectory taken by the torso and the right foot.*

Foot Movement in the Sagittal Plane

The foot trajectory for the k'th step is illustrated in Figure 9.4. It is defined to start in the DSP when the heel of the rear foot leaves the ground, at time t_1, as depicted in

Figure 9.4(a). The k'th step ends when the heel of the other foot in Figure 9.4(d) leaves the ground at time t_4. The SSP starts at time t_2, with the toe of the rear foot leaving the ground, this is illustrated in Figure 9.4(b). Then at time t_3 the ankle of the swing leg, passes through its maximum point, $(x, z) = (l_{\text{a,max}}, h_{\text{a,max}})$, as illustrated in Figure 9.4(c). And finally the SSP is terminated when the swing leg impacts the ground at time t_4, shown in Figure 9.4(d). It is chosen to impact the ground with a flat foot, since the foot is not design for impact and tipping on heel. The time instants are given as follows:

$$\begin{aligned} t_1 &= (k-1)T_{\text{step}} \\ t_2 &= kT_{\text{step}} - T_{\text{SSP}} \\ t_3 &= (k-1)T_{\text{step}} + T_{\text{max}} \\ t_4 &= kT_{\text{step}} \end{aligned} \tag{9.6}$$

where T_{SSP} is the duration of the SSP and T_{max} is the duration from the cycle starts until the swing foot reaches it maximum height.

From the above, a number of constraints on the trajectory can be found:

$$x_{\text{a}}(t) = \begin{cases} (k-1)l_{\text{step}} & ,t = t_1 \\ (k-1)l_{\text{step}} + a_{1\text{z}}\sin(\theta_{\text{lo}}) + l_{\text{toe}}\left(1 - \cos(\theta_{\text{lo}})\right) & ,t = t_2 \\ (k-1)l_{\text{step}} + l_{\text{a,max}} & ,t = t_3 \\ kl_{\text{step}} & ,t = t_4 \end{cases} \tag{9.7}$$

$$z_{\text{a}}(t) = \begin{cases} a_{1\text{z}} & ,t = t_1 \\ a_{1\text{z}}\cos(\theta_{\text{lo}}) + l_{\text{toe}}\sin(\theta_{\text{lo}}) & ,t = t_2 \\ h_{\text{a,max}} & ,t = t_3 \\ a_{1\text{z}} & ,t = t_4 \end{cases} \tag{9.8}$$

where $x_{\text{a}}(t)$ and $z_{\text{a}}(t)$ are the position of the ankle in the sagittal plane at time t, as depicted in Figure 9.4, $a_{1\text{z}}$ is the height of the ankle and l_{toe} is the length from the ankle to the toe, both shown in Figure 9.4, θ_{lo} is the angle of the foot at lift off. Furthermore some constraints on the orientation of the foot can be found:

$$\boldsymbol{o}_{\text{f}}(t) = \begin{cases} [0\ 0\ 0]^{\text{T}} & ,t = t_1 \\ [0\ -\theta_{\text{lo}}\ 0]^{\text{T}} & ,t = t_2 \\ [0\ 0\ 0]^{\text{T}} & ,t = t_3 \\ [0\ 0\ 0]^{\text{T}} & ,t = t_4 \end{cases} \tag{9.9}$$

where $\boldsymbol{o}_{\text{f}}(t)$ is the orientation of the foot, given as XYZ Euler angles fixed to the initial frame, as described in Chapter 8 on page 101. Some constraints can also be attached to the velocity of the ankle. It is known that the foot is placed flat on the ground at time t_1, thus the following constraints can be applied:

$$\begin{aligned} \dot{x}_{\text{a}}(t_1) &= 0 \\ \dot{z}_{\text{a}}(t_1) &= 0 \\ \dot{\boldsymbol{o}}_{\text{f}}(t_1) &= [0\ 0\ 0]^{\text{T}} \end{aligned} \tag{9.10}$$

Furthermore the velocity trajectories, $\dot{x}_{\text{a}}(t)$,$\dot{z}_{\text{a}}(t)$ and $\dot{\boldsymbol{o}}_{\text{f}}(t)$, have to be differentiable and continuous for all t and their derivative, $\ddot{x}_{\text{a}}(t)$, $\ddot{z}_{\text{a}}(t)$ and $\ddot{\boldsymbol{o}}_{\text{f}}(t)$, has to be continuous for all t. Note that this should also apply in the breakpoints, where $t = kT_{\text{step}}$. These requirements ensure smooth trajectories with no sudden accelerations.

Foot Movement in the Frontal Plane

If straight walk is the only goal, no movement of the feet in the frontal plane is necessary, thus:

$$\dot{y}_{\text{a}}(t) = 0 \tag{9.11}$$

This is because the movement in the frontal plane is performed by the torso.

9.2.3 Torso Trajectory

As explained earlier in this chapter, the trajectory of the torso will be designed such that the locomotion of the biped robot fulfills the stability requirement, including considerations on the energy efficiency. The stability requirement is that the ZMP should be inside the acceptable region, as illustrated in Figure 9.3(a). In Section 3.2 on page 16 it was decided, that the posture of the torso should always be maintained upright. Which results in the following constraints:

$$\boldsymbol{o}_{\text{t}}(t) = [0\ 0\ \theta_{\text{t,z}}]^{\text{T}} \tag{9.12}$$

where $\boldsymbol{o}_{\text{t}}(t)$ is the orientation of the torso, and $\theta_{\text{t,z}}$ is the rotation around the z-axis. As for the design of the foot trajectories, this design is also divided into a part concerning the trajectories of sagittal movement and of frontal movement. When considering the stability, the stability margin, d_{ZMP}, is divided into two parts, x_{ZMP} and y_{ZMP}, which denotes the stability margin in the sagittal and frontal direction, respectively.

Torso Movement in the Sagittal Plane

The torso of a human moves relatively little up and down during walk. Based on these observations, it is chosen that the trajectory of the torso in the z-direction should have a simple appearance, where it reaches maximum at time $t = t_3$, as shown in Figure 9.4(c), and reaches minimum in the middle of DSP at time $t = t_1$, as illustrated in Figure 9.4(a). These constraints can be formulated as:

$$z_{\text{t}}(t) = \begin{cases} h_{\text{t,min}} & , t = t_1 \\ h_{\text{t,max}} & , t = t_3 \end{cases} \tag{9.13}$$

The values of $h_{\text{t,min}}$ and $h_{\text{t,max}}$ are derived from human walk, this is described in Section 9.2.5.

For the trajectories to be cyclic $z_{\text{t}}(t_1) = z_{\text{t}}(t_4)$, $\dot{z}_{\text{t}}(t_1) = \dot{z}_{\text{t}}(t_4)$ and $\ddot{z}_{\text{t}}(t_1) = \ddot{z}_{\text{t}}(t_4)$. Furthermore the constrains in Equation (9.13) are global maximum and minimum, thus $\dot{z}_{\text{t}}(t_1) = \dot{z}_{\text{t}}(t_3) = 0$.

As earlier mentioned, it is difficult to track a predefined path of the ZMP, and there it is difficult to make the walk appear human-like. Therefore a series of hip motions in the sagittal plane are designed, and one is selected based on its stability and energy properties. In [Huang et al., 2001] trajectories for the torso are made by varying two parameters. The first is the distance in the x-direction from the hip to the ankle of the supporting foot at the beginning of SSP, $x_{\text{t,b}}$, and the second is the distance from the hip to the same point at the end of SSP, $x_{\text{t,e}}$, as depicted in Figure 9.4(b) and 9.4(d) respectively. This approach is also used in this project, and the two parameters are set to vary within a range of:

$$\begin{array}{rcl} 0.2l_{\text{step}} \leq & x_{\text{t,b}} & \leq 0.5l_{\text{step}} \\ 0.3l_{\text{step}} \leq & x_{\text{t,e}} & \leq 0.7l_{\text{step}} \end{array} \tag{9.14}$$

The ranges of Equation (9.14) are found through repeating simulations, if values of $x_{\text{t,b}}$ and $x_{\text{t,e}}$ outside this range are used, the biped robot will not be walking. This results from the fact that the swing leg will simply not be able to reach the ground, due to the length of the leg.

Now the following constraints for the torso trajectory can be formulated:

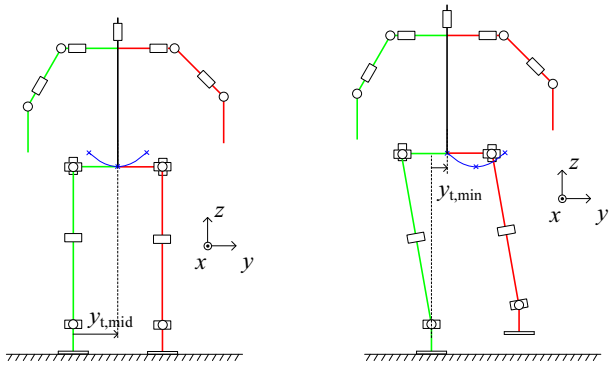
$$x_{\text{t}}(t) = \begin{cases} kl_{\text{step}} - x_{\text{t,b}} & , t = t_2 \\ kl_{\text{step}} + x_{\text{t,e}} & , t = t_4 \end{cases} \tag{9.15}$$

Furthermore, if the trajectory should be smooth and periodic, the following constraints should be fulfilled:

$$\begin{aligned} x_{\text{t}}(t_1) &= x_{\text{t}}(t_4) - l_{\text{step}} \\ \dot{x}_{\text{t}}(t_1) &= \dot{x}_{\text{t}}(t_4) \\ \ddot{x}_{\text{t}}(t_1) &= \ddot{x}_{\text{t}}(t_4) \end{aligned} \tag{9.16}$$

Torso Movement in the Frontal Plane

The trajectory describing the motion of the torso in the frontal plane, $y_{\text{t}}(t)$, is constructed using the same approach as for the sagittal plane. The trajectory is illustrated in Figure 9.5, where Figure 9.5(a) shows the middle of the DSP, at time t_1, where the torso is placed right between the legs. Figure 9.5(b) shows the middle of SSP, at time t_3, where the torso reaches its minimum distance from the right ankle. This distance is called $y_{\text{t,min}}$, and is measured from the ankle of the supporting foot to the spinal column. $y_{\text{t,min}}$ will be varied to obtain a series of smooth trajectories, and between those satisfying the stability requirement, the one using least amount of energy will be chosen.



(a) DSP: $t_1 = (k-1)T_{\text{step}}$ (b) SSP: $t_3 = (k-1)T_{\text{step}} + T_{\text{min}}$

Figure 9.5: *Illustration of the walking cycle in the frontal plane. 9.5(a) shows the middle of the DSP, where the torso is placed right between the legs, in 9.5(b) the torso reaches its maximum divergence from the center position. The blue line illustrates the trajectory taken by the torso.*

The trajectories for the frontal movement should satisfy the following constraints:

$$y_{\text{t}}(t) = \begin{cases} y_{\text{t,mid}} & , t = t_1 \\ y_{\text{t,min}} & , t = t_3 \end{cases} \tag{9.17}$$

where $y_{\text{t,min}}$ is set to vary within a fixed range of:

$$-0.2y_{\text{t,mid}} \leq y_{\text{t,min}} \leq 0.4y_{\text{t,mid}} \tag{9.18}$$

The ranges specified in Equation (9.18) are found through presimulations. Furthermore, as the motion should be cyclic, following constraints should be fulfilled:

$$\begin{aligned} y_{\text{t}}(t_1) &= y_{\text{t}}(t_4) \\ \dot{y}_{\text{t}}(t_1) &= -\dot{y}_{\text{t}}(t_4) \\ \ddot{y}_{\text{t}}(t_1) &= \ddot{y}_{\text{t}}(t_4) \end{aligned} \tag{9.19}$$

and since $y_{\text{t}}(t) = y_{\text{t,min}}$ should be the maximum deviation from center, the following constraint must be satisfied:

$$\dot{y}_{\text{t}}(t_3) = 0 \tag{9.20}$$

9.2.4 Start-Up and Stop Phase

The establishment of the trajectories for the start-up and stop phase is almost identical to the steady-state walk. The only difference is, that the start-up phase trajectories are responsible for moving the robot from initial position and into the steady state walk. The trajectories of the stop phase are responsible for taking the robot from steady state walk into initial position. In order to minimize the redundancy, the establishment of the trajectories for the start-up and stop phase are placed in Appendix I on page 243.

9.2.5 Scaling Parameters from Human Walk

In Section 3.2.2 on page 17 some parameters describing the human walk were found in [Borghese et al., 1996]. It has been chosen to use data from [Borghese et al., 1996] to create the trajectories, but more detailed data than those found in Section 3.2.2 has to be found. In Table 9.1, the data from the human test subject is listed. It should however be noted that, the data of Table 9.1 can not be found directly in [Borghese et al., 1996], but can be derived from the presented data. To scale the human walking parameters, the height of the test subject compared to the robot is used. This is calculated to be:

$$\rho_{\text{h,r}} = \frac{h_{\text{test subject}}}{h_{\text{robot}}} = \frac{1.69\,\text{m}}{0.58\,\text{m}} = 2.914 \tag{9.21}$$

The data has been extracted from human walk, where a human test subject walked the same path, at approximately the same speed six times (maximum deviation in walking speed was $0.027\,\text{m/s}$).

The scaled parameters can however not be applied directly on the robot, since the maximum velocity of the actuators introduces constraints. The scaled walking velocity of $0.374\,\text{m/s}$ can not be obtained, as a result it is necessary to change this parameter, and through repeating simulation on the model of the biped robot, it is found that the maximum possible walking velocity is approximately half the scaled value, thus $v_{\text{walking}} = 0.187\,\text{m/s}$. Furthermore it is not possible to walk with a step length of $0.226\,\text{m}$, as it should be possible to vary the parameters $x_{\text{t,b}}$, $x_{\text{t,e}}$ and $y_{\text{t,min}}$ within the ranges specified by Equation (9.14) and (9.18). The walking speed is reduced, which supports an reduction in the step length. It is therefore chosen to reduce the step length with 30% since this will permit the intended variations, thus $l_{\text{step}} = 0.158\,\text{m}$. Only v_{walking} and l_{step} have been changed, compared to the scaled values.

Parameter	Human	Scaled
v_{walking}	$1.084\,\text{m/s}$	$0.374\,\text{m/s}$
l_{step}	$0.658\,\text{m}$	$0.226\,\text{m}$
$h_{\text{a,max}}$	$0.112\,\text{m}$	$0.039\,\text{m}$
$l_{\text{a,max}}$	$1.48 l_{\text{step}}$	$1.48 l_{\text{step}}$
$h_{\text{t,max}}$	$0.702\,\text{m}$	$0.270\,\text{m}$
$h_{\text{t,min}}$	$0.590\,\text{m}$	$0.240\,\text{m}$
θ_{lo}	$1.40\,\text{rad}$	$1.40\,\text{rad}$
T_{SSP}	$0.8 T_{\text{step}}$	$0.8 T_{\text{step}}$
T_{max}	$0.48 T_{\text{step}}$	$0.48 T_{\text{step}}$

Table 9.1: *Parameters used to generate the trajectories. The scaled parameters are obtained from the parameters derived from human walk, by using the $\rho_{h,r}$ of Equation* (9.21)*. Note that $v_{walking}$ and l_{step} are modified, in order to adapt the parameters to the actuators. T_{step} is found from Equation* (9.5)*.*

9.2.6 Cubic Splines

One polynomial for each trajectory, fulfilling all the requirements on position, velocity and acceleration, would have a very high order, and therefore not be efficient for implementation. For this reason it is chosen to use cubic splines, which are widely used to create the trajectories in joint space in the area of robotics. Cubic splines consist of interpolated third-order polynomials, and can assure continuity of velocity and acceleration [Guan et al., 2005]. It also has been investigated if it is possible to use MATLAB's polyfit, this is however very important that the trajectories have the same velocity in the end of a cycle, as in the beginning, otherwise the trajectories are not cyclic. Polyfit does not offer the possibility of specifying velocities in certain points, as the cubic spline method does, and it is therefore not possible to use polyfit to create the trajectories.

9.3 Simulation of Trajectories

In order to find the best trajectory a number of simulations are performed on the complete model of the biped robot. The derivation of the model is described in Chapter 7.

The simulation is divided into several parts, first a number of simulations are performed to find the values of $x_{\text{t,b}}$ and $x_{\text{t,e}}$ that provide the largest stability index. Those providing an acceptable stability index, are compared on energy efficiency, and the final values of $x_{\text{t,b}}$ and $x_{\text{t,e}}$ are chosen to be those using the least amount of energy, which is described in Section 9.3.1. Next step, described in Section 9.3.2, is to find the value of $y_{\text{t,max}}$ providing the largest stability index and using the least amount of energy, this is conducted using the same approach as for the x-direction. Next a number of simulation are performed to find the values of $x_{\text{t,max}}$, $y_{\text{t,max,up}}$ and $T_{\text{y,max,up}}$ which describes the start-up phase, the simulation of the start-up phase is placed in Section I.1 on page 243. These parameters will be chosen solely on their stability properties, with no considerations to energy consumption. As a last step the parameters for the stop phase are determined, this is done exactly like for the start-up phase, and the description of the parameter determination can be found in Section I.2 on page 246.

9.3.1 Finding Parameters for Sagittal Movement

The trajectories for the torso are found by varying $x_{\text{t,b}}$ and $x_{\text{t,e}}$ both in steps of $0.01 l_{\text{step}}$ and within the ranges specified in Equation (9.14). This gives a total number of 1271 different trajectories and thereby 1271 simulations. For each simulation the stability index is calculated using Equation (9.2) on page 115, this equation is however general, and the exact equation that is used for the stability index of the sagittal plane is as follows:

$$i_{\text{stab,x}} = \int_0^{T_{\text{sim}}} \left(d^2_{\text{accept,f}} + d^2_{\text{accept,b}} c_{\text{b}} \right) \left(1 + c_{\text{PoS}} \right) dt \tag{9.22}$$

where d_{accept} and c_{PoS} are the same as explained on 115, only now d_{accept} is divided into those in front of the foot, and those in the back (denoted with a subscripted 'f' for front or 'b' for back). When walking dynamical, humans fall forward, it is therefore chosen to punish a ZMP placed in the back of the foot harder than one in front of the foot. This is done with the penalty constant c_{b}. The following values are used for the constants, when calculating $i_{\text{stab,x}}$: $l_{\text{accept}} = 3\,\text{mm}$, $c_{\text{PoS}} = 2$ and $c_{\text{b}} = 5$. Note that to neglect the starting difficulties when calculating $i_{\text{stab,x}}$, two walking cycles are simulated, but only the last one is used to calculate the stability index.

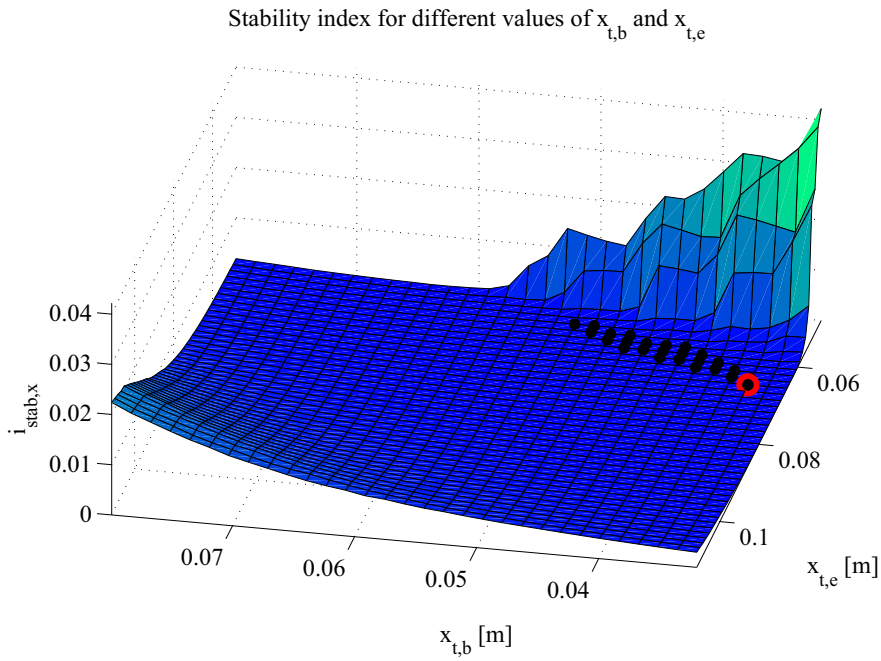
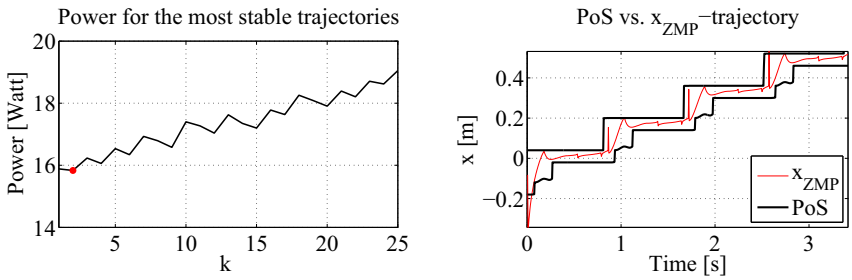


Figure 9.6: *The stability index shown as a function of $x_{t,b}$ and $x_{t,e}$. The black dots, mark the trajectories which are compared on energy consumption. The red dot marks the final trajectory.*

The values of $i_{\text{stab,x}}$ calculated for the 1271 simulations can be seen in Figure 9.6. Between all the trajectories, the 2 percent having the smallest stability index are chosen to be compared on energy consumption. Those are the ones marked with a black dot. The total energy consumption of those trajectories are calculated using Equation (9.4) on page 116. The result can be seen in Figure 9.7(a), where the red dot marks the

trajectory which uses the least amount of energy, thus it is the final trajectory of torso in the sagittal plane.



(a) Power consumption (b) PoS and x_{ZMP}

Figure 9.7: *9.7(a) shows the power consumption of the trajectories having the best stability. The red dot marks the one using the least amount of power. 9.7(b) shows the trajectory of x_{ZMP} during two walking cycles, this is plotted together with the PoS during the two cycles.*

In Figure 9.7(b) the PoS in the x-direction is plotted together with the trajectory of x_{ZMP}. This is the result of the simulation where the final trajectories were used. It can be seen that the ZMP is inside the PoS at all times, except at initial value. The spikes that can be seen in Figure 9.7(b) result from phase shifts in the model. The final values of the varied parameters are:

$$\begin{aligned} x_{\text{t,b}} &= 35.2\,\text{mm} \\ x_{\text{t,e}} &= 65.5\,\text{mm} \end{aligned} \tag{9.23}$$

9.3.2 Finding Parameters for Frontal Movement

The same procedure as for sagittal movement is applied for the frontal. The parameter $y_{\text{t,max}}$ is varied within the ranges specified in Equation (9.18) on page 121, in steps of $0.01 y_{\text{t,mid}}$. This makes a total of 61 simulations.

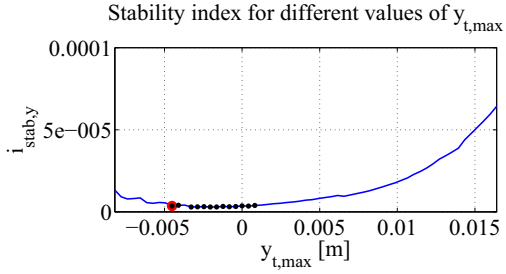
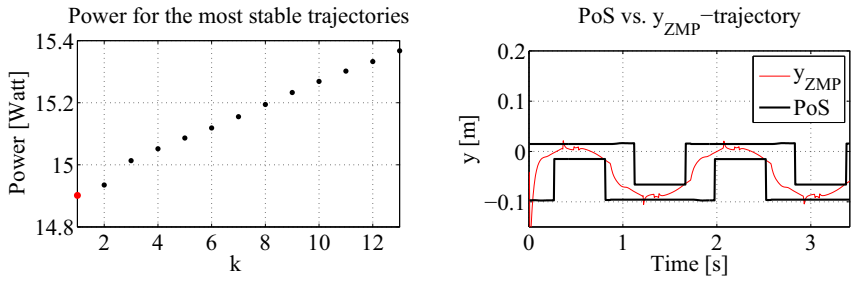


Figure 9.8: *The stability index shown as a function of $y_{t,max}$. The black dots, mark the trajectories which are compared on energy consumption. The red dot marks the final trajectory.*

For each simulation the stability index, $i_{\text{stab,y}}$, is calculated as in Equation (9.2) on page 115, the result is shown in Figure 9.8. The 20 % having the smallest stability index, those marked with a black dot, are compared on power consumption, which is done using Equation (9.4). The 20 % is chosen to get a sufficient representation of

the simulations. The resulting power consumption for the considered trajectories can be seen in Figure 9.9(a). Opposite what was expected, it is actually the trajectories that moves the torso the most, that results in the smallest power consumption. This is properly because these trajectories places the torso in a position, where the joints are not as stressed. The final value of $y_{\text{t,max}}$ is:

$$y_{\text{t,max}} = -4.5\,\text{mm} \tag{9.24}$$



(a) Power consumption

(b) PoS and y_{ZMP}

Figure 9.9: *9.9(a) shows the power consumption of the trajectories having the best stability. The red dot marks the one using the least amount of power. 9.9(b) shows the trajectory of x_{ZMP} during two walking cycles, which is plotted together with the PoS during the two cycles.*

The motion of the ZMP in the y-direction, y_{ZMP}, can be seen together with the PoS in Figure 9.9(b), when the final trajectories of the torso are applied to the model.

9.4 Resulting Trajectories

In this section the resulting polynomials, describing the trajectories, will be presented.

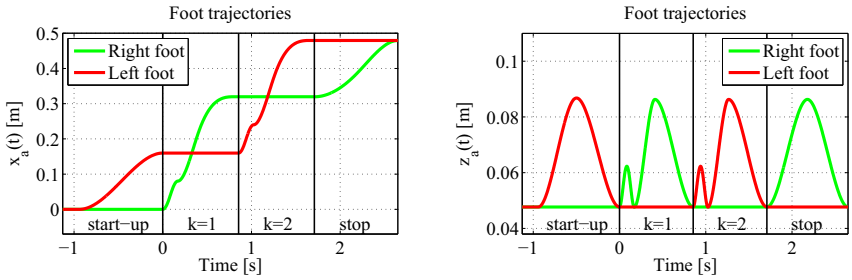
9.4.1 Polynomials for Feet

The trajectories for the feet, during the start up, one walking cycle and the stop phase can be seen in Figure 9.10. Note that one walking cycle consists of one step on the right foot, and one on the left foot.

The polynomial description of the developed trajectories for the feet can be found in Appendix I.3.1 on page 248. The trajectories for $x_{\text{a}}(t)$ and $z_{\text{a}}(t)$ can be combined into a trajectory in the sagittal plane which is shown in Figure 9.11.

9.4.2 Polynomials for Torso

The trajectories for the torso can be seen in Figure 9.12, where the chosen trajectories are those plotted as the blue line labeled 'Torso'.



(a) Trajectories for x-direction (b) Trajectories for z-direction

Figure 9.10: *Trajectories for the right and left foot for start-up, one walking cycle $k = [1, 2]$ and the stop phase. 9.10(a) shows the trajectories in x-direction, and 9.10(b) shows the trajectories in z-direction.*

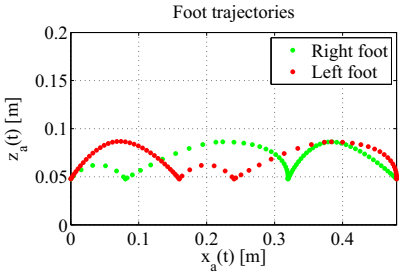
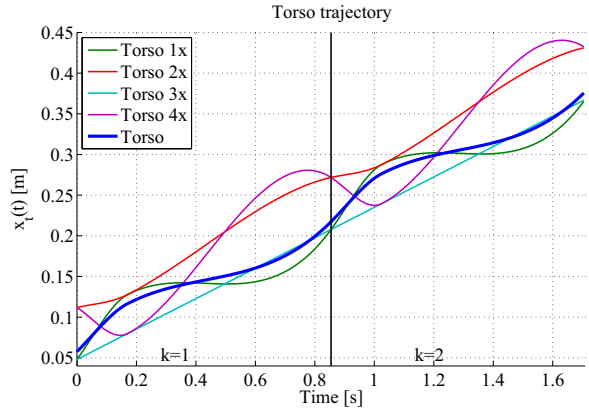
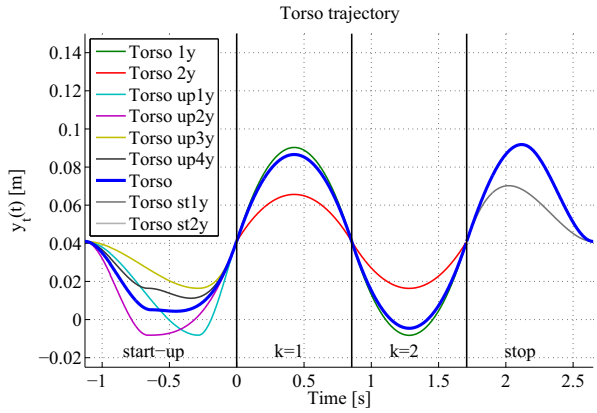


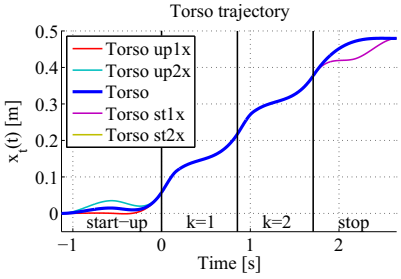
Figure 9.11: *Trajectories for the right and left foot in the sagittal plane for start-up, one walking cycle and the stop phase. To emphasize the time dependency, the trajectories are plotted as dotted lines, with* 25 ms *between each dot.*



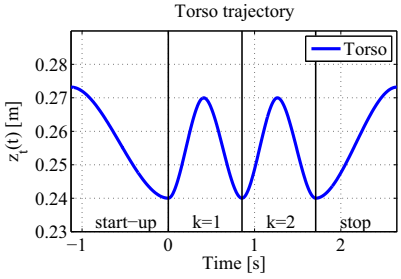
(a) Trajectory for x-direction



(b) Trajectory for y-direction



(c) Trajectory for x-direction



(d) Trajectory for z-direction

Figure 9.12: *Trajectories for the torso, for the start-up phase, the following walking cycle $k = [1, 2]$, and the stop phase. 9.12(a) shows the trajectory in the x-direction, for different values of $x_{t,b}$ and $x_{t,e}$. 9.12(b) shows the trajectory in y-direction, for different values of $y_{t,max}$, $y_{t,up}$, $T_{y,up}$ and $y_{t,stop}$. 9.12(c) shows the trajectories in x-direction for different values of the start-up parameter $x_{t,up}$ and the stop parameter $x_{t,stop}$. 9.12(d) shows the trajectory in z-direction, $z_t(t)$. In all figures, the wider blue line represents the chosen trajectory.*

Additionally Figure 9.12(a) shows some of the tested trajectories for the exterior values of $x_{\text{t,b}}$ and $x_{\text{t,e}}$. Specifically:

$$\begin{array}{lll} \text{Torso 1x} : & x_{\text{t,b}} = 0.2l_{\text{step}} & , \quad x_{\text{t,e}} = 0.3l_{\text{step}} \\ \text{Torso 2x} : & x_{\text{t,b}} = 0.2l_{\text{step}} & , \quad x_{\text{t,e}} = 0.7l_{\text{step}} \\ \text{Torso 3x} : & x_{\text{t,b}} = 0.5l_{\text{step}} & , \quad x_{\text{t,e}} = 0.3l_{\text{step}} \\ \text{Torso 4x} : & x_{\text{t,b}} = 0.5l_{\text{step}} & , \quad x_{\text{t,e}} = 0.7l_{\text{step}} \end{array}$$

Figure 9.12(b) also includes some of the tested trajectories, specifically for the exterior values of $y_{\text{t,max}}$, $y_{\text{t,up}}$, $T_{\text{y,up}}$ and $y_{\text{t,stop}}$:

$$\begin{array}{lllll} \text{Torso 1y} : & y_{\text{t,max}} = & -0.2y_{\text{t,mid}} & & \\ \text{Torso 2y} : & y_{\text{t,max}} = & 0.4y_{\text{t,mid}} & & \\ \text{Torso up1y} : & y_{\text{t,up}} = & -0.2y_{\text{t,mid}} & , & T_{\text{y,up}} = 0.3T_{\text{init}} \\ \text{Torso up2y} : & y_{\text{t,up}} = & -0.2y_{\text{t,mid}} & , & T_{\text{y,up}} = 0.7T_{\text{init}} \\ \text{Torso up3y} : & y_{\text{t,up}} = & 0.4y_{\text{t,mid}} & , & T_{\text{y,up}} = 0.3T_{\text{init}} \\ \text{Torso up4y} : & y_{\text{t,up}} = & 0.4y_{\text{t,mid}} & , & T_{\text{y,up}} = 0.7T_{\text{init}} \\ \text{Torso st1y} : & y_{\text{t,stop}} = & 1.6y_{\text{t,mid}} & & \\ \text{Torso st2y} : & y_{\text{t,stop}} = & 2.2y_{\text{t,mid}} & & \end{array}$$

And finally Figure 9.12(c) shows the trajectories for the exterior values of $x_{\text{t,up}}$ and $x_{\text{t,stop}}$, specifically:

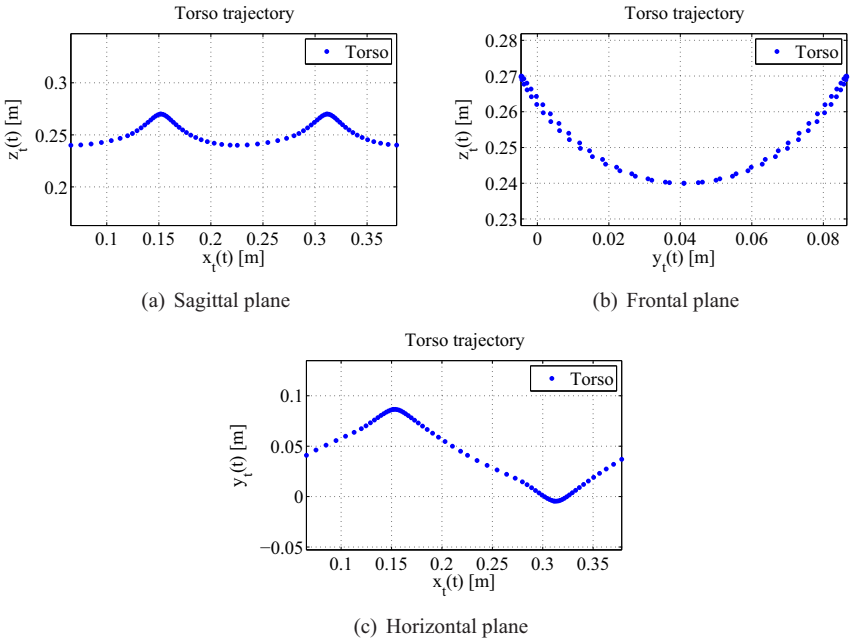
$$\begin{array}{ll} \text{Torso up1x} : & x_{\text{t,up}} = 0 \\ \text{Torso up2x} : & x_{\text{t,up}} = 0.6(x_{\text{t,e}} - l_{\text{step}}) \\ \text{Torso st1x} : & x_{\text{t,stop}} = 0.65l_{\text{step}} \\ \text{Torso st2x} : & x_{\text{t,stop}} = 0.95l_{\text{step}} \end{array}$$

The polynomial description of the torso trajectories can be found in Appendix I.3.2 on page 249. The trajectories for $x_{\text{t}}(t)$, $z_{\text{t}}(t)$ and $y_{\text{t}}(t)$ can be combined into a trajectories in the sagittal, frontal and horizontal plane which are shown in Figure 9.13(a), 9.13(b) and 9.13(c) respectively.

9.5 Conclusion on Trajectory Generation

The trajectories described in the this chapter are tested on the real system. As expected the robot was not able to maintain balance, when only given these trajectories in a feed forward manner, and with no sensor feedback. However during the test another crucial property of the system was observed. The joints were moving in steps; they quickly moved to the reference given to them, and then stopped until the next reference was given. This unwanted behavior resulted in very large accelerations/decelerations making the ZMP jumping back and forth. This is due to the relative low update frequency of the system of $40\,\text{Hz}$. This observation can be supported by a simulation of the system, where the trajectories are clocked in with a frequency of $40\,\text{Hz}$, the resulting ZMP can be seen in Figure 9.14.

There is a number of possible solutions for this problem;



(a) Sagittal plane

(b) Frontal plane

(c) Horizontal plane

Figure 9.13: *Trajectory for the torso shown in the three planes, for one walking cycle* $k = [1, 2]$. *To emphasize the time dependency, the trajectory is plotted as a dotted line, with* $25\,\mathrm{ms}$ *between each dot.*

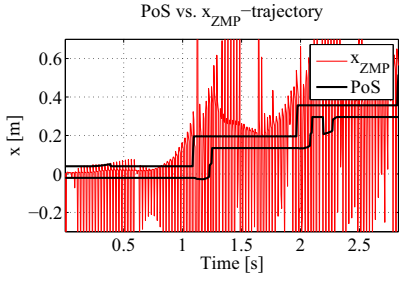


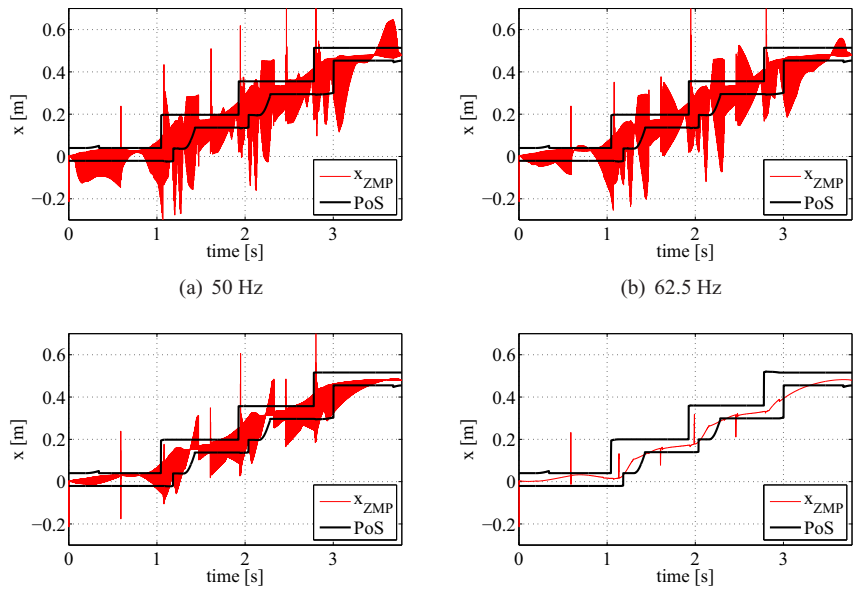
Figure 9.14: *The figure shows how the* x_{ZMP} *behaves, when the trajectories are clocked in with a frequency of* $40\,\mathrm{Hz}$.

Increase the update frequency of the system: Increasing the update frequency of the system would however not remove this problem, since the servo motors are controlled with a standard input, as described in Section 5.3.2, that only allows one to control them with a position reference given every $20\,\mathrm{ms}$. Hence it is only possible to control the servo motors with a maximum frequency of $50\,\mathrm{Hz}$, and a faster system would therefore not remove this problem.

Change trajectories: The trajectories can be changed such that the distance between the reference points are smaller. This way the influence on the stability due to this unwanted feature of the actuators can be minimized.

Acquire new actuators: The only way to completely remove the problem is to acquire new actuators, that provides the possibility of velocity control. Such that they can be controlled more smooth. This solution is however not considered an option, since new actuators are very expensive, and since implementation of for instance DC-motors with a velocity controller would be very time demanding.

Modify the actuators: The given actuators could be modified, by either removing the internal print and create a new or by reprogramming the controller already placed in the servo. This way the communication standard could be changed, and a faster control could be achieved. In order to find out how fast the control should be, a simulation has been performed, where the control frequency has been changed. The result of this simulation is seen in Figure 9.15, where it is seen that the control frequency should be around $250\,\mathrm{Hz}$.

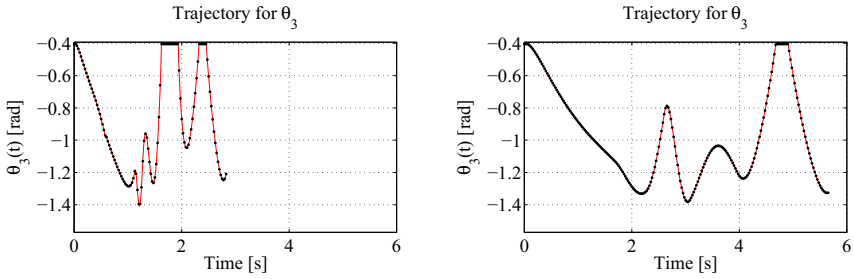


(a) 50 Hz (b) 62.5 Hz

(c) 125 Hz (d) 250 Hz

Figure 9.15: *The figures show the ZMP, with four different control frequencies of the servo motor model.*

The only feasible solution, if dynamical walk is to be obtained with the biped robot, is to alter the trajectories. The trajectories can either be altered by decreasing the walking speed or by imposing a limit on the acceleration, in order to smoothen out the trajectories. It is decided to decrease the walking speed, since this will result in more reference points during each step, and a more smooth motion can be obtained. The method of approach is exactly the same as described in this chapter, and the derivation of the trajectories will therefore not be examined again. The final trajectories are presented in Appendix I.4. From a number of simulations it was decided to decrease the walking speed with one fourth, such that the new walking speed is now, $v_{\text{walking}} = 9.35\,\text{cm/s}$. Figure 9.16 shows an example of the new trajectories, it shows the trajectory of joint 3, where the black dots marks the reference points given to the servo motor placed in this joint. Figure 9.16(a) shows the trajectory before altering it, and Figure 9.16(b) shows the trajectory after it has been altered.

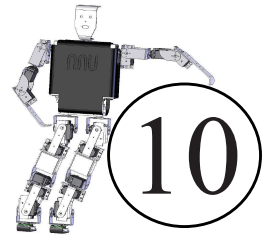


(a) $v_{\text{walking}} = 37.4\,\text{cm/s}$ (b) $v_{\text{walking}} = 9.35\,\text{cm/s}$

Figure 9.16: *The figures show the trajectory of θ_3, where Figure 9.16(a) shows the fast trajectory made in the first iteration, and Figure 9.16(b) shows the slow trajectory made in the second iteration. The black dots marks the reference points given to the servo placed in joint 3, and Figure 9.16(b) clearly shows that the distance between the each point is decreased. The showed trajectory is for the start-up phase, and the following walking cycle.*

It is clearly seen that the distance between the reference points given to the actuator has been reduced. The new trajectories are tested on the real system, and though the biped robot is still not able to maintain balance the movement is now much more smooth. The next step toward making the biped robot walk, will be to use the information from the sensors placed on the robot, in order to control the movement, and thereby maintain balance. This will be considered in the next chapter.

CONTROLLING THE BIPED ROBOT



10

Chapter contents

THIS CHAPTER *describes the development and implementation of two controllers for the biped robot. Two controllers are designed and implemented, each with a specific purpose. A hybrid posture controller controls the attitude of the torso and a ZMP controller controls the position of the ZMP. The design of a third controller, the inverse ZMP controller is proposed and the method is discussed.*

So far, a complete model and an inverse kinematic model has been derived. The model was used to calculate dynamical walking trajectories based on human gait data. The walking trajectories were improved using the ZMP stability, that is a measure of how stable the gait is along the walking path. Due to backlash, model uncertainties and external disturbances, it will not be possible to perform dynamic walk by feeding the trajectories directly to the inverse kinematic model and then to the robot, hence a controller must be implemented.

When studying control of biped robots, several different approaches are seen. In [Christensen et al., 2006] a simple fuzzy controller is implemented on a 21 DoF robot in order to maintain balance if the robot is exposed to smaller disturbances. This fuzzy controller made it possible to set up a simple, but large, set of control rules, enabling smaller disturbances to be suppressed, even though the control input was unreliable. A large disturbance controller, that was capable of avoiding a fall by taking a step in the correct direction, was also implemented and tested. Only static walk was achieved in [Christensen et al., 2006].

In [Takahashi et al., 2005] two controllers are implemented on a 5 DoF biped robot in order to achieve dynamic walk by using CPG. To ensure a correct angle of all the joints, a PD controller that controls the torque in each joint is implemented. A CPG controller is implemented to control the leg movements in order to maintain balance. The CPG controller, that also generates the walking trajectories, also uses the torque in each joint as control variable. In systems where the torque can be measured, torque control has an advantage if energy efficient walk is desired, since the torque in each joint can be converted directly to energy.

Another control approach is to use the angular acceleration as control input and by using an inverse dynamics model convert this to a torque in each joint. This approach is seen in [Kondak and Hommel, 2003], where a 5 DoF biped robot is simulated. Here, two controller types are implemented. Three inner PD controllers are used to maintain the correct position of the servos. The inputs to these controllers are the desired position of the joints and the output is the angular acceleration. A second controller, a non-linear controller, ensures that the movements stay within the stability region. This controller converts the angular acceleration to joint torques and apply these to each motor. The output of the system is the angular velocity and position, which is used for the inner control loops as feedback.

An interesting control approach is seen in [Bachar, 2004] where the inverse Jacobian, describing the change of the ZMP based on the change of the generalized coordinates, is calculated for the model of a 22 DoF biped robot. This inverse Jacobian, can then be used to move the ZMP in a desired direction. The thesis describes how a ZMP counter balance controller is calculated and used to regain balance, if exposed to an external disturbance. This method seems very applicable since it would be possible to ensure stability if the reference ZMP is set to be inside the PoS. Unfortunately only simulations are carried out in the mentioned thesis.

10.1 Control Design Approach

It has been chosen to design two controllers in this project; a ZMP controller and a hybrid posture controller. Furthermore, the design of an inverse ZMP controller is proposed. The overall control strategy can be seen from Figure 10.1, where the implementation of both the controllers is seen.

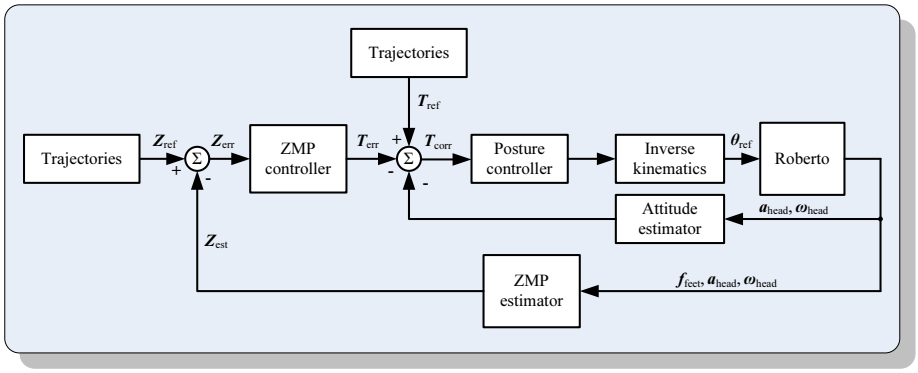


Figure 10.1: *Overview of the control strategy used in the project.*

The input to the control system is the trajectories calculated based on the model. Not only the walking trajectories are used, but also the ZMP trajectory. Using these the robot should perform human-like walk. But as the real system do not behave in the exactly same manner as the model, certain disturbances should be expected. These disturbances can be seen as a disturbance in the ZMP, and are canceled using a ZMP controller. It is observed that the input trajectories can not be completely held by the robot due to backlash in the system. Because of the weight of the torso, the upright posture needed to perform human-like walk is not kept. Therefore a posture controller is introduced, which has the purpose to keep the desired posture of the torso. This controller can be seen as a fast inner control-loop which main purpose is to ensure that the posture is correct at all times during the gait.

Further it can be seen from Figure 10.1 that the input signal to the controllers have to be the attitude of the torso and the position of the ZMP. But as these signals are not directly available from the sensors, they are estimated using a ZMP estimator and an attitude estimator.

Next the estimation of the control input signals is explained and the design of the controllers are explained after that.

10.2 Estimation of Control Input

As control inputs, four different measurements are available. These are: Pressure measurements from the feet, acceleration and angular velocity of the head, and position feedback from the servo motors in the legs. A description of these is found in Chapter 5 on page 33. The following will describe how the control signals are generated.

An overview of the different estimators used throughout the project is seen from Figure 10.2. It is seen that as the system to be controlled can be considered hybrid a phase estimator has to be constructed as well. The phase of this estimator is input to the other estimators, since they are dependent on the current phase.

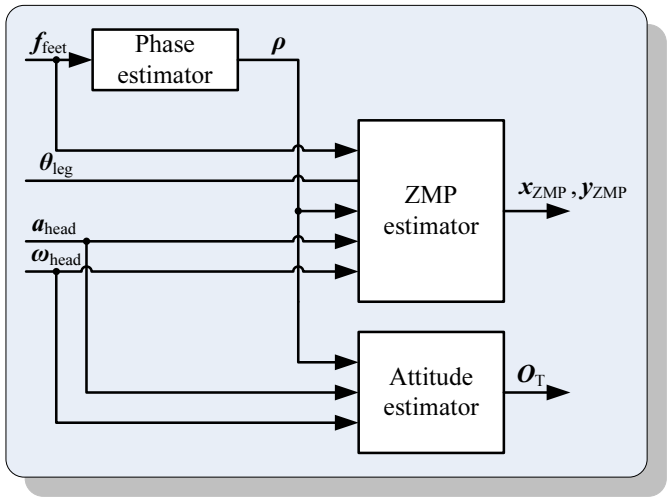


Figure 10.2: *Overview of the estimators used in the project.*

10.2.1 Phase Estimation

The phase is estimated using the pressure measurements. The input to the phase estimator is $\boldsymbol{f}_{\text{feet}}$ and the output is an estimate of the phase. This can be seen from Figure 10.3.

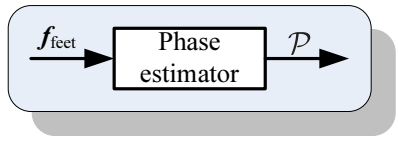


Figure 10.3: *Input and output of the phase estimator.*

The robot has three discrete phases, DSP, SSP-R, and SSP-L, as described in Section 7.5.1 on page 88. In order to use the proper controller it is necessary to determine which of the phases $\{\rho_1, \rho_2, \rho_3\} = \{\text{SSP-R}, \text{DSP}, \text{SSP-L}\}$ the system is in. The estimator uses the pressure measurements to determine the phase, which is done the following way:

$$\mathcal{P} = \begin{cases} \rho_1 & \text{if} \quad \sum_i f_{\text{i,R}} > \mu_{\text{limit}} \wedge \sum_i f_{\text{i,L}} < \mu_{\text{limit}} \\ \rho_2 & \text{if} \quad \sum_i f_{\text{i,R}} > \mu_{\text{limit}} \wedge \sum_i f_{\text{i,L}} > \mu_{\text{limit}} \\ \rho_3 & \text{if} \quad \sum_i f_{\text{i,R}} < \mu_{\text{limit}} \wedge \sum_i f_{\text{i,L}} > \mu_{\text{limit}} \end{cases} \tag{10.1}$$

where μ_{limit} is set to $2\,\text{N}$. In Equation (10.1) it can be seen that if none of the feet are exposed to a force larger than $2\,\text{N}$ then the phase can not be determined. Given the phase of the system, the ZMP estimator can be developed, which is done next.

10.2.2 ZMP Estimation

The inputs and outputs of the ZMP estimator are seen from Figure 10.4. Given the available sensor measurements on the system two methods for estimating the ZMP are investigated.

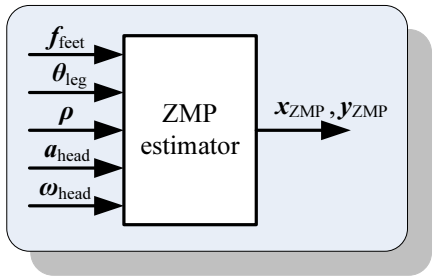


Figure 10.4: *Input/output relation of the ZMP estimator.*

The first method estimates the ZMP using the output from the IMU, namely the acceleration and the angular velocity. Another method investigated is estimation of the ZMP using the pressure measurements.

ZMP Estimation Using the IMU

In the developed model the ZMP is determined using Equation (2.5) and (2.6) on page 9. This approach is not feasible on the robot since the linear and the angular acceleration of all links have to be determined. The developed model could be used to estimate the accelerations, but it is however not feasible to implement the developed model on the real system due to the relatively low processing power. Instead another property of the robot is exploited. The weight of the robot is $3.7\,\text{kg}$ in total and of this, the torso weighs $1.2\,\text{kg}$, that is, 32.4% of the total weight is in the torso meaning the ZMP is mostly determined by the torso. This has resulted in the assumptions seen in Equation (10.3) to (10.5).

$$x_{\text{ZMP,m}} \approx x_{\text{ZMP,t}} \tag{10.2}$$

$$\frac{\sum_{i=1}^{22} m_i g_z x_i - \sum_{i=1}^{22} m_i \ddot{x}_i z_i}{\sum_{i=1}^{22} m_i g_z} \approx \frac{g_z x_t - \ddot{x} z_t}{g_z} \tag{10.3}$$

$$y_{\text{ZMP,m}} \approx y_{\text{ZMP,t}} \tag{10.4}$$

$$\frac{\sum_{i=1}^{22} m_i g_z y_i - \sum_{i=1}^{22} m_i \ddot{y}_i z_i}{\sum_{i=1}^{22} m_i g_z} \approx \frac{g_z y_t - \ddot{y} z_t}{g_z} \tag{10.5}$$

To verify if these assumptions are valid the walking trajectories, developed in Chapter 9 on page 109, are used as input to the complete model where the simplified ZMP from Equation (10.3) and (10.5) is implemented. The result can be seen in Figure 10.5.

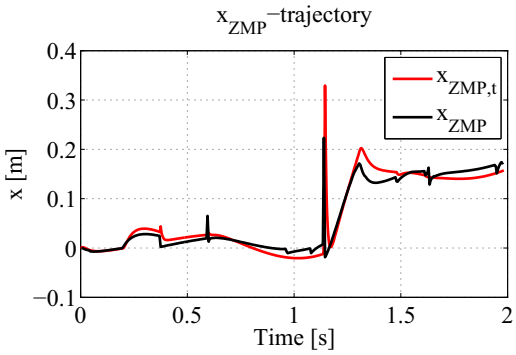


Figure 10.5: *Simplified ZMP estimated comparred with real ZMP.*

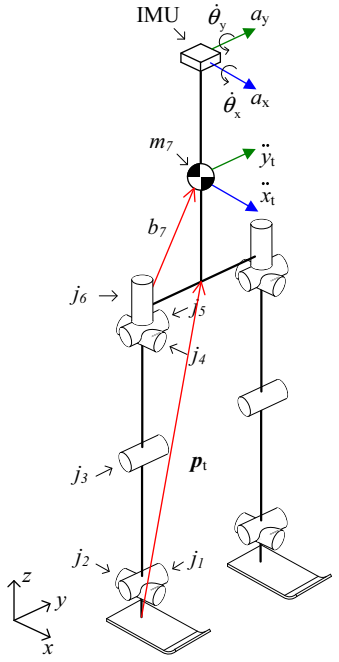


Figure 10.6: *The ZMP is determined using only the position and the acceleration of the torso CoM.*

As it can be seen from the figure, the simplified ZMP only differs slightly from the true ZMP. It therefore seems reasonable to use the simplified ZMP on the robot. Inspecting Equation (10.3) and (10.5) it is seen that the positions x_t, y_t, and z_t have to be determined along with the accelerations $\ddot{x}_t$ and $\ddot{y}_t$. The accelerations that are measured by the IMU have to be related to the acceleration of the CoM of the torso, m_7. Figure 10.6 shows the direction of the accelerations of m_7, $\ddot{x}_t$ and $\ddot{y}_t$, along with the direction of the accelerations that can be measured by the IMU, a_x and a_y. From the figure it can be seen that the orientation of the IMU is required to transfer a_x and a_y to $\ddot{x}_t$ and $\ddot{y}_t$. First the gravity acceleration that is present in the measurement is removed based on the orientation of the IMU:

$$\ddot{x}_t = a_x - g\sin(\theta_y) \tag{10.6}$$
$$\ddot{y}_t = a_y - g\sin(\theta_x) \tag{10.7}$$

Now the acceleration given in the IMU is rotated, such that the accelerations are given with respect to the base frame using Equation (10.8):

$$\boldsymbol{a}_t = \begin{bmatrix} \ddot{x}_t \\ \ddot{y}_t \\ \ddot{z}_t \end{bmatrix} = {}^0_6\boldsymbol{R}\,{}^6\boldsymbol{a}_t \ , \ \text{for } \rho = \{\rho_1, \rho_2\} \tag{10.8}$$

And for the SSP-L it is given by Equation (10.9):

$$\boldsymbol{a}_t = \begin{bmatrix} \ddot{x}_t \\ \ddot{y}_t \\ \ddot{z}_t \end{bmatrix} = {}^{12}_{7}\boldsymbol{R}\,{}^7\boldsymbol{a}_t \ , \ \text{for } \rho = \{\rho_3\} \tag{10.9}$$

What remains from Equation (10.3) and (10.5) is to determine the positions x_t, y_t, and z_t. From Figure 10.6 it can be seen that these can be determined by the kinematic chain for the supporting leg described by Equation (10.10):

$$\begin{bmatrix} \boldsymbol{p}_t \\ 1 \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix} = {}^0_6\boldsymbol{T} \begin{bmatrix} {}^6\boldsymbol{b}_7 \\ 1 \end{bmatrix} \ , \ \text{for } \rho = \{\rho_1, \rho_2\} \tag{10.10}$$

Equation (10.10) is only valid in SSP-R and in DSP. In SSP-L the kinematic chain for the left leg is used:

$$\begin{bmatrix} \boldsymbol{p}_t \\ 1 \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ z_t \\ 1 \end{bmatrix} = {}^{12}_{7}\boldsymbol{T} \begin{bmatrix} {}^7\boldsymbol{b}_7 \\ 1 \end{bmatrix} \ , \ \text{for } \rho = \{\rho_3\} \tag{10.11}$$

Since the phase, $\mathcal{P}$, is required to determine if the ZMP, the estimator becomes a hybrid estimator.

ZMP Estimation Using the Pressure Sensors

In Section 2.3.3 on page 10 it was stated that if the ZMP is within the PoS then the CoP is the same as the ZMP. CoP can be directly measured using the pressure measurements in the feet. Therefore Equation (2.9) and (2.10), from Section 2.3.3 on page 10 are used for SSP and DSP respectively, which are repeated here, in Equation (10.13), for convenience:

$$\boldsymbol{p}_{\text{ZMP,SSP}} = \frac{\sum_{i=1}^{3} f_i \boldsymbol{r}_{i\text{f}}}{\sum_{i=1}^{3} f_i} \tag{10.12}$$

$$\boldsymbol{p}_{\text{ZMP,DSP}} = \frac{\left(\sum_{i=1}^{3} f_{i\text{r}}\right)\boldsymbol{p}_{\text{CoP,r}} + \left(\sum_{i=1}^{3} f_{i\text{l}}\right)\boldsymbol{p}_{\text{CoP,l}}}{\sum_{i=1}^{3} f_{i\text{l}} + \sum_{i=1}^{3} f_{i\text{r}}} \tag{10.13}$$

Verification of the ZMP Estimators

The two ZMP estimators have both been implemented and tested on the robot. The verification is found in Appendix J on page 255. It is seen that the ZMP estimate calculated using the pressure sensors provide a usable ZMP estimate. However, due to the small feet, the ZMP estimate is very dependent on the location and position of the feet.

The ZMP estimate calculated on behalf of the accelerometer and the servo motor position feedback, is however not reliable. Due to noise in the accelerometer measurements and noise in the potentiometer measurements, the ZMP estimate does not reach a steady state value, and is furthermore dominated by noise. It is therefore concluded that this ZMP estimator cannot be used as a control input.

10.2.3 Attitude Estimation

In this section the attitude estimator is described. The inputs and output of the attitude estimator are seen from Figure 10.7.

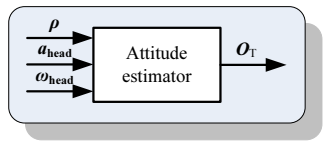


Figure 10.7: *Inputs and output of attitude estimator.*

In order to control the posture of the robot, the orientation of the torso has to be determined. This section seeks to develop a reliable way of estimating the orientation of the torso using the IMU. The function used to calculate the error of the estimate is shown in Equation (10.14).

$$R^2 = 1 - \frac{\int \left(y_{\text{real}}(t) - y_{\text{model}}(t)\right)^2 dt}{\int \left(y_{\text{real}}(t) - \mu_{\text{real}}\right)^2 dt} \tag{10.14}$$

The section will start with describing the different methods investigated for attitude determination including a comparison with data from the real system. The performance is evaluated using the error function given in Equation (10.14) and the variance. The attitude is tried estimated using the gyroscope, the accelerometer and a combination of them both. First a test setup, which is used in all three cases, is explained.

IMU Test Setup

In order to test the performance of the attitude estimation, the IMU is subjected to a series of tests. Figure 10.8 shows the test setup. As it can be seen on the test setup the IMU is displaced from the rotation axis. This introduces accelerations on the *x*- and *y*-axis, besides the gravity force, which is a disturbance. Such disturbances will no doubt be present on the robot when it walks. The rotation is accomplished using a HS645MG servo motor. Four series of measurements are conducted, which each subjects the IMU to rotations at different frequencies. The inputs used in the test series are listed in Equation (10.15) to (10.18).

$$\theta_{\text{test1}}(t) = \frac{20\pi}{180}\sin(1t)\ [\text{rad}] \tag{10.15}$$

$$\theta_{\text{test2}}(t) = \frac{20\pi}{180}\sin(2t)\ [\text{rad}] \tag{10.16}$$

$$\theta_{\text{test3}}(t) = \frac{20\pi}{180}\sin(3t)\ [\text{rad}] \tag{10.17}$$

$$\theta_{\text{test4}}(t) = \frac{20\pi}{180}\sin(4t)\ [\text{rad}] \tag{10.18}$$

During the four tests the acceleration and the angular velocity, measured by the IMU are recorded. The tests only introduce a rotation about the *x*-axis. The data sheet for the gyro, [InvenSense Inc, 2006], states that the *x*- and *y*-axis will yield the same result if subjected to the same rotation which makes it enough to only have data from one of these axes.

Attitude Estimation Using the Gyroscope

The IMU is equipped with an IDG-300 gyroscope (gyro) which measures the angular velocity about two axes. The gyro is placed such that the angular velocity about the *x*- and *y*-axis are measured. Since the signal from the gyroscope is the angular velocity, it has to be integrated in order to give an inclination. The data sheet [InvenSense Inc, 2006] states the sensitivity of the gyro to be $s_{gyro} = 2\,\text{mV/deg/s}$. Converting the measured voltage signal, v_{gyro} can be done using Equation (10.19).

$$\theta_{\text{g}} = \int_0^\infty (v_{\text{gyro}} - 1.5\,\text{V})\frac{1}{\text{s}_{\text{gyro}}}\text{dt}[\text{deg}] \tag{10.19}$$

Figure 10.9 shows the results from test 1 to 4, where the signal is integrated to get θ_{g}. The tests show drift in θ_{g} which is caused by bias in the measurement that, when

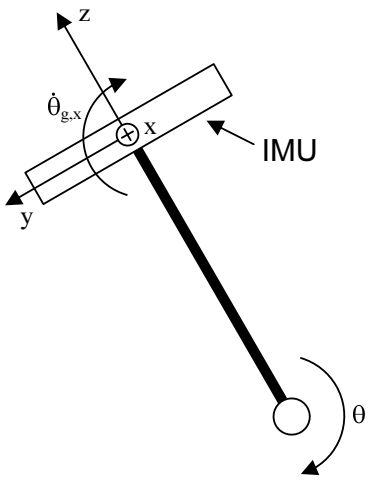


Figure 10.8: *Test setup for recording data from the IMU.*

integrated, will lead to an unbounded error. Determining better initial values, so the bias is minimized, will lead to better performance of the gyro and will let the robot operate over a longer period of time before the accumulated error becomes to large. In [Hsiao et al., 2006] it was shown that, with carefully calibration, the attitude could be not determined reliably for more than an hour using a gyro. After that the gyro had to be recalibrated. Since the goal of the this project is to make the robot autonomous, it has to able to measure the attitude reliably over an infinite time period. It is therefore not a viable choice to estimate the attitude of the robot based on the gyro alone.

Attitude Estimation Using the Accelerometer

The accelerometer can be used to determine the attitude by measuring the magnitude of the gravity force imposed on the three axes. Using these three vectors the orientation of the torso can be determined using Equation (10.20) and (10.21).

$$\theta_{\text{a,x}} = \arctan\left(\frac{a_{\text{y,g}}}{a_{\text{z,g}}}\right)[rad] \tag{10.20}$$

$$\theta_{\text{a,y}} = \arctan\left(\frac{a_{\text{x,g}}}{a_{\text{z,g}}}\right)[rad] \tag{10.21}$$

Figure 10.10 shows the results from test 1 to 4 where Equation (10.20) has been used. Table 10.1 shows the error in the estimation. It can be seen from Table 10.1 that as the frequency increases so does the error. The IMU low-pass filters the accelerations through a first order filter with a cut-off frequency at $0.87\,\mathrm{rad/s}$ which distorts the measurements more as the frequency increases. The values of the components used for the passive low-pass filter is shown in [Spark Fun Electronics, 2006]. From Table 10.1 it can be seen that the accelerometer has a rather high R^2, Test 4 yielding the worst goodness of fit of 94.71. However, σ_{e}^2 is rather high which is caused by noise in the measurements. A possibility is to low-pass filter the measurements to reduce the noise, however, this will decrease the accuracy at high frequencies due to phase delay. Comparing the results from the accelerometer and the gyro makes the accelerometer a better choice for estimating the attitude since the error is bounded.

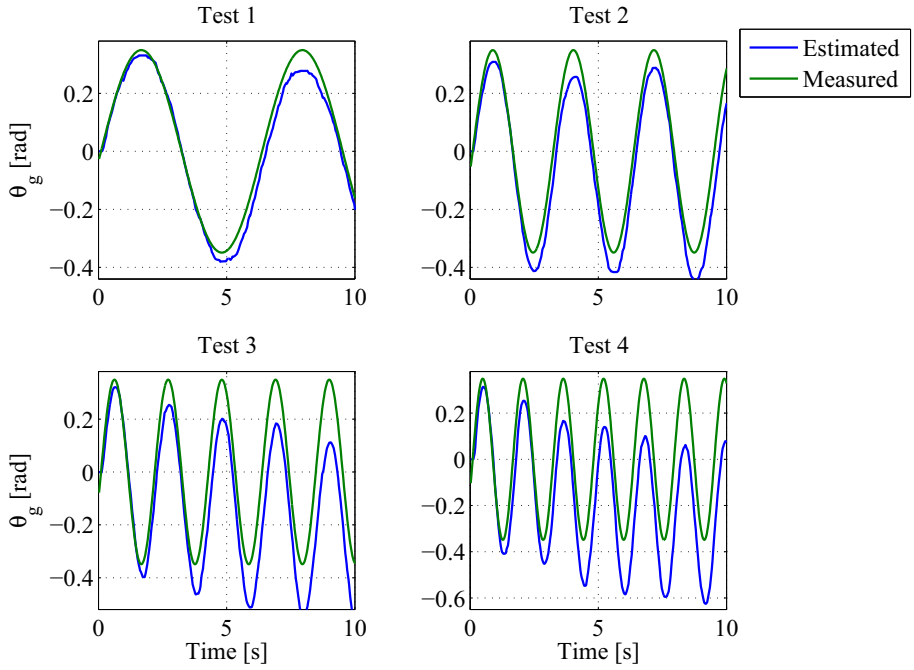


Figure 10.9: *The signal from the gyro has been integrated to get $\theta_{g,x}$. The green line shows the real orientation and the blue shows $\theta_{g,x}$.*

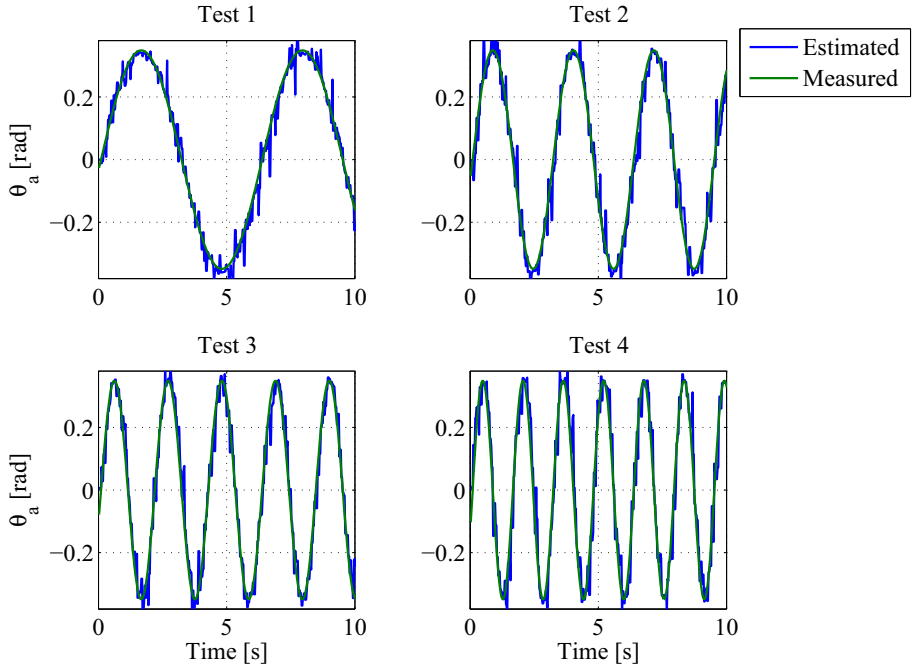


Figure 10.10: *The rotation about the x-axis estimated using the accelerometer.*

	R^2[%]	σ_e^2
Test 1	98.06 %	3.41
Test 2	96.45 %	6.87
Test 3	96.85 %	6.22
Test 4	94.71 %	10.57
Mean	**96.52 %**	**6.77**

Table 10.1: *Error when the accelerometer is used to estimate the attitude of the robot.*

Attitude Estimation Using Sensor Fusion

Extensive research has been conducted in fusing gyroscopes, accelerometers, and inclinometers to give a good estimate of the attitude of autonomous systems. In [Rehbinder and Hu, 2001] the measurements from a gyro and an accelerometer were fused using a modified Kalman filter. The approach made use of two models which were switched between based on the magnitude of the acceleration. During high acceleration the gyro was used and during low acceleration the accelerometer was used. The algorithm showed errors no greater than 0.6 deg when the accelerometer was used, however when the gyro was used the algorithm still showed drift in the estimation. Like [Rehbinder and Hu, 2001] other authors have shown in [Foxlin, 1996], [Rehbinder and Hu, 2000], and [Vaganay et al., 1993] various ways of combining measurements about the attitude using Kalman filters.

In [Baerveldt and Klang, 1997] the authors developed an attitude estimation system for an autonomous helicopter which fused an inclinometer and a gyro through a complementary filter. This approach showed an error of about $2\,\mathrm{deg}$ without drift. This approach was implemented succesfully in [Löffler et al., 2004] on a biped robot. In [Hsiao et al., 2006] the same approach was used but, instead of an inclinometer, an accelerometer was used. Since the computational power on the robot is limited it is decided to use the approach in [Hsiao et al., 2006]. Here in it was suggested that the measurements from the accelerometer and the gyro should only be used in the frequency domains where it can be considered ideal:

- The accelerometer uses the gravity force to determine the angle which means that accelerations caused by movement decreases the accuracy. Therefore the accelerometer should be considered ideal at low frequencies.
- Integrating the signal from the gyroscope will lead to unbounded drift due to bias in the signal. Therefore the gyroscope should be considered ideal at high frequencies.

To achieve the above [Baerveldt and Klang, 1997] suggested the use of a complementary filter with the following property:

$$H_a(s)G_a(s) + sH_g(s)G_g(s) \quad = \quad 1 \tag{10.22}$$

Here $H_a(s)$ and $H_g(s)$ are the sensor dynamics of the accelerometer and gyroscope respectively. $G_a(s)$ is a low-pass filter for the measurement from the accelerometer and $G_g(s)$ is a high-pass filter for the measurements from the gyro. The sensors are considered ideal, thus $H_g(s) = H_a(s) = 1$. Figure 10.11 shows the implemented

filter. [Baerveldt and Klang, 1997] suggested that $G_{\mathrm{a}}(s)$ be a first order low-pass filter in series with a lead filter and $G_g(s)$ be a second order high-pass filter:

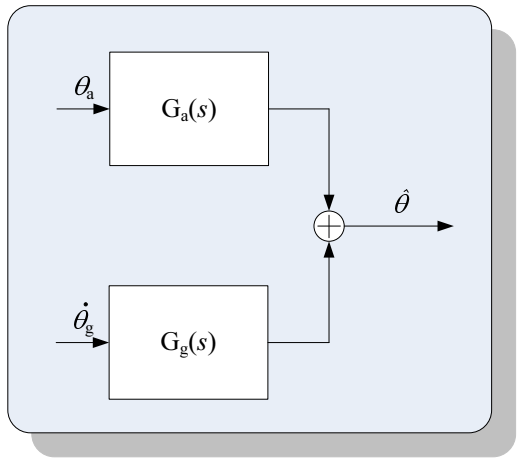


Figure 10.11: *Implemented filter, used for the inclination estimator.*

$$G_{\mathrm{a}}(s) = \frac{2\tau s + 1}{(\tau s + 1)^2} \tag{10.23}$$

$$G_{\mathrm{g}}(s) = \frac{\tau^2 s}{(\tau s + 1)^2} \tag{10.24}$$

Which gives the following filter for the estimator:

$$\hat{\theta}(s) = \frac{2\tau s + 1}{(\tau s + 1)^2} + s\frac{\tau^2 s}{(\tau s + 1)^2} \tag{10.25}$$

The complementary filter in Equation (10.22) only has a single design parameter, τ. τ should be chosen low enough to effectively remove the offset in the gyro and high enough not to filter, and thereby distort, the measurements from the accelerometer. The performance of the filter is evaluated using the R^2 performance function given in Equation (10.14) on page 141.

τ should be chosen so that Equation (10.14) is minimized which is done using the steepest descent method. It is assumed that Equation (10.14) has a global maximum. A value of τ that minimizes the error for test 1 to 4 is determined. τ_{i} is the value of τ that minimizes the error in test i. Table 10.2 shows the value of τ_{i} that minimizes the error in each of the tests along with the error and variance.

From Table 10.2 it can be seen that in order to minimize the error at high frequencies τ has to be smaller than at low frequencies. This means that that at high frequencies the gyro is trusted the most and at low frequencies the accelerometer is trusted the most. This supports the assumptions about where the accelerometer and the gyro should be considered ideal.

	$\tau_1 = 0.525$		$\tau_2 = 0.491$		$\tau_3 = 0.362$		$\tau_4 = 0.296$	
	$R^2[\%]$	σ_e^2	$R^2[\%]$	σ_e^2	$R^2[\%]$	σ_e^2	$R^2[\%]$	σ_e^2
Test 1	99.60	0.706	99.59	0.712	99.54	0.817	99.47	0.919
Test 2	99.39	1.187	99.40	1.160	99.21	1.526	98.96	2.019
Test 3	99.02	1.957	99.15	1.684	99.41	1.175	99.30	1.376
Test 4	97.41	5.228	97.65	4.708	98.42	3.149	98.59	2.823
Mean	**98.85**	**2.269**	**98.94**	**2.067**	**99.14**	**1.667**	**99.08**	**1.784**

Table 10.2: *Performance of the filter where the τ that minimizes R^2 has been determined using steepest descent. τ_i is the τ that minimizes R^2 in test i.*

The choice of τ can either be to minimize the error at a certain frequency or minimize the error over the frequency range recorded. If the latter, then τ should be 0.362 where the mean of R^2 is 99.14% and the variance 1.667 as can be seen in Table 10.2. This is chosen, since it is expected that the input will consist of a wide range of frequencies. Figure 10.12 shows the estimated inclination along with the actual inclination.

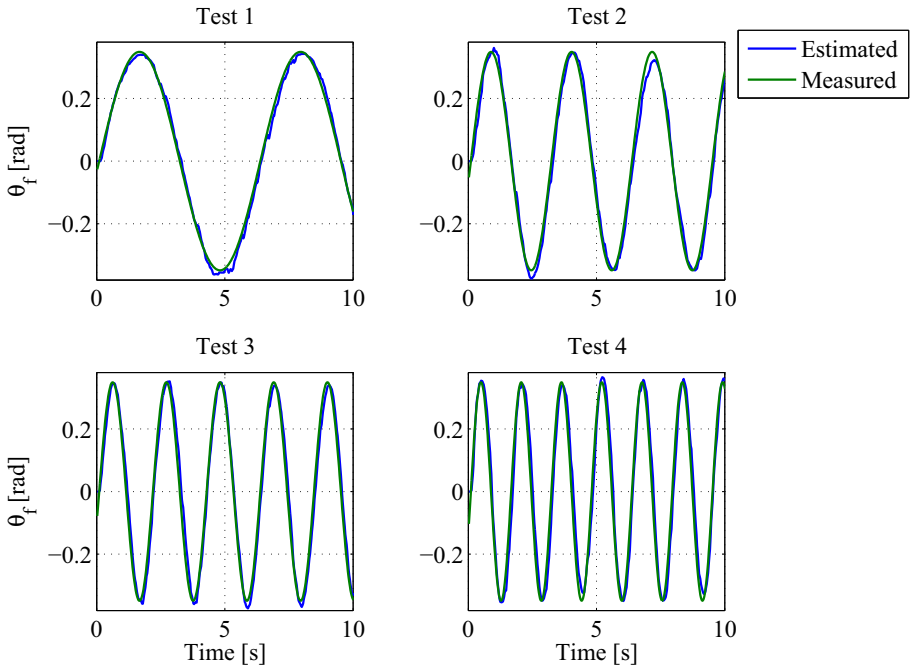


Figure 10.12: *Actual and estimated inclination using sensor fusion. Estimated inclination is shown as blue and the actual inclination is shown as green.*

10.3 Inverse ZMP Controller

The inverse ZMP controller could seem like the most optimal controller for this application. The basic idea of the inverse ZMP controller is to calculate how much the torso should be moved, in order to obtain the desired position of the ZMP. This is done by first calculating the Jacobian, being the matrix describing the rate of change of the

ZMP location with respect to a change of the generalized position, as given in Equation (10.26).

$$\boldsymbol{J}_{\text{ZMP}} \equiv \frac{\partial \boldsymbol{p}_{\text{zmp}}}{\partial \boldsymbol{q}} \tag{10.26}$$

Then, by calculating the inverse Jacobian, it is possible to calculate how the generalized positions should be changed, in order to achieve a desired ZMP displacement:

$$\Delta \boldsymbol{q} = \Delta \boldsymbol{p}_{\text{ZMP}} \boldsymbol{J}_{\text{ZMP}}^{-1} \tag{10.27}$$

Note that (10.27) only holds for small $\Delta \boldsymbol{p}_{\text{ZMP}}$. However, differentiating the ZMP with respect to the position of each links CoM would be a cumbersome task. To simplify the calculations, only the position and acceleration of the torso could be used. This simplification is possible, with minor deviation compared to the complete model of the robot, since most of the mass is located in the torso, as were seen in Figure 10.5. The displacement of the torso, when given a desired displacement in the ZMP, can now be found to be:

$$\Delta \boldsymbol{p}_{\text{torso}} \quad = \quad \Delta \boldsymbol{p}_{\text{ZMP}} \boldsymbol{J}_{\text{ZMP,t}}^{-1} \tag{10.28}$$

Where the Jacobian, $\boldsymbol{J}_{\text{ZMP,t}}$ can be calculated as:

$$\boldsymbol{J}_{\text{ZMP,t}} \equiv \frac{\partial \boldsymbol{p}_{\text{zmp}}}{\partial \boldsymbol{p}_{\text{torso}}} = \frac{\partial \begin{bmatrix} x_{\text{ZMP,t}} \\ y_{\text{ZMP,t}} \end{bmatrix}}{\partial \begin{bmatrix} x_{\text{torso}} \\ y_{\text{torso}} \end{bmatrix}} = \begin{bmatrix} \frac{\partial x_{\text{ZMP,t}}}{\partial x_{\text{torso}}} & \frac{\partial y_{\text{ZMP,t}}}{\partial x_{\text{torso}}} \\ \frac{\partial x_{\text{ZMP,t}}}{\partial y_{\text{torso}}} & \frac{\partial y_{\text{ZMP,t}}}{\partial y_{\text{torso}}} \end{bmatrix} \tag{10.29}$$

Equation (10.3) and (10.5) shows how the ZMP is calculated. Since y_{ZMP} is independent by changes of movement along the x-axis and since x_{ZMP} is independent by movements along the y-axis, Equation (10.29) can then be simplified to:

$$\boldsymbol{J}_{\text{ZMP,t}} = \begin{bmatrix} \frac{\partial x_{\text{ZMP,t}}}{\partial x_{\text{torso}}} & 0 \\ 0 & \frac{\partial y_{\text{ZMP,t}}}{\partial y_{\text{torso}}} \end{bmatrix} \tag{10.30}$$

Now, when calculating $\frac{\partial x_{\text{ZMP,t}}}{\partial x_{\text{torso}}}$ a problem arises. When calculating this, it is necessary to find the triple derivative of x_{torso}. This is seen in Equation (10.31).

$$\begin{aligned} \frac{\partial x_{\text{ZMP,t}}}{\partial x_{\text{torso}}} &= 1 - \frac{z_{\text{t}}}{g_{\text{z}}} \frac{\partial \ddot{x}_{\text{torso}}}{\partial x_{\text{torso}}} \\ &= 1 - \frac{z_{\text{t}}}{g_{\text{z}}} \frac{\partial \ddot{x}_{\text{torso}}}{\partial t} \frac{\partial t}{\partial x_{\text{torso}}} \\ &= 1 - \frac{z_{\text{t}}}{g_{\text{z}}} \dddot{x}_{\text{torso}} \dot{x}_{\text{torso}}^{-1} \end{aligned} \tag{10.31}$$

The same procedure is used for calculating $\frac{\partial y_{\text{ZMP,t}}}{\partial y_{\text{torso}}}$ and is therefore not shown. Due to noise in the measurement data, taking the derivative of the acceleration will not yield any useful information. The inverse ZMP controller will therefore not be implemented on the robot in this project.

10.4 ZMP Controller

In Chapter 9 on page 109 where the walking trajectories were calculated, a number of different trajectories were found, and a specific walking path was chosen based on the ZMP stability. To calculate the ZMP stability, the position of the ZMP was calculated along the walking path which was stable, if the ZMP trajectory was within PoS. The trajectory which had the largest stability margin was chosen. This ZMP trajectory is used as a reference input to the ZMP controller that is designed in this section.

The basic control concept is shown in Figure 10.13. In this figure the actual ZMP is estimated using the ZMP estimator explained in Section 10.2.2 on page 137. The ZMP estimate is subtracted from the ZMP reference trajectory which result in a position error of the ZMP, named $\mathbf{Z}_{\text{err}}$. This signal is fed into the ZMP controller, and the output, which is a correction of the reference trajectory, is converted into a displacement and inclination of the torso.

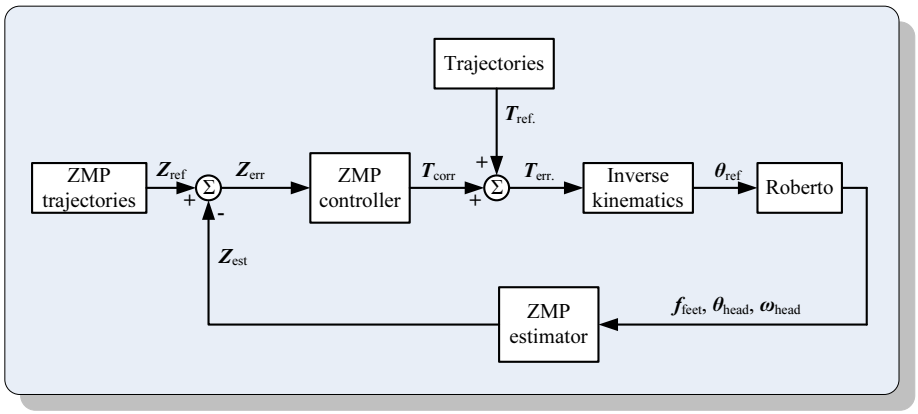


Figure 10.13: *Concept of the ZMP controller.*

The displacement and inclination of the torso is added to the actual reference trajectories before they are fed into the inverse kinematic model. The actual controller is a PI-controller, that is tuned using simulations and actual measurement data gathered during experiments. The ZMP controller is designed in the following section.

10.4.1 Designing the ZMP Controller

As mentioned in the previous section, the ZMP controller consists of a PI-controller. The basic structure of the PI-controller is shown in Figure 10.14.

Two PI-controllers have been implemented in the ZMP controller; one controlling the position of the torso along the y-axis, and one controlling the inclination around the y-axis. The first mentioned controller is capable of moving the ZMP along the y-axis and the last mentioned is capable of moving the ZMP along the x-axis. The implementation of the two is seen in Figure 10.15.

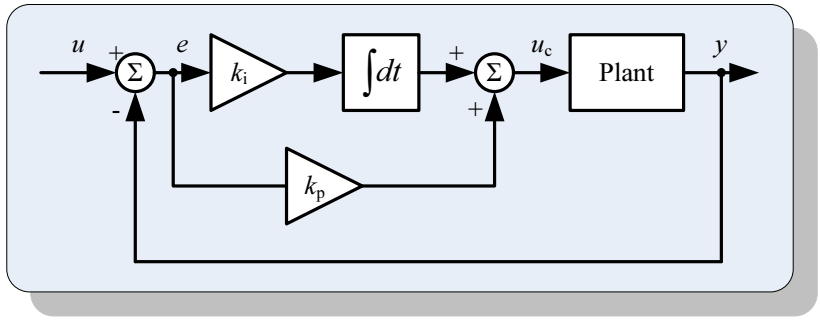


Figure 10.14: *A standard PI-controller.*

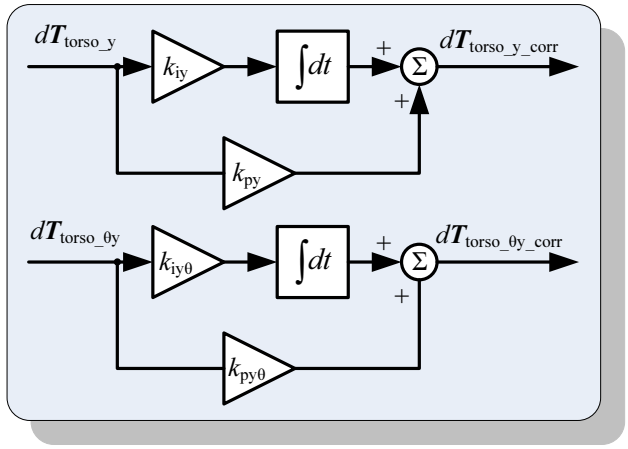


Figure 10.15: *Implementation of two PI-controllers used to control the ZMP position.*

The main subject of this design is how to determine the controller gains, these being the proportional gains and the integral gains. No general rise-time, overshoot nor other control parameters can be calculated, meaning that no traditional controller tuning methods can be applied. Similar, it is not possible to analyze the stability of the controller using standardized methods since the system is a non-linear hybrid system.

Due to this, it has been necessary to tune the ZMP controller by hand. This was done in the model to realize a starting point for fine-tuning of the controller on the robot. By using the model, the gains shown in Table 10.3 was found.

	k_{iy}	k_{py}	$k_{\mathrm{iy}\theta}$	$k_{\mathrm{py}\theta}$
Tuned using model	0.050	0.050	0.010	0.005
Fine-tuned using robot	0.050	0.170	0.005	0.100

Table 10.3: *Controller gains used in the ZMP controller.*

After implementation of the controller, the gains were fine-tuned in order to achieve a better result. The new gains are also found in this table. Surprisingly, it was possible to increase the controller gains, and hereby achieve a better performance than simulated. Unfortunately, it was not possible to simulate the controller performance with the new parameters since these made the system unstable. The simulation of the controller is described in the following section. It is seen that the values for the proportional gains are rather small. This is due to the fact that larger values will result in large accelerations that will make the ZMP move in the opposite direction. This can be seen from Equation (2.8), where the basic equations used for ZMP calculation is found.

10.4.2 Simulating the ZMP Controller Performance

To test the ZMP controller, four tests were carried out. First, the controller was tested while standing in zero-position, meaning that the robot was in DSP. Next, the controller was tested in SSP. In both DSP and SSP, steps in the ZMP reference are applied along both the x- and y-axis.

Simulation data and comments for all simulations and tests are found in Appendix K on page 261. The results from the simulations and tests are summarized in this chapter.

ZMP Controller Simulation in DSP

The controller was implemented and simulated on the complete model, and two experiments were conducted. First a $1\,\mathrm{cm}$ step in the ZMP reference is applied along the x-axis. Next, a $2\,\mathrm{cm}$ step was applied along the y-axis. The simulated performance of the controller is seen in Figure K.1 on page 262 and K.2 on page 262.

The reason for using a larger step along the y-axis is that the PoS is greater than along the x-axis. Overshoot has to be avoided since this will make the robot fall. The cost of this is a slower controller.

ZMP Controller Simulation in SSP

To test the ZMP controller in SSP, the robot was positioned in SSP and then the right leg was lifted $2\,\mathrm{cm}$ above the ground. After the robot movements had settled, a $1\,\mathrm{cm}$

step was applied along the x-axis. Next, a $0.5\,\mathrm{cm}$ step was applied along the y-axis. The response of the system is shown in Figure K.3 on page 263 and K.4 on page 263.

It must be concluded that a sampling frequency of $40\,\mathrm{Hz}$ is too slow for applications like a walking biped robot. It is doubtable that ZMP controller would work if implemented on a walking biped robot due to the slow response times that can be seen from the figures. The ZMP controller was, however, capable of moving the ZMP to the correct position, so the method must be considered applicable.

10.4.3 Implementation and Verification

The ZMP controller was implemented on the actual system and with minor changes to the controller gains ZMP control was realized. The controller was implemented as shown in Figure 10.13. To test the controller, steps were given to the reference position of the ZMP.

Testing ZMP Controller in *x*-direction

In this test, a step of $2.5\,\mathrm{cm}$ on the reference to the ZMP is applied in the x-direction. The corresponding position of the ZMP can be seen in Figure K.5 on page 264. It is seen that the ZMP controller manages to move the position of the ZMP to the desired location within 10 seconds which is too slow if the controller should be used while the robot is walking.

Testing ZMP Controller in *y*-direction

In this test, a step was applied to the position of the ZMP in the y-direction. The step is applied with a magnitude of $2.5\,\mathrm{cm}$ compared to the starting position. The corresponding position of the ZMP can be seen in Figure 10.16.

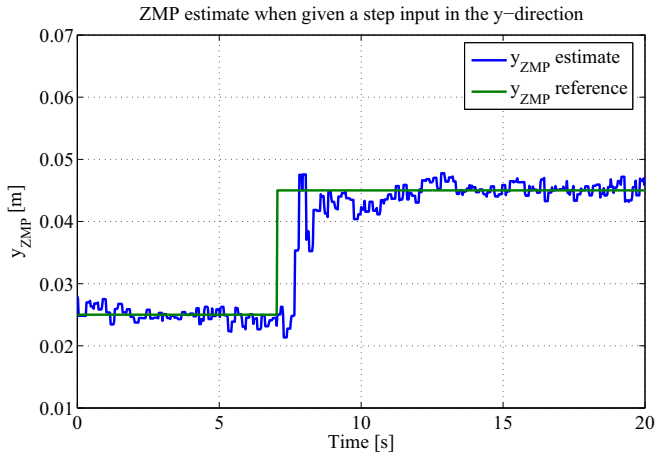


Figure 10.16: *Testing the ZMP controller on the real system in DSP by applying a step in the y-direction.*

It is seen that the ZMP controller is faster in the y-direction than in the x-direction, and that it reaches the reference position after one second. This is mainly due to the

controller gains, that are larger than those for the controller in the x-direction.

To summarize, the slow response of the ZMP controller is not as pronounced on the actual system due to the possibility of using larger controller gains. The controller is capable of moving the ZMP to a desired location and it must be concluded that this method is applicable for use on a walking biped robot if faster sampling time and controller update frequency is possible.

10.5 Hybrid Posture Controller

A hybrid posture controller is implemented in order to maintain the correct attitude of the torso at all times. This controller is necessary due to backlash in the joints. The basic structure of the controller is shown in Figure 10.17.

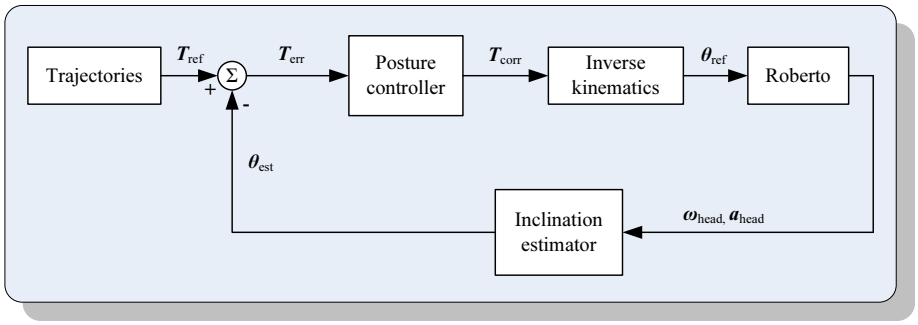


Figure 10.17: *Overview of the posture controller.*

The basic concept of this controller is as follows. Using the inclination estimator, described in Section 10.2.3 on page 140, the inclination of the torso around the x- and y-axis is estimated. This estimate is subtracted from the torso inclination reference (found in $\boldsymbol{T}_{\text{ref}}$). The result is the walking trajectories, with the torso inclination error. These trajectories are then fed into the controller which control output goes to the inverse kinematic model.

Due to pronounced backlash in SSP, this controller has to be a hybrid controller, capable of switching between three different controllers based on the current phase; SSP-L, SSP-R or DSP. The following section describes the development of the three controllers. The specific controller is chosen by using the phase estimate described in Section 10.2.1.

10.5.1 Designing the Posture Controller

The hybrid posture controller consists of three different PI-controllers. This section describes the development of the DSP controller, as well as the two SSP controller.

DSP Posture Controller Design

In DSP, the backlash around the x-axis is limited, hence no further backlash compensation other than the controller it self is necessary. The structure of this controller

complies with the overall structure seen in Figure 10.17. The PI-controller is similar to the ZMP-controller, being a standard PI-controller.

The design problem consists of finding the controller gains. By using the complete model, the controller performance is simulated and the controller gains have been found to those listed in Table 10.4.

	$k_{\text{i\_sagittal}}$	$k_{\text{p\_sagittal}}$	$k_{\text{i\_frontal}}$	$k_{\text{p\_frontal}}$
Tuned using model	0.100	0.250	0.009	0.080
Fine-tuned using robot	0.015	0.300	0.015	0.128

Table 10.4: *Controller gains used in the posture controller.*

After implementation, the controller gains were fine-tuned. The final values are shown in the same table. All gains were increased to improve controller performance. As with the ZMP controller, simulation of the controller with the new gains was not possible since the system became unstable and started to oscillate.

SSP Posture Controller Design

In SSP, the backlash is so severe that static SSP is impossible to obtain. The backlash is pronounced about both the x- and y-axis. To compensate for this and as an attempt to realize a static SSP posture, the design of the posture controller for SSP diverts slightly from the controller used to control the torso attitude in DSP.

The basic idea is that an inclination error of the torso will be compensated for by adding half of the error inclination to the torso, and the other half to the supporting ankle. This is done since it is estimated that the backlash is equal for both these joints as the load is almost the same. This can be seen from Figure 10.18. Here, the inclination of the torso should be zero degrees, but due to backlash, the posture is altered. The actual angle of the torso, $\theta_{\text{torso\_x}}$ is the error. This error inclination will be corrected by subtracting half of this error from θ_2 and θ_4.

The structure of this controller complies with the overall structure seen in Figure 10.17 with minor alterations. The SSP posture controller is shown in Figure 10.19. The two SSP controllers, for SSP-L and SSP-R are similar.

As seen, two controllers are used to maintain the correct attitude in SSP. The design problem consists of finding the controller gains for these controllers. By using the model the controller performance is simulated and the controller gains has been found to these listed in Table 10.5.

	$k_{\text{i\_sagittal}}$	$k_{\text{p\_sagittal}}$	$k_{\text{i\_frontal}}$	$k_{\text{p\_frontal}}$
Hip and ankle controller	0.010	0.025	0.007	0.06

Table 10.5: *Controller gains used in the posture controller.*

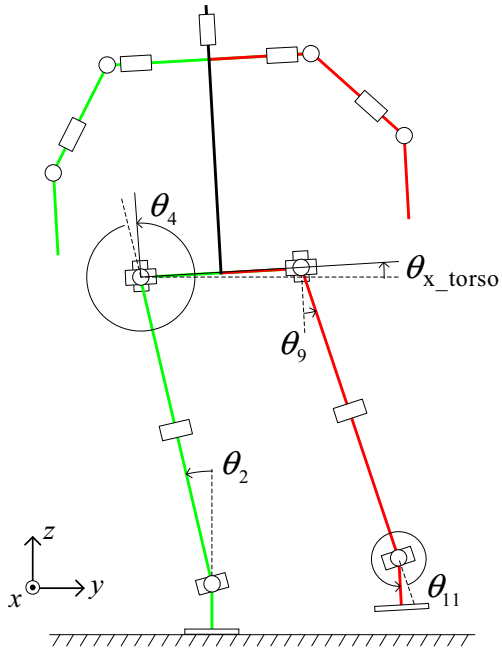


Figure 10.18: *Illustration of the biped. Due to backlash the torso inclination needs to be corrected.*

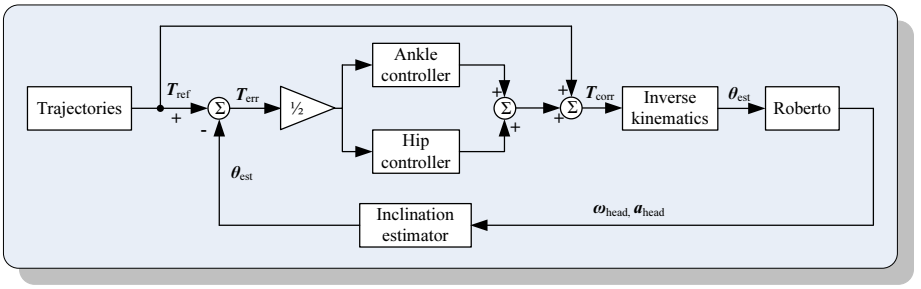


Figure 10.19: *Posture controller used for SSP.*

In previous design sections, the controllers were implemented and the controller gains were fine-tuned in order to achieve better performance. This was however not possible in SSP.

10.5.2 Simulating the Posture Controller Performance

The performance of the hybrid posture controller was tested in both DSP and SSP. First, the controller was tested in DSP and steps were given to the inclination of the torso around the x-axis and around the y-axis in two different tests. Next, the posture controller was tested in SSP. To verify that the controller is capable of suppressing disturbances caused by backlash, $0.087\,\mathrm{rad}$ is subtracted from the inclination instead of giving a step with the reference. This will simulate of effect caused by backlash. This substitution will only be applied in SSP in around the x-axis, as backlash is most pronounced around this axis.

Posture Controller Simulation in DSP

In this test, a a step of $0.09\,\mathrm{rad}$ was applied about the x-axis. Next, a step of $0.09\,\mathrm{rad}$ was applied about the y-axis. The robot was set in zero-position before the tests were carried out. The responses are seen in Figure K.7 on page 266 and K.8 on page 267.

It can be seen that the controllers work as intended but the settling times are too slow. This is mainly due to the slow sampling rate of the system, that puts a limit on the gains.

Posture Controller Simulation in SSP

This test differs from the test conducted in DSP, but only for the test regarding step-input about the x-axis. Instead of giving an inclination step input, $0.087\,\mathrm{rad}$ is subtracted from the inclination estimate. This simulates the behavior of backlash in the system. The posture controller will then suppress this disturbance by increasing the inclination in both the ankle and the torso as described in the design section. The response is seen in Figure K.9 on page 267.

Finally, a step on $0.087\,\mathrm{rad}$ was applied about the y-axis. The response of the system is shown in Figure K.10 on page 268.

Only the controller for SSP-L is tested, since the controller for the right leg is similar, the simulation is not shown.

10.5.3 Implementation and Verification

This section describes the implementation of the DSP posture controller. The controller will be tested by two experiments, where a step in the inclination is given, in both the sagittal and the frontal plane individually. It was not possible to verify the performance of the SSP controller due to backlash and wobbling. This will be described more detailed in the following.

Testing the DSP Posture Controller About the x-axis

To test the posture controller, the robot was positioned in zero position. A $0.09\,\mathrm{rad}$ step was then given on the inclination reference about the x-axis. The measured inclination, $\theta_{\mathrm{x,ref}}$, and reference input is seen in Figure K.11 on page 269. In this figure, it is seen

that there is a $0.03\,\mathrm{rad}$ overshoot but that the controller otherwise settles at the correct inclination.

Testing the DSP Posture Controller About the y-axis

To test the posture controller about the y-axis, a $0.09\,\mathrm{rad}$ step was given. The biped was placed in zero position with zero inclination. After 15 seconds the step was applied. The resulting inclination is seen in Figure K.12 on page 270.

Similar to the posture controller around the x-axis it must be concluded that this controller is too slow to realize posture control while performing dynamic walk. Once again, backlash, wobbling in the joints and a slow inclination estimator is the main reason for this.

10.6 Partial Conclusion to Control

In this chapter, control signals have been generated by using all the sensors available on the robot. The sensors; being an accelerometer, a gyroscope, pressure sensors, and position feedback all have advantages and disadvantages.

To provide control input for the hybrid posture controller, an inclination estimator has been developed. Through tests, it has been concluded that the inclination estimator, described in Section 7.3 is too slow to provide a descent control input for the posture controller. The rate of change of the signal is too slow to allow real-time posture control. A faster inclination estimator must be implemented in order to design a posture controller, working in an inner control loop.

The ZMP estimate provided by the accelerometer and position feedback from the servos is also unreliable due to the high noise contribution on position measurements when heavily loaded. Also the accelerometer measurements had large noise contributions in the signal. It has therefore been chosen not to use this estimate for control purpose.

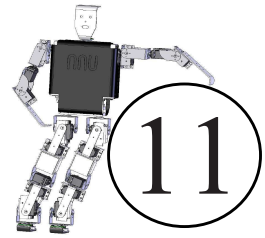
The ZMP estimate, calculated on behalf of the pressure measurements, provided a control signal, that was not dominated by noise, and made it possible to implement a ZMP controller. Implementing a Kalman-filter, using input from the model, it would be possible to achieve a better ZMP estimate if the accelerometer and position feedback was included in the estimate. This seems plausible since the estimate contains some level of information. It has, however, not been verified due to the limited processing power available on the robot.

Even though the output from the ZMP estimate is usable, there is one major problem with the pressure sensors. Due to the small feet, the sensor output highly depends on the positioning of the feet. If the feet are moved as little as $1\,\mathrm{mm}$ the ZMP estimate tends to change dramatically. Because of this, the position of the feet had to be calibrated before each controller test, by measuring the load on each pressure sensors. The feet were then repositioned until descent pressure measurements were achieved. This is not a feasible solution, and it must therefore be concluded that the pressure sensors alone do not provide a sufficient ZMP estimate.

The ZMP controller was simulated and implemented with a successful result for DSP only, and successful simulated for SSP. This controller only used the ZMP estimate provided by the feet as control input. By changing the position of the feet until a descent ZMP estimate was achieved, it was clear that the ZMP position was moved to the desired location. It has only been possible to implement the controller for use in DSP while standing still. This is mainly due to the slow controller frequency. Only a $40\,\mathrm{Hz}$ update frequency could be achieved on the system due to the limited processing power. This results in jerky movements of the robot and noise dominated estimates.

The posture controller was simulated in both DSP and SSP and implemented for DSP. Due to backlash, SSP is not possible when standing still and it has there for only been tested using simulations.

EPILOGUE



Chapter contents

THIS CHAPTER CONTAINS *the epilogue of this master's thesis. It will start with a discussion of some of the main problems experienced throughout the project. All the main areas such as mechanical design, hardware design, software design, modeling, trajectory generation and control will be discussed. Afterward a conclusion on the entire master's thesis will be given.*

11.1 Discussion

This section will discuss some of the areas covered in this master's thesis. Mainly areas that have been problematic in some sense will be accentuated and discussed.

11.1.1 Mechanical Design

A complete mechanical solution was given for a humanoid robot, which should be able to resemble human walk. It was emphasized that the solution should build on available mechanical solutions, but first and foremost it should result in a robot with human proportions. Therefore some of the mechanical parts have been custom-made for this project. An entire humanoid platform was constructed, which using the multi DoF joint created enables straight and curved walk.

An issue regarding the mechanical solution was an observed backlash in the system, especially outspoken in the legs. At the design stage, this was sought minimized by using custom-made solutions, such as the multi DoF joint used in the hip. These have been satisfactory for the single joint, but in such a complex mechanical construction, backlash is difficult to avoid entirely. Therefore, when building up a kinematical chain, with a large number of joints, the backlash becomes more significant. A consequence was that when the robot was positioned in zero position, it could not be guaranteed that it was in the exact same position every time. The weight distribution on the feet therefore changed from time to time and the measurements of the strain gauges were not always reliable. A solution to this problem could be to measure the backlash by mounting a potentiometer directly on the joint. The position feedback provided by the internal potentiometers does not measure backlash, and it is therefore not possible to control the backlash of the mechanics.

Custom-made feet were constructed which made it possible to measure the force distribution of the robot on the feet. The feet were made as a bendable construction of spring steel plates, making it possible to measure the strain on the plates when exposed to a force. Cross-talk was minimized using dampers inserted. To minimize the oscillating effect, caused by the spring steel plate, the dampers should have been designed especially for this. As the feet were designed to have humanoid proportions they were small in size, with only $15\,\mathrm{mm}$ between the inner and outer strain plate. This results in a very small PoS. The oscillating effect of the spring steel plates had the observed effect that the robot would sway some degrees forward and backwards in the y-direction, exerting a limit-cycle behavior, and the whole system would wobbling. The effect could be observed both in frontal and sagittal plane, but was most outspoken in the sagittal plane when standing in DSP. When the robot should stand in SSP, the PoS was further minimized in the y-direction as there was only one strain plate to give support at the heel. While the strain plates, at the left and right side of each foot, should give support by minimizing the pressure on the middle strain gauge plate, they rather amplified the oscillations due to the spring effect. The stability of the robot in SSP could then be compared to a pivoted beam, as the weight would keep shifting from side to side. This problem is also illustrated in Figure 11.1. A solution to this problem could be to design a mechanical solution which would utilize the entire foot as support, rather than only these three points, and a construction that is more solid than the bendable spring steel. Thereby the introduced oscillations could be minimized.

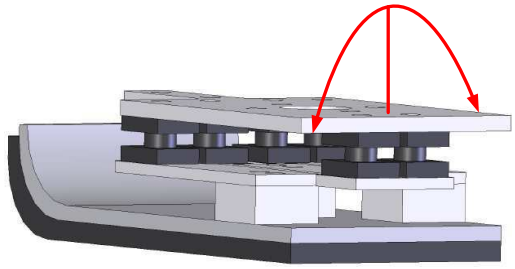


Figure 11.1: *When the weight of the robot moves from side to side the whole system starts wobbling.*

11.1.2 Hardware Design

A hardware solution was fully designed and integrated on-board the robot. This included battery and a wireless interface. An ARM9 development board was bought, providing a 200 MHz processor. Furthermore a custom-made input/output PCB was made, enabling filtering and multiplexing of sensor measurements. A PCB was also designed for rejection of common mode noise and amplification of the measurements from the strain gauges mounted on the feet. All the implemented hardware worked and performed without any remarks.

As actuators, standard servo motors were used. These are limited to be controlled with a maximum frequency of 50 Hz. By designing a PCB for each servo motor this update frequency could be increased. This would require that the current PCB located internally in the servo motor, was to be removed. A new software protocol and hardware solution allowing faster update frequency should be designed. It was found that the number of intermediate positions which were possible to obtain, with the trajectories designed for dynamical walk, were too low with the limited update frequency. Tests from the model have shown that to obtain dynamical walk, the servo motors would have to be updated with a frequency of 250 Hz. Another solution could also be the implementation of true velocity control, instead of the velocity control available from the Polulu boards.

11.1.3 Software Design

The software was designed using an optimized Debian solution, thereby implementing a complete Linux platform on-board the robot. This solution utilized a real-time target solution for MATLAB. A multi-threaded software platform was developed and the system was thereby further enhanced. The platform integrated on the robot was found feasible as it had an integrated FPU and a 200 MHz clock frequency. Unfortunately the FPU could not be used, since it is not supported by Kernel 2.4, and GCC 3.3.5. The kernel could not be updated to version 2.6 as the object files, supplied by Technologic Systems, were compiled for version 2.4. A solution to this could have been to program all the mathematical functions in assembler, but this task was found too cumbersome to realize in the project period. Due to the unsupported FPU, floating point operations had to be emulated in software.

Throughout the project, solutions have been made which took in mind the limited processing power. It would have been advantageous to be able to implement some of the model for use in model-based control. But this was not possible with the current

processor. Experiments showed that a $10\,\mathrm{s}$ simulation of the model was conducted in $42.6\,\mathrm{s}$, which is approximately $1/4$ real-time, on a computer with an Intel P4 $2.8\,\mathrm{GHz}$ processor and $512\,\mathrm{MB}$ ram running Windows XP. A feasible solution would be to implement the needed states of the model on an Intel P4 $2\,\mathrm{GHz}$ processor with a fully supported FPU. It is evaluated that this would be sufficient to implement the most important parts of the model.

11.1.4 Modeling

A complete model was developed which gave a full description of the system, giving a complete input to output model of the actual system. Tests showed that the model was able to give a good estimate of the real system. But some of the issues the test revealed were that some unmodeled effects in the real system had a high impact on the results, which resulted in an offset error between the output from the model and the measured output on the real system. Especially the spring effect of the strain plates mounted on the feet, causing an amplified bend of the plates, should be modeled if a more precise description of the real system is wanted. The spring effect further caused oscillations in the real system, while this was not the case for the model.

The model gave a description of the ZMP, but if the ZMP was to move outside the PoS the robot did not fall. This effect could be added by modeling the tip of the foot as an extra DoF, which would limit the model to fall forward but still give a better indication of the dynamics during walk.

As the model also included all the internal states of the system, it was a good base for learning the dynamics of the system and for development and testing of the controllers. The trajectories used in the project were created using the model and the controllers were developed and tested using the model as well.

11.1.5 Trajectory Design

A complete set of dynamic walking trajectories were created, including both the start-up and the stop phase. These trajectories were generated off-line and optimized toward keeping the ZMP inside the PoS at all times. Furthermore the trajectories were designed to minimize the energy consumption. It was found that the designed dynamical trajectories for the model kept the ZMP inside the PoS, but for the real robot the ZMP was not kept inside the PoS. Further it was found that because of the control frequency of the servos and backlash in the joints, it was not possible to implement the designed trajectories. Some new and slower trajectories were designed, and performed better than the first set of trajectories, but the problems with backlash were still too significant to obtain walk.

As the trajectories were designed using the model, these could have been optimized further, if the model was able to roll on the front of the feet. This could have aided the design of the trajectories in such a way that a more human-like walk could have been achieved.

11.1.6 Control Design

In the control design, several individually estimators and controllers were developed. These controllers aimed at controlling the position of the ZMP and the posture of the robot. This was found advantageous as the ZMP was also used to design the trajectories. Simulations showed that the controllers were able to minimize the error in the position of the ZMP during DSP and SSP. In DSP, the ZMP was tested with a successful result. Similar, the posture controller was tested in DSP and SSP, and verified in DSP. It was concluded that both controllers was too slow to control a walking biped robot.

As an important factor in human walk is to foresee what happens next, it would be beneficial to use model-based control to make the system more adaptive. Thereby the position of the ZMP could have been predicted some steps ahead, which would have been important for correct control of the robot during walk.

11.2 Conclusion

The main goal of this project was to develop a humanoid robot and make it perform dynamic walk. The first part, concerning the development was achieved by designing a 21 DoF biped robot from scratch. The developed robot possesses human proportions, i.e., the correct relationship between length of limbs and the height of the robot. Furthermore a special joint was developed to imitate the ankle and the hip joints in human beings, where several degrees of freedom are located in one joint. The feet were also designed to fit the size of a human, making the support area on ground very small. A big advantage of the robot-design is the possibility to walk in a curved path, using the multi DoF joint which has been designed.

To control this mechanical structure, 21 servo motors were inserted, such that each DoF became actuated. The servo motors enable a simple interface to control the position of each joint. Furthermore a hardware platform has been developed that enables the humanoid robot to be completely autonomous. An on-board computer was placed on the robot in order to control all actuators and to process the inputs from the large number of sensors located on the system. Furthermore the system was equipped with batteries, making it possible to run for around one and a half hour without recharging. A complete software platform was constructed, including an interface to the robot which enables easy testing of controllers. The software was designed as multi-threaded real-time system, and was verified working.

Before any controllers for the system were developed, a model was derived. The model is a complete mathematical description of the complex non-linear dynamical system. Using this model it was possible to simulate complete walking cycles and extract information on stability properties of the walk, information on how forces and torques worked in different joints during the walk. The most challenging part of the modeling was the dynamical part of the system, especially in DSP where a closed kinematical chain made the modeling a complex task. The problems were however solved, using a novel solution proposed in this thesis, and a complete dynamical model, taking all 21 DoF and all three dimensions into consideration, has been developed. The model was verified through a series of tests performed on the sub-models, and finally a test that verified the complete model. The result of the tests were, that the model is valid, but

if a more precise description of the system is essential, it is necessary to expand the model to include the bendable plates in the feet as well as the backlash that is present in the system. It is concluded that the accuracy of the model is sufficient for the scope of this project.

In order to obtain dynamical walk, a set of dynamical walking trajectories were developed, which were optimized toward improving the stability properties and lowering the energy consumption. The trajectories gave a satisfying result when simulated, but due to some drawbacks on the real system, it was not possible to obtain dynamical walk. The main reason for this, is that the control frequency of the servo motors is only $50\,\mathrm{Hz}$, resulting in a rough set of position references to the servo motors, which caused a shaking of the entire system. This was sought solved by creating slower trajectories, but could not be removed completely. Another decisive reason for not being able to perform dynamical walk, was the backlash in the system.

The reliability of the control input is also an essential problem in the system. Due to the small feet, the ZMP estimate is very dependent on the orientation of the feet.

To maintain stability of the system two controllers were developed; a hybrid posture controller and a ZMP controller. In order to minimize the effect of backlash, a posture controller was implemented. It was concluded that the posture controller worked as intended in DSP, both in sagittal and in frontal plane. A test of the controller in SSP could not be performed due to backlash. It was however concluded that the controller was too slow to be implemented on a walking robot, due to the low sample frequency of the system.

The ZMP controller worked as intended both in sagittal and frontal plane. As with the posture controller, it was not fast enough to be implemented on a walking robot. It was not possible to test the ZMP controller on the actual system in SSP, but simulations documents the performance in this phase.

To summarize the conclusion; a humanoid robot platform was developed, capable of performing dynamic walk and even curved gait. This consists of a complete mechanical solution designed from scratch, an on-board computer and a layered software platform with a wireless interface that supports easy implementation of controllers and trajectories. A complete dynamical model has been derived, and verified, with the same inputs and outputs as the real system. Using the model, dynamic stable walking trajectories have been designed. Controllers, which increase stability of the robot, have been designed and tested. To conclude; this thesis gives a solution to the *development, modeling and control of a humanoid robot*.

BIBLIOGRAPHY

[Analog Devices, 2000] Analog Devices (2000). *Precision Single Supply Instrumentation Amplifier, AMP04*. [CD-ROM, 2007, literature/datasheets/amp04.pdf].

[Applieddata.net, 2007] Applieddata.net (2007). Applied data's root filesystem with support for the maverick engine. `http://armel.applieddata.net/developers/linux/eabi/armel-root-fs.tar.bz2`.

[Arakawa and Fukuda, 1996] Arakawa, T. and Fukuda, T. (1996). Natural motion trajectory generation of biped locomotion robot using genetic algorithm through energy optimization. *IEEE International Conference on Systems, Man, and Cybernetics*. [CD-ROM, 2007, literature/trajectory generation/natural_motion_trajectory_generation_energy_optimization.pdf].

[Azevedo et al., 2005] Azevedo, C., Espiau, B., Amblard, B., and Assaiante, C. (2005). Bipedal locomotion: toward unified concepts in robotics and neuroscience. *Biological Cybernetrics*. [CD-ROM, 2007, literature/general/bipedal_locomotion_toward_unified.pdf].

[Bachar, 2004] Bachar, Y. (2004). Developing controllers for biped humanoid locomotion. Technical report, University of Edinburgh, School of Informatics. [CD-ROM, 2007, literature/control/developing_controllers_for_biped.pdf].

[Baerveldt and Klang, 1997] Baerveldt, A.-J. and Klang, R. (1997). A low-cost and low-weight attitude estimation system for an autonomous helicopter. *Proc. of the IEEE Int. Conf. on Intelligent Engineering Systems*. [CD-ROM, 2007, literature/sensor_fusion/attitude_estimation_system_for_an_autonomous_helicopter.pdf].

[Bak and Izadi-Zamanabadi, 2006] Bak, T. and Izadi-Zamanabadi, R. (2006). Lecture notes - hybrid systems. Technical report, Aalborg University. [CD-ROM, 2007, literature/hybrid_systems/hs.pdf].

[Bhanderi, 2006] Bhanderi, D. (2006). Linux soft real-time target v2.3. `http://www.control.aau.dk/~danji/downloads`.

[Borghese et al., 1996] Borghese, N. A., Bianchi, L., and Lacquaniti, F. (1996). Kinematic determinants of human locomotion. *Journal of Physiology*. [CD-ROM, 2007, literature/trajectory generation/kinematic_determinants_of_human_locomotion.pdf].

[Bruneau et al., 2001] Bruneau, O., Ouezdou, F. B., and Fontaine, J.-G. (2001). Dynamic walk of a bipedal robot having flexible feet. *International Conference on Intelligent Robots and Systems*. [CD-ROM, 2007, literature/gait/dynamic_walk_flex_feet.pdf].

[Buschmann et al., 2006] Buschmann, T., Lohmeier, S., Ulbrich, H., and Pfeiffer, F. (2006). Dynamics simulation for a biped robot: Modeling and experimental verification. *Proceedings of the 2006 IEEE International Conference on Robotics and Automation Orlando, Florida - May 2006*. [CD-ROM, 2007, literature/modeling/Dynamics_Simulation_for_a_Biped_Robot-Modeling_and_Experimental_Verification.pdf].

[Buss et al., 2003] Buss, M., Hardt, M., Kiener, J., Sobotka, M., Stelzer, M., von Stryk, O., , and Wollherr, D. (2003). Towards an autonomous, humanoid, and dynamically walking robot: Modeling, optimal trajectory planning, hardware architecture, and experiments. *Proceedings of the 3rd International Conference on Humanoid Robots 2003, Karlsruhe, Germany*. [CD-ROM, 2007, literature/-control/towards_an_autonomous.pdf].

[Capi et al., 2003] Capi, G., Nasu, Y., Barolli, L., and Mitobe, K. (2003). Real time gait generation for autonomous humanoid robots: A case study for walking. *Robotics and Autonomous Systems, vol. 42*. [CD-ROM, 2007, literature/-gait/real_time_gait_generation.pdf].

[CD-ROM, 2007] CD-ROM (2007). The enclosed CD-ROM.

[Chaillet et al., 1994] Chaillet, N., Abba, G., and Ostertag, E. (1994). Double dynamic modelling and computed-torque control of a biped robot. [CD-ROM, 2007, literature/modeling/double_dynamic_model.pdf].

[Choi et al., 2004] Choi, Y., You, B.-J., and Oh, S.-R. (2004). On the stability of indirect zmp controller for biped robot systems. [CD-ROM, 2007, literature/trajectory generation/on_the_stability_of_indirect_zmp.pdf].

[Christensen et al., 2006] Christensen, J., Nielsen, J. L., Svendsen, M. S., Svenstrup, M. S., Winter, K., and Ørts, P. F. (2006). Modelling and control of a biped robot. Technical report, Aalborg University, Institute of electronic systems, Department of Control Engineering. [CD-ROM, 2007, literature/modeling/p8_report.pdf].

[Cirrus Logic, 2004] Cirrus Logic (2004). *High Speed ARM9 System-on-Chip Processor with MaverickCrunch, EP9302*. [CD-ROM, 2007, literature/datasheets/EP9302-arm9.pdf].

[CodeSourcery, 2007] CodeSourcery (2007). Gnu toolchain for arm eabi. http://www.codesourcery.com/gnu_toolchains/arm/download.html.

[Collins et al., 2001] Collins, S. H., Wisse, M., and Ruina, A. (2001). A three-dimensional passive-dynamics walking robot with two legs and kness. *The International Journal of Robotics Research Vol. 20, No. 7, July 2001, pp. 607-615*. [CD-ROM, 2007, literature/modeling/3d_passive_dynamic.pdf].

[Courtine and Schieppati, 2004] Courtine, G. and Schieppati, M. (2004). Human walking along a curved path: Body trajectory, segment orientation and the effect of vision. *European Journal of Neuroscience, vol. 18*. [CD-ROM, 2007, literature/gait/human_walk_curved_path_1.pdf].

[Craig, 1989] Craig, J. J. (1989). *Introduction to Robotics - Mechanics and Control, Second Edition*. Addison Wesley.

[Debian, 2007] Debian (2007). Debian arm eabi port. `http://wiki.debian.org/ArmEabiPort`.

[Deo and Walker, 1997] Deo, A. S. and Walker, I. D. (1997). Minimum effort inverse kinematics for redundant manipulators. *IEEE Transactions on Robotics and Automation, vol. 13, no. 5,*. [CD-ROM, 2007, literature/modeling/min_effort_inverse_kinematics.pdf].

[Devices, 2006] Devices, A. (2006). *ADXL: Small, Low Power, 3-Axis +/- 3 g, iMEMS Accelerometer*. [CD-ROM, 2007, literature/datasheets/ADXL330_0_imu_accelerometer.pdf].

[Djoudi et al., 2005] Djoudi, D., Chevallereau, C., and Aoustin, Y. (2005). Optimal reference motions for walking of a biped robot. *International conference on robotics and automation*. [CD-ROM, 2007, literature/gait/optimal_reference_motions.pdf].

[Doi et al.,] Doi, M., Hasegawa, Y., and Fukuda, T. Realization of 3-dimensional dynamic walking based on the assumption of point-contact. *International Conference on Robotics and Automation, Barcelona, Spain, April 2005*.

[Erbatur et al., 2002] Erbatur, K., Okazaki, A., Obiya, K., Takahashi, T., and Kawamura, A. (2002). A study on the zero moment point measurement for biped walking robots. *7th International Workshop on Advanced Motion Control, IEEE*. [CD-ROM, 2007, literature/modeling/zmp_measure.pdf].

[Foxlin, 1996] Foxlin, E. (1996). Inertial head-tracker sensor fusion by a complementary separate-bias kalman filter. *Proceedings of the IEEE 1996 virtual reality annual international symposium, Santa Clara, CA*. [CD-ROM, 2007, literature/sensor_fusion/intertial_head_tracker_sensor_fusion.pdf].

[Gielo-Perczak, 2001] Gielo-Perczak, K. (2001). The golden section as a harmonizing feature of human dimensions and workplace design. *Theoretical Issues in Ergonomics Science*. [CD-ROM, 2007, literature/gait/human_proportions.pdf].

[Goldenberg et al., 1985] Goldenberg, A. A., Benhabib, B., and Fenton, R. G. (1985). A complete generalized solution to the inverse kinematics of robots. *IEEE Journal of Robotics and Automation, vol. ra-1, no. 1*. [CD-ROM, 2007, literature/-modeling/A_Complete_Generalized_Solution_to_the_Inverse_Kinematics.pdf].

[Guan et al., 2005] Guan, Y., Yokoi, K., Stasse, O., and Kheddar, A. (2005). On robotic trajectory planning using polynomial interpolations. *IEEE International Conference on Robotics and Biomimetrics*. [CD-ROM, 2007, literature/trajectory generation/on_robotic_trajectory_planning_using_polynomial_interpolations.pdf].

[Hardt et al., 1999] Hardt, M., Kreutz-Delgado, K., and Helton, J. W. (1999). Optimal biped walking with a complete dynamical model. [CD-ROM, 2007, literature/-modeling/optimal_walk_complete_dynamical_model.pdf].

[HiTec, 2005] HiTec (2005). *Announced specification of HS-645MG standard deluxe high torque servo*. [CD-ROM, 2007, literature/datasheets/HS-645MG_servomotor.pdf].

[HiTec, 2006] HiTec (2006). *Announced specification of HSR-5995TG standard deluxe high torque servo*. [CD-ROM, 2007, literature/datasheets/HSR-5995TG_spec.pdf].

[Hsiao et al., 2006] Hsiao, F.-B., Liu, T.-L., Chien, Y.-H., and Lee, M.-T. (2006). The development of a target-lock-on optical remote sensing system. *Aeronautical Journal*. [CD-ROM, 2007, literature/sensor_fusion/The_development_of_a_target_lock.pdf].

[Huang et al., 2000] Huang, Q., Kaneko, K., Yokoi, K., Kajita, S., Kotoku, T., Koyachi, N., Arai, H., Imamura, N., Komoriya, K., and

Tanie, K. (2000). Balance control of a biped robot combining off-line pattern with real-time modification. *IEEE International Conference on Robotics & Automation*. [CD-ROM, 2007, literature/trajectory generation/balance_control_of_a_biped_robot_combining_offline_pattern_with_realtime_ modification.pdf].

[Huang et al., 2001] Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Arai, H., Koyachi, N., and Tanie, K. (2001). Planning walking patterns for a biped robot. *IEEE Transactions on Robotics and Automation*. [CD-ROM, 2007, literature/trajectory generation/planning_walking_patterns_for_a_biped_robot.pdf].

[Huelsman, 1993] Huelsman, L. P. (1993). *Active and Passive Analog Filter Design, an introduction*. McGraw-Hill, Inc.

[Hurmuzlu, 2001] Hurmuzlu, Y. (2001). Dynamics of bipedal gait part i: Objective functions and the contact event of a planar five-link biped. [CD-ROM, 2007, literature/modeling/impact_phase/Impact_phase.pdf].

[InvenSense Inc, 2006] InvenSense Inc (2006). *Integrated Dual-Axis Gyro, IDG-300*. [CD-ROM, 2007, literature/datasheets/IDG-300_Datasheet_imu_gyroscope.pdf].

[Ito and Kawasaki, 2005] Ito, S. and Kawasaki, H. (2005). Regularity in an environment produces an internal torque pattern for biped balance control. *Biological Cybernetrics*. [CD-ROM, 2007, literature/modeling/biped_balance_control_periodic_disturbance.pdf].

[Ito et al., 2004] Ito, S., Saka, Y., and Kawasaki, H. (2004). Where center of pressure should be controlled in biped upright posture. *System and computers in Japan, vol. 35, no. 5*. [CD-ROM, 2007, literature/modeling/center_of_presure_model_3.pdf].

[Kajita et al., 2001] Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., and Hirukawa, H. (2001). The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. *International Conference on Intelligent Robots and Systems. Proceedings of the 2001*. [CD-ROM, 2007, literature/modelling/the_3d_linear_inverted_pendulum_mode].

[Kondak and Hommel, 2003] Kondak, K. and Hommel, G. (2003). Control and online computation of stable movement for biped robots. *Proceedings of the 2003 IEEE/RSJ international Conference on Intelligent Robots and Systems*. [CD-ROM, 2007, literature/control/control_and_online_computation.pdf].

[Kuo, 1997] Kuo, A. D. (1997). An optimal control model of human balance: Can it provide theoretical insight to neural control of movement? *American Control Conference, 1997. Proceedings of the 1997*. [CD-ROM, 2007, literature/balance/optimal_control_model_human_balance.pdf].

[Lee et al., 2003] Lee, S., Song, J., Choi, W., and Hong, D. (2003). Position control of a stewart platform using inverse dynamics control with approximate dynamics. *Mechatronics vol. 13*. [CD-ROM, 2007, literature/modeling/stewart_platform.pdf].

[Löffler et al., 2004] Löffler, K., Gienger, M., Pfeiffer, F., and Ulbrich, H. (2004). Sensors and control concept of a biped robot. *IEEE transactions on industrial electronics, vol. 51, no. 5, October 2004*. [CD-ROM, 2007, literature/sensor_fusion/sensor_control_of_johnny.pdf].

[Li et al., 1993] Li, Q., Takanishi, A., and Kato, I. (1993). Learning control for a biped walking robot with a trunk. *International Conference on Intelligent Robots & Systems, IEEE*. [CD-ROM, 2007, literature/control/learning_control.pdf].

[Maxim Integrated Products, 2006] Maxim Integrated Products (2006). *MAX396 - Precision, 16 channel, Low-Voltage, CMOS Analog Multiplexer*. [CD-ROM, 2007, literature/datasheets/MAX396_multiplexer.pdf].

[McGeer, 1990] McGeer, T. (1990). Passive walking with knees. *IEEE international Conference on Robotics and Automation*. [CD-ROM, 2007, literature/gait/-tad_mcgeer_passive_walking.pdf].

[Mu and Wu, 2003] Mu, X. and Wu, Q. (2003). A complete dynamic model of five-link bipedal walking. *Proceedings of the American Control Conference*. [CD-ROM, 2007, literature/modeling/complete_dynamic_model.pdf].

[Mu and Wu, 2004] Mu, X. and Wu, Q. (2004). Development of a complete dynamic model of a planar five-link biped and sliding mode control of its locomotion during the double support phase. *International Journal of Control*. [CD-ROM, 2007, literature/modeling/dynamic_model_biped_mu_wu.pdf].

[Nielsen, 2003] Nielsen, J. B. (2003). How we walk: Central control of muscle activity during human walking. *Neuroscientist 2003*. [CD-ROM, 2007, literature/-gait/how_we_walk.pdf].

[Ogino et al., 2003] Ogino, M., Hosoda, K., and Asada3, M. (2003). Learning energy efficient walking based on ballistics. *SICE -ANNUAL CONFERENCE-*. [CD-ROM, 2007, literature/trajectory generation/learning_energy_efficielt_walking_based_on_ballistics.pdf].

[Peng et al., 2005] Peng, Z., Huang, Q., Zhang, L., Jafri, A. R., Zhang, W., and Li, K. (2005). Humanoid on-line pattern generation based on parameters of off-line typical walk patterns. *IEEE International Conference on Robotics and Automation*. [CD-ROM, 2007, literature/trajectory generation/humanoid_online_pattern_genration.pdf].

[Popovic, 2006] Popovic, D. B. (2006). Control of walking in humans. [CD-ROM, 2007, literature/gait/popovic_control_of_walking_in_humans.pdf].

[Rehbinder and Hu, 2000] Rehbinder, H. and Hu, X. (2000). Nonlinear pitch and roll estimation for walking robots. *Proceedings of the 2000 IEEE international conference on robotics and automation, Vol. 3, San Francisco, CA, USA*. [CD-ROM, 2007, literature/sensor_fusion/nonlinear_pitch_and_roll_estimation.pdf].

[Rehbinder and Hu, 2001] Rehbinder, H. and Hu, X. (2001). Drift-free attitude estimation for accelerated rigid bodies. *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*. [CD-ROM, 2007, literature/sensor_fusion/Drift_free_attitude_estimation_for_accelerated_rigid_bodies.pdf].

[Robotics and Electronics, 2005] Robotics, P. and Electronics (2005). *Micro Serial Servo Controller, User's Guide*. [CD-ROM, 2007, literature/datasheets/pololu_servo_controller.pdf].

[Santos and Silva, 2005] Santos, V. M. F. and Silva, F. M. T. (2005). Engineering solutions to build an inexpensive humanoid robot based on a distributed control architecture -. *5th IEEE-RAS International Conference on Humanoid Robots*. [CD-ROM, 2007, literature/feet/straingauge_biped2.pdf].

[Sciavicco and Siliano, 1988] Sciavicco, L. and Siliano, B. (1988). A solution algorithm to the inverse kinematic problem for redundant manipulators. *IEEE Journal of Robotics and Automation, vol. 4, no. 4,*. [CD-ROM, 2007, literature/modeling/inverse_kinematic_obstacle_avoid.pdf].

[Serway and Beichner, 2000] Serway, R. A. and Beichner, R. J. (2000). *Physics For Scientists and Engineers with Modern Physics, Fifth Edition*. Saunders College Publishing.

[Shanmugan and Breipohl, 1988] Shanmugan, K. S. and Breipohl, A. (1988). *Random Signals: Detectiobn, Estimation and Data Analysis*. John Wiley and Sons, Inc.

[Shih, 1996] Shih, C.-L. (1996). Analysis of the dynamics of a biped robot with seven degrees of freedom. *International Conference on Robotics and Automation, IEEE*. [CD-ROM, 2007, literature/modeling/biped_dynamics_single_support.pdf].

[Spark Fun Electronics, 2006] Spark Fun Electronics (2006). Spark Fun Electronics. `http://www.sparkfun.com`.

[Takahashi et al., 2005] Takahashi, M., Narukawa, T., Miyakawa, K., and Yoshida, K. (2005). Combined control of cpg and torso attitude control for biped locomotion. *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference*. [CD-ROM, 2007, literature/control/combined_control_of_CPG_and_torso_attitude_control.pdf].

[Technologic Systems, 2006] Technologic Systems (2006). *TS-7400 Data sheet - System on Module with 1.1s Ultra-Fast Bootup to Linux.* [CD-ROM, 2007, literature/datasheets/ts-7400-datasheet.pdf].

[Udwadia and Kalaba, 2001] Udwadia, F. E. and Kalaba, R. E. (2001). Explicit equations of motion for mechanical systems with nonideal constraints. *Transaction of the ASME*. [CD-ROM, 2007, literature/modeling/nonideal_constraints_udwadia.pdf].

[Udwadia and Kalaba, 2002] Udwadia, F. E. and Kalaba, R. E. (2002). On the foundations of analytical dynamics. *International Journal of Non-Linear Mechanics*. [CD-ROM, 2007, literature/modeling/analytical_dynamics_udwadia.pdf].

[Vaganay et al., 1993] Vaganay, J., Aldon, M. J., and Fournier, A. (1993). Mobile robot attitude estimation by fusion of inertial data. *Proceedings of the 1993 IEEE international conference on robotics and automation, Vol. 1*. [CD-ROM, 2007, literature/sensor_fusion/mobile_robot_attitude_estimation.pdf].

[Vukobratović et al., 2006] Vukobratović, M., Borovac, B., and Potkonjak, V. (2006). Towards a unified understanding of basic notions and terms in humanoid robotics. *Robotica, Cambridge University Press*. [CD-ROM, 2007, literature/-general/Towards_a_unified_understanding_of_basic_notions.pdf].

[Vukobratović and Juriĉić, 1969] Vukobratović, M. and Juriĉić, D. (1969). Contribution to the synthesis of biped gait. *Proceedings of the IFAC Symposium on Technical and Biological Problem of Control*. Available from the Library.

[Wisniewski, 2006] Wisniewski, R. (2006). Mechanical systems ii - lagrange mechanics. Technical report, Aalborg University. Lecture notes from lecture three of the course Modelling of Mechanical Systems II. [CD-ROM, 2007, literature/modeling/MechanicalSystemsSection3.pdf].

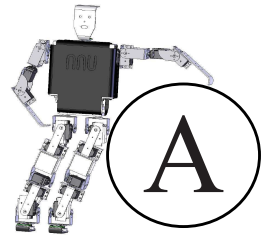
[Yamaguchi et al., 1999] Yamaguchi, J., Soga, E., Inoue, S., and Takanishi, A. (1999). Development of a bipedal humanoid robot - control method of whole body cooperative dynamic biped walking -. *International Conference on Robotics & Automation, IEEE*. [CD-ROM, 2007, literature/control/Whole_body_cooperative.pdf].

[You et al., 2004] You, B.-J., Oh, S.-R., and Choi, Y. (2004). On the stability of indirect zmp controller for biped robot systems. *IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2.* [CD-ROM, 2007, literature/modeling/zmp_calc2.pdf].

[Zhao et al., 1997] Zhao, W., Wu, G., and III, H. J. S. (1997). Closed-form kinematics for a spartial closed-chain mechanism modeling biped stance. *Mechanical Machine Theory Vol. 33 No. 4, Elsevier*. [CD-ROM, 2007, literature/modeling/biped_dynamics_doubble_support.pdf].

[Zheng and Luh, 1988] Zheng, Y. F. and Luh, J. (1988). Optimal load distribution for two industrial robots handling a single object. *IEEE International Conference on Robotics and Automation & Systems, IEEE*. [CD-ROM, 2007, literature/-modeling/load_distribution_of_two_arms.pdf].

Servo Controller Board



The control signal is generated by a servo driver board, *Pololu Micro Serial Servo Controller* [Robotics and Electronics, 2005], acquired from Pololu Robotics and Electronics. It is possible to connect up to eight servo motors to one servo board. To increase the possible update rate, it has been chosen to use a separate serial line for each of the three servo controller boards. The input to the servo board is a serial bit-stream with TTL levels. Up to 16 servo boards can be connected to the same serial input signal, enabling the user to control up to 128 servo motors with only one serial line. Thus, three servo boards are sufficient for controlling all 21 servos on the biped robot. Each board has a size of only $23\,\mathrm{mm} \times 23\,\mathrm{mm}$, which makes them well suited for this project because of the limited space for hardware.

The supply voltage should be within $5\,\mathrm{V}$ and $16\,\mathrm{V}$, and the current consumption for one board is $5\,\mathrm{mA}$ in average. A picture of the board is shown in Figure A.1.

A.1 Setup of the Servo Board

The boards operate in two different modes: Mini SSC II mode and Pololu mode. In this configuration Pololu mode is used, as this supports a baud rate up to $38,400\,\mathrm{baud}$, whereas Mini SSC II mode only supports a baud rate of $2,400\,\mathrm{baud}$ or $9,600\,\mathrm{baud}$. Pololu mode also enables some more advanced features, e.g. setting the turning rate, altering the neutral position as well as the turning direction of the servos. The Pololu mode is set by removing the jumper in the top left corner of the picture in Figure A.1.

The power supply to the servos is connected through a common supply connector on the board, which is separate from the power supply to the board itself. Depending on the load of the servos, the current consumption will vary from around $200\,\mathrm{mA}$ to around $5\,\mathrm{A}$. If several servos are stalling simultaneously, the current consumption can exceed $10\,\mathrm{A}$, and measures should be taken to avoid this.´

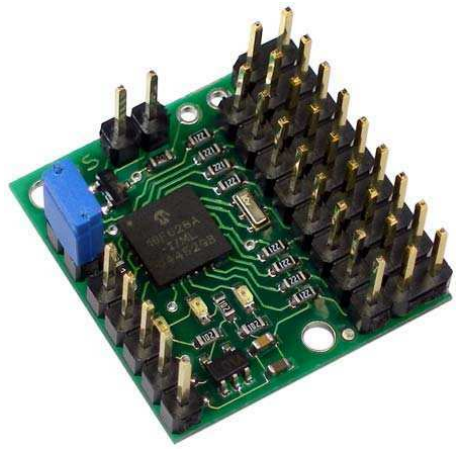


Figure A.1: *The Pololu Micro Serial Servo Controller. The connectors for the eight servo motors are placed on the right side. The jumper in the top left corner should be removed to set the board to "Pololu mode".*

A.2 Controlling the Servo Motors

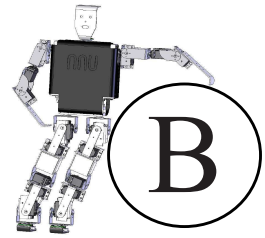
To control the servo motors, the following protocol is used. A string of five or six bytes is transmitted to the boards:

<startbyte>	<device ID>	<command>	<servo number>	<data 1>	<data 2>

- <startbyte>: Always 128.
- <device ID>: 1 for the Pololu Micro Serial Servo Controller.
- <command>: One of six different commands to send. An integer between 0 and 5, both included. With this byte different parameters can be set e.g. if the servo should be velocity or position controlled, the turning rate, the size of the input, change of neutral position etc. For specific information consult the manual for the servo controller board
  [Robotics and Electronics, 2005].
- <servo number>: The number of the servo motor to control - in this case an integer between 0 and 20, both included.
- <data 1>: First data byte. An integer between 0 and 127, both included. The range does not go from 0 to 255 due to the reason that the first bit always should be set to zero. This means that only 7 bits can be used to set the value of the byte.
- <data 2>: Second data byte (not used for <command> = 0, 1, and 2). An integer between 0 and 127, both included.

An elaboration of how to retrieve the potentiometer measurements from the servo motors can be seen in Appendix B.

FEEDBACK FROM SERVO MOTORS



This appendix will concern the extraction of the position measurement from potentiometers inside the servo motor. First it is explained how the measurement is physically extracted from the servo motor, next the measurement circuit, including a low-pass filtering, is designed.

B.1 Extracting the Potentiometer Measurement

One great advantage of the HSR-5995TG servo motor is that it is possible to reach the potentiometer inside the servo motor, allowing feedback from the servo motor. By removing the back plate, as seen in Figure B.1(a), the PCB (Printed Circuit Board) containing the internal control circuit of the servo motor can be reached. If the three solderings marked in the figure are removed, the PCB can be lifted, providing direct access to the potentiometer as seen in Figure B.1(b). A wire is soldered to the wiper and lead out of the servo motor case, and position feedback of the servo motor is now provided. This procedure is carried out for all 12 servo motors of the type HSR-5995TG, placed in the legs.

It is now possible to measure the position of the servo motor, which was tested by applying a sine wave as an input. The measured signal from the potentiometer can be seen in Figure B.2(a). The signal contains a significant level of noise, and in order to get a measurement of this noise the signal-to-noise power ratio (SNR) is calculated from [Shanmugan and Breipohl, 1988][p. 330]:

$$\text{SNR} = \frac{E\{S^2(t)\}}{E\{N^2(t)\}} \tag{B.1}$$

where $S(t)$ is the input signal, and $N(t)$ is the noise superimposed on the signal $S(t)$. In practice Equation (B.1) is implemented like:

(a) Solderings (b) Potentiometer

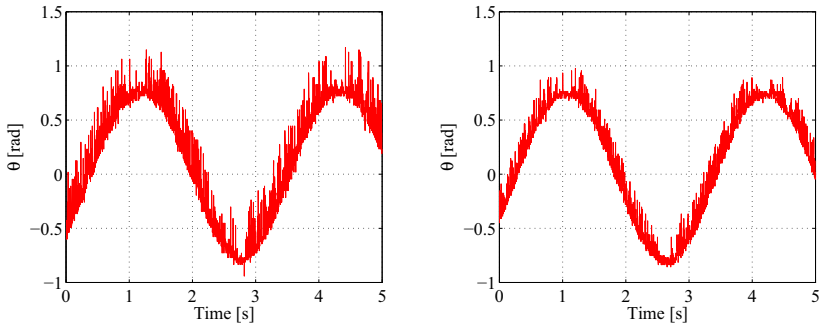
Figure B.1: *When the back plate of the HSR-5995TG servo motor is moved the printed circuit board can be reached. The three solderings marked in (a) are connected to the motor. If the PCB is moved, there is direct access to the potentiometer, providing the position feedback to the internal control circuit in the servo motor. The potentiometer is marked in (b) with a red circle.*

$$\text{SNR} = \frac{\sum_{\text{k=1}}^{N}(S[k])^2}{\sum_{\text{k=1}}^{N}(M[k] - S[k])^2} \tag{B.2}$$

where M[k] is the measured signal, N is the number of samples, in this case 2000. From Equation (B.2) the SNR of the measurement illustrated in Figure B.2(a) is calculated to be 14.1. In this section the SNR will not be used as a value itself, but instead as a benchmark.

The relative large noise contribution on the signal is most likely generated by the servo motor, which generates noise while turning. Furthermore the signal wire runs close to the discrete control signal from the servo motor driver boards. In order to shield from this imposed noise, a wire with a grounded screen is used for the position measurement from the servo motor. This improves the measurements, as seen in Figure B.2(b), and the SNR can now be calculated to 37.3, which is an improvement of 165%.

To further remove some of the noise, the circuit in Figure B.3 is used. The circuit is a common mode rejection filter, designed to remove noise which is imposed on both signals. As seen in the figure, the signal is measured relative to a virtual ground. This virtual ground is constructed by inserting voltage divider, R_1 and R_2, inside the servo motor case. The resistors are chosen so that $R_1 = R_2$ and $R_1 + R_2 = R_\text{p}$. Meeting the last requirement ensures that the same current runs in the two branches, and thereby the surrounding noise should have approximately the same influence. This is impor-



(a) Unscreened, SNR = 14.1 (b) Screened, SNR = 37.3

Figure B.2: *Measurement from the potentiometer, where a sine wave is given as a reference for the servo motor. (a) shows the raw measurements, no filtering, no screen etc. In (b) the signal wire is screened.*

tant since the AMP04 [Analog Devices, 2000] is an instrumental amplifier, specially designed to remove the noise which exist on both channels, as it has a high common mode rejection.

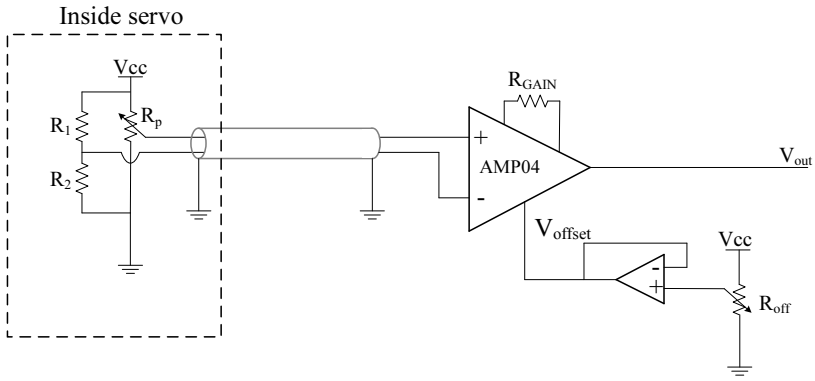


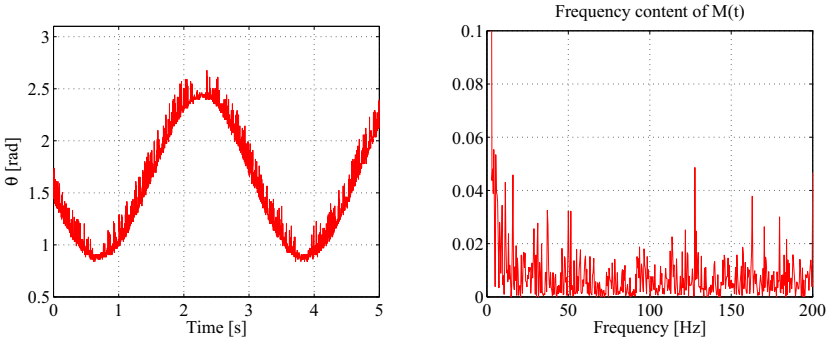
Figure B.3: *The measurement test circuit for the servo motor potentiometer.*

The virtual ground and the measured signal is led into the two input terminals of the instrumentation amplifier, AMP04. The gain of the AMP04 is set with R_{GAIN}. Since the amplitude of the signal is sufficient, unit gain is utilized. It is desired to represent both negative and positive values with a unipolar signal at the output, since the input range on the A/D converter of the on-board computer is $0\,\text{V} - 3.3\,\text{V}$ [Cirrus Logic, 2004]. This can be obtained by using the offset input, V_{offset}, on the AMP04. Now the output, V_{out}, is given as:

$$V_{\text{out}} = V_{\text{in+}} - V_{\text{in-}} + V_{\text{offset}} \tag{B.3}$$

where V_{offset} is chosen to $\frac{1}{2}V_{\text{cc}}$. The advantage of this circuit is that V_{out} is found relative to a virtual ground that is imposed by the same noise as the signal itself. Thus in

a best case outcome, the only difference between the two signals should be the signal corresponding to the position of the servo motor. However in practice this is not the case. Applying the same sine wave as in the previous cases, and measuring V_{out} gives the result shown in Figure B.4(a). The SNR is now calculated to be 43.1, an improvement of $15\,\%$, compared to the case where the signal is just shielded.



(a) Filtered, SNR $= 43.1$ (b) FFT

Figure B.4: *In (a) the servo motor is applied a sine wave and the position output is filtered through an instrumentation amplifier. (b) shows an FFT of the signal from Figure B.2(b).*

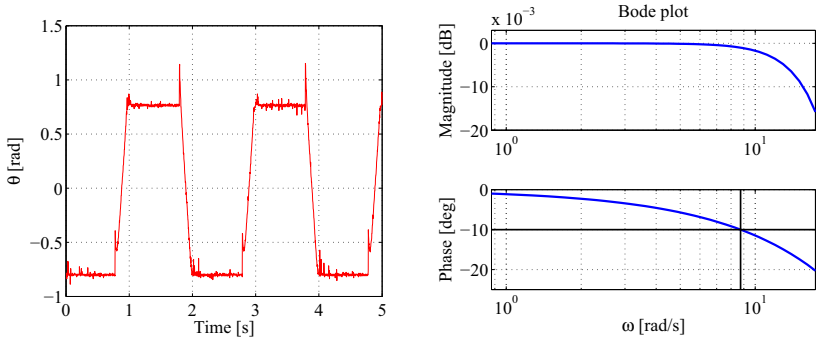
Considering the improvement in relation to the large number of components that should be used, if this circuit is incorporated into all 12 position measurements, it is chosen not to implement the AMP04 circuit.

To investigate the noise contents in the signal, a Fast Fourier Transform (FFT) is made on the noise signal, given as $N[k] = M[k] - S[k]$, the FFT can be seen in Figure B.4(b). As the signal is distributed rather smoothly throughout the frequency span, the noise is considered as white gausian noise. Thus, applying a filter to the signal, could improve the SNR significantly.

B.2 Filtering the Potentiometer Signal

From Table 5.1 on page 38 it is given that the maximum operating speed, ω_{max}, is $8.73\,\text{rad/s}$. Therefore the highest frequency to occur in the signal, except from noise, must be ω_{max}. However ω_{max} will most likely occur, when the servo motor is driven in position reference mode. In this mode a position reference is given to the servo motor, and an internal proportional controller ensures that the shaft is moved to the right position, therefore a position error above a certain level results in the servo motor turning with ω_{max}. This can be seen in Figure B.5(a), where the servo motor is given a new position reference in intervals of one second, and the shaft is turning with an angular velocity of ω_{max}.

Figure B.5(a) illustrates a positive property of the system, namely that the noise level is reduced when the servo motor is not turning. In the case, where the servo motor is



(a) A measurement from the potentiometer.

(b) A bode plot of the filter.

Figure B.5: *Measurement of the potentiometer signal and a bode plot of the suggested filter. In Figure B.5(a) the servo motor is given a reference of* $-\pi/4\,\mathrm{rad}$ *and* $\pi/4\,\mathrm{rad}$ *in intervals of* $1\,\mathrm{s}$. *Note that ind Figure B.5(b) the phase delay at* ω_{max} *(*$\omega = 8.73\,\mathrm{rad/s}$*) is equal to* θ_{delay} *(*$-10\,\mathrm{deg.}$*).*

given a sine wave as a reference, as in Figure B.2(a), it is turning all the time, causing a large level of noise. This noise is only present when the shaft rotates.

B.3 Design of the Filter

It is desired to design a filter for the measurements from the potentiometer of the servo motor, which should remove the noise but introduce minimum phase delay. The design will consist of a trade off between these two properties. Since the noise is considered white, and the highest frequency of the signal is limited to ω_{max}, it is evaluated that a second order filter will be sufficient. From A it is known that the servo motors can maximum be updated with a frequency of $50\,\mathrm{Hz}$. Therefore it is chosen to design the filter so that the measurement will not be delayed more than $t_{\mathrm{max}} = 20\,\mathrm{ms}$, which corresponds to one sample.

The second order filter is given as [Huelsman, 1993][p. 249]:

$$\frac{V_{\mathrm{o}}(s)}{V_{\mathrm{i}}(s)} = \frac{H_0\omega_{\mathrm{n}}^2}{s^2 + (\omega_{\mathrm{n}}/Q)s + \omega_{\mathrm{n}}^2} \tag{B.4}$$

where H_0 is the DC-gain, ω_{n} is the undamped natural frequency, and Q is a quality factor defining the sharpness of the peak occurring at ω_{n}. The input to the filter is in the range $0\,\mathrm{V} - 3.2\,\mathrm{V}$, and the output should be in the range $0\,\mathrm{V} - 3.3\,\mathrm{V}$, to comply with the AD converter of the on-board computer [Cirrus Logic, 2004][p. 7]. Therefore the filter is selected to be a unit gain filter ($H_0 = 1$). Since no more than $3.3\,\mathrm{V}$ is allowed on the output, it is crucial that no frequencies are amplified by more than a factor of one. Therefore Q is selected to $\frac{\sqrt{2}}{2}$, resulting in a Butterworth characteristic, with no overshoot at ω_n. The only design parameter left is ω_{n}, which should be selected to overcome the requirement of a maximum delay of t_{max} for $\omega \leq \omega_{\mathrm{max}}$. It is chosen to design the filter so that the frequency ω_{max} is delayed exactly $20\,\mathrm{ms}$ corresponding to the time of one sample.

A time delay for a certain frequency can be translated into a phase delay, θ_{delay}:

$$\theta_{\text{delay}} = \omega_{\text{max}} t_{\text{max}} = 8.73\,\text{rad/s}\;20\,\text{ms} = 0.1746\,\text{rad} \tag{B.5}$$

which is approximately $10\,\text{deg}$. Substituting $s = j\omega$ in Equation (B.4) gives:

$$\frac{V_{\text{o}}(j\omega)}{V_{\text{i}}(j\omega)} = \frac{H_0\omega_{\text{n}}^2}{(j\omega)^2 + (\omega_{\text{n}}/Q)(j\omega) + \omega_{\text{n}}^2} = \frac{H_0\omega_{\text{n}}^2}{\omega_{\text{n}}^2 - \omega^2 + j(\omega_{\text{n}}/Q)\omega} \tag{B.6}$$

inserting $\omega = \omega_{\text{max}}$ and separating the real and imaginary part gives:

$$\frac{V_{\text{o}}(j\omega)}{V_{\text{i}}(j\omega)} = \frac{H_0\omega_{\text{n}}^2(\omega_{\text{n}}^2 - \omega_{\text{max}})}{(\omega_{\text{n}}^2 - \omega_{\text{max}}^2)^2 - \omega_{\text{max}}^2\omega_{\text{n}}^2/Q^2} - j\frac{H_0\omega_{\text{n}}^3\omega_{\text{max}}/Q}{(\omega_{\text{n}}^2 - \omega_{\text{max}}^2)^2 - \omega_{\text{max}}^2\omega_{\text{n}}^2/Q^2} \tag{B.7}$$

and the phase delay of the filter at $\omega = \omega_{\text{max}}$ can now be found as:

$$\theta_{\text{delay}} = \arctan\left(\frac{\left(\frac{-H_0\omega_{\text{n}}^3\omega_{\text{max}}/Q}{(\omega_{\text{n}}^2-\omega_{\text{max}}^2)^2-\omega_{\text{max}}^2\omega_{\text{n}}^2/Q^2}\right)}{\left(\frac{H_0\omega_{\text{n}}^2(\omega_{\text{n}}^2-\omega_{\text{max}})}{(\omega_{\text{n}}^2-\omega_{\text{max}}^2)^2-\omega_{\text{max}}^2\omega_{\text{n}}^2/Q^2}\right)}\right) = \text{atan}\left(\frac{-\omega_{\text{max}}\omega_{\text{n}}/Q}{\omega_{\text{n}}^2 - \omega_{\text{max}}^2}\right) \tag{B.8}$$

Which can be rewritten to:

$$0 = Q\tan(\theta_{\text{delay}})\omega_{\text{n}}^2 + \omega_{\text{max}}\omega_{\text{n}} - \omega_{\text{max}}^2 Q\text{tan}(\theta_{\text{delay}}) \tag{B.9}$$

Solving this second order equation results in a natural undamped frequency of:

$$\omega_{\text{n}} = 71.06\,\text{rad/s} \tag{B.10}$$

Setting $H_0 = 1$, $Q = \frac{\sqrt{2}}{2}$ and $\omega_{\text{n}} = 71.06\,\text{rad/s}$ results in following filter:

$$\frac{V_{\text{o}}(s)}{V_{\text{i}}(s)} = \frac{5050}{s^2 + 100.5s + 5050} \tag{B.11}$$

A bode plot of the filter can be seen in Figure B.5(b). Note that the phase delay at ω_{max} is $10\,\text{deg}$.

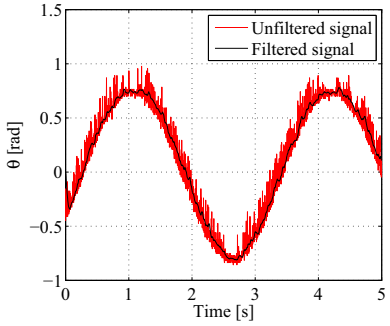


Figure B.6: *The measured signal before and after applying the filter implemented in MATLAB, SNR* $= 146.9$.

The filter is implemented in MATLAB, and the measured signal, illustrated in Figure B.2(b), is passed through the filter. The result from this simulation can be seen in Figure B.6, and as seen the filter removes a significant proportion of the noise. The SNR is now found to be 146.9, which is a theoretical improvement of 294%. It could be chosen to use this MATLAB implementation of the filter, and run it on the on-board computer. It is however chosen to realize the filter in hardware, in order to ease the burden on the computer. The next section will concern the realization of the filter.

B.4 Realization of the Filter

It is chosen to realize the filter as a Sallen and Key low-pass filter shown in Figure B.7 [Huelsman, 1993][p. 252]. The filter in Equation (B.4) can be written as:

$$\frac{V_2(s)}{V_1(s)} = \frac{\frac{K}{R_1R_3C_2C_4}}{s^2 + s\left(\frac{1}{R_3C_4} + \frac{1}{R_1C_2} + \frac{1}{R_3C_2} - \frac{K}{R_3C_4}\right) + \frac{1}{R_1R_3C_2C_4}} \tag{B.12}$$

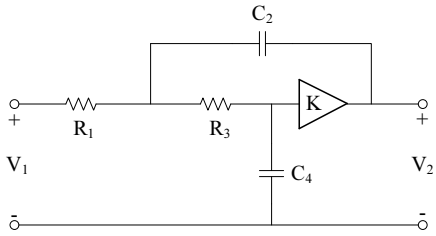


Figure B.7: *A low-pass Sallen and Key Filter.*

Comparing Equation (B.4) and Equation (B.12) gives the following:

$$\omega_\mathrm{n} = \frac{1}{\sqrt{R_1R_3C_2C_4}} \tag{B.13}$$

$$\frac{1}{Q} = \sqrt{\frac{R_3C_4}{R_1C_2}} + \sqrt{\frac{R_1C_4}{R_3C_2}} + (1-K)\sqrt{\frac{R_1C_2}{R_3C_4}} \tag{B.14}$$

$$H_0 = K \tag{B.15}$$

As stated earlier the filter should be a unity gain filter, thus $K = 1$. Furthermore two parameters are defined for the ratios of the resistor values and capacitor values respectively:

$$n = \frac{R_3}{R_1} \quad , \quad m = \frac{C_4}{C_2} \tag{B.16}$$

now Equation (B.13) and (B.14) becomes:

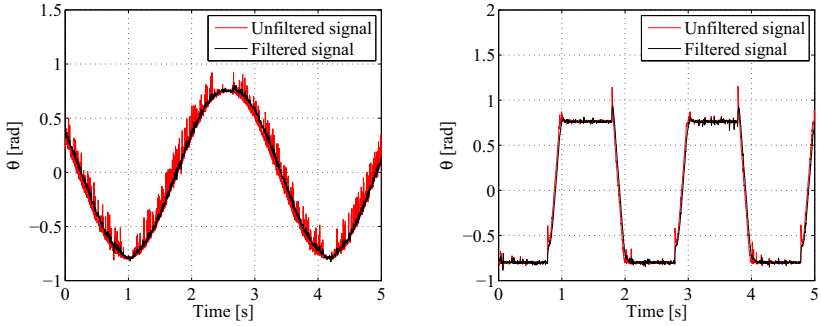
$$\omega_\mathrm{n} = \frac{1}{\sqrt{mn}R_1C_2} \tag{B.17}$$

$$\frac{1}{Q} = (n+1)\sqrt{\frac{m}{n}} \tag{B.18}$$

An expression for n can now be found to be [Huelsman, 1993][p. 254]:

$$n = \left(\frac{1}{2mQ^2} - 1\right) \pm \frac{1}{2mQ^2}\sqrt{1 - 4mQ^2} \tag{B.19}$$

From Equation (B.19) it can be seen that $m \leq \frac{1}{4Q^2}$. It is chosen to use $m = \frac{1}{4Q^2}$, thus having a ratio of the capacitor values of $m = 0.5$. Inserting Q and m in Equation (B.19), gives $n = 1$. Consequently the ratio of the resistor values is 1 ($R_1 = R_2$).



(a) Sine wave input

(b) Square wave input, SNR = 102.5

Figure B.8: *Signal before and after the filter realized in hardware. In Figure B.8(a) a sine wave is introduced, and in B.8(b) the input is the square wave signal introduced in Section B.2.*

It is chosen that $C_2 = 0.1\,\mu\text{F}$, since $m = 0.5$, $C_4 = 50\,\text{nF}$. Inserting this in Equation (B.17) together with the natural undamped frequency, $\omega_\text{n} = 71.06\,\text{rad/s}$ from Equation (B.10), gives:

$$R_1 = R_3 = \frac{1}{\sqrt{mn}\omega_\text{n} C_2} \approx 200\,\text{k}\Omega \tag{B.20}$$

Now the circuit in Figure B.7 is realized and tested. The result from this test can be seen in Figure B.8(a).

B.5 Conclusion on the Servo Motor Feedback

Even though the filter implemented in hardware is not as good as the filter tested in MATLAB (Figure B.6), it still improves the signal. Compared to the unfiltered, the SNR is improved from 37.3 to 102.5, an improvement of 175 %. The implemented filter is furthermore tested on the square wave signal, the result from this test kan be seen in Figure B.8(b). Since there is less noise on this signal, the improvement is not as distinct as for the sine wave. It can however be seen that the filter reduces some the lager spikes. Particularly the spikes orcurring right after the reference goes high (at time $= \{0.8\,\text{s}, 2.8\,\text{s}, 4.8\,\text{s}\}$) are removed.

To further emphasize the improvement an FFT is made on the filtered sine wave signal and plotted in Figure B.9 together with the FFT on the unfiltered signal from Figure

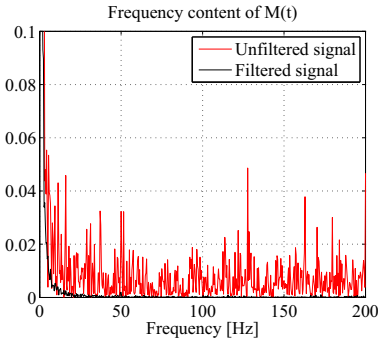


Figure B.9: *An FFT on the unfiltered and the filtered measurement.*

B.4(b). As seen the high frequency noise has been reduced significantly.

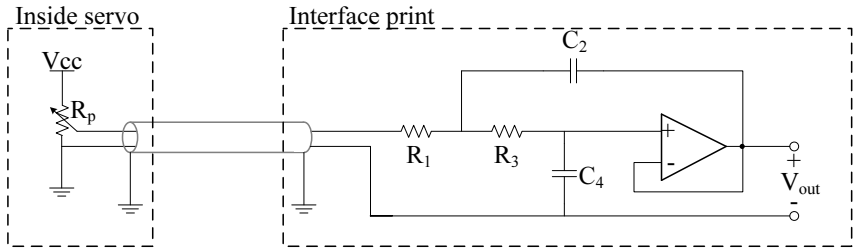
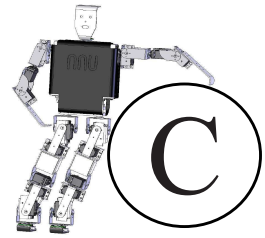


Figure B.10: *The realization of the Sallen and Key filter, including the screened wire.*

The total improvement made to the raw measurements, those of Figure B.2(a), is a change of the SNR from 14.1 to 102.5 which is an improvement of 626.9%. The final implementation of the Sallen and Key filter can be seen in Figure B.10, where it is stressed out which part that is placed inside the servo motor, and which part on the interface print. This circuit is applied for all 12 servo motors of the type HSR-5995TG.

HARDWARE WIRING DESCRIPTION



This appendix describes the wiring of the biped robot. This includes sensors, servo motors, control signals and power wiring. The section is divided into three sections. The first section describes the wiring of the two main switches located on the robot. The second section concerns the wiring of the 21 servo motors located on the biped. The third section concerns the wiring of the pressure sensors, the five DoF IMU, and the 12 potentiometer feedback signals.

C.1 Wiring of the Main Switches

To be able to choose between being powered from an external PSU or from the battery supply, two switches have been mounted on the robot. These switches also provide the possibility of charging the two batteries located inside the robot by using the same DC-connector that is used when running from the PSU. The placement of the DC-connector can be seen in Figure C.2(a). A schematic showing the wiring of the main

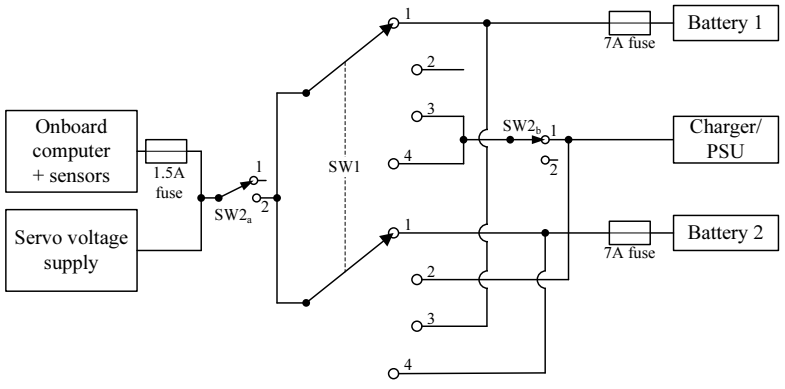
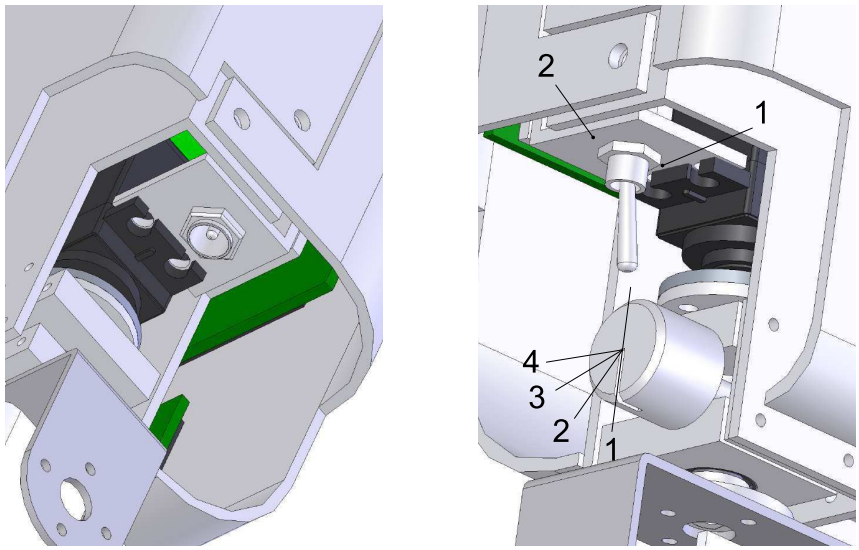


Figure C.1: *Schematic showing the wiring of the main power cords on the biped robot.*

power cords can be seen in Figure C.1. Table C.1 shows the combination of the two

main switches; SW1 and SW2.



(a) DC-connectors (b) Main switches

Figure C.2: *C.2(a) shows the placement of the DC-connector on the robot. This connector is used to recharge the batteries located inside the robot, and is used as a connector for an external power supply. C.2(a) shows the placement of the two main switches on the robot. The on/off switch is referred to as SW2, and the circular switch is referred to as SW1.*

It is important to note that a PSU never should be connected when the robot is in charging mode as this could damage the batteries and robot. The two switches $SW2_a$ and $SW2_b$, seen in Figure C.1, consist of a dual bipolar switch, meaning that $SW2_a$ and $SW2_b$ are located in the same switch. Figure C.2 shows where the two switches are mounted and where the positions of the switches are located.

To protect the batteries and other electronics located on the biped robot, three fuses have been inserted. One fuse is inserted at each battery to protect these, and one has been inserted to protect the on-board computer and the hardware interfacing the sensors. Each battery is protected by a 7 A fuse, and a 1.5 A fuse is mounted to protect the electronics. To identify the two batteries, these have been colour coded inside the robot. The wire, marked with a green label is connected to battery one. This battery is located closest to the front of the robot. The wire marked with a yellow label is battery two. This battery is located closest to the back of the robot.

SW1	SW2	Robot state
1	1	Robot OFF (Battery 1 ‖ battery 2)
1	2	Robot ON -Running on battery (1 ‖ 2)
2	1	Robot OFF
2	2	Robot ON -Running on PSU
3	1	Robot OFF -Charging battery 1
3	2	Robot ON -Running on battery 1
4	1	Robot OFF - Charging battery 2
4	2	Robot ON -Running on battery 2

Table C.1: *Table showing all the switch combinations on the robot.*

This concludes the description of the wiring of the main power cords inside the robot. The following two sections describe the wiring of the servo motors and sensors located on the robot.

C.2 Wiring of the 21 Servo Motors

The 21 servo motors are controlled using three Pololu servo controllers [Robotics and Electronics, 2005] as explained in Section 5.3.2 on page 37. Each of the servo controllers are connected to their own serial port on the on-board computer. Table C.2 shows how the 21 servos are connected to the three servo controllers.

	Servo controller 1	Servo controller 2	Servo controller 3
Connector 0	Servo 7	Servo 13	Servo 6
Connector 1	Servo 8	Servo 14	Servo 5
Connector 2	Servo 9	Servo 15	Servo 4
Connector 3	Servo 10	Servo 16	Servo 3
Connector 4	Servo 11	Servo 17	Servo 2
Connector 5	Servo 12	Servo 18	Servo 1
Connector 6	Servo 21	Servo 19	N/C
Connector 7	N/C	Servo 20	N/C

Table C.2: *Connection of the 21 servos to the three servo controllers.*

The servo controllers are located side by side in the order 1, 2, 3 from left to right. To see where the servo motors are located on the robot, refer to Figure C.3. This figure presents the naming convention for each servo motor. The servo controllers are connected to the interface print as listed in Table C.3. The label J9 refers to the specific connector located on the interface print, and pin refers to the specific pin in that connector. See Figure C.4 on page 190 where the silk-layer for the interface print can be found.

Servo Controller	Logic level serial input	Reset pin
1 (left)	UART 1 (J9, Pin 3)	J9, Pin 4
2 (middle)	UART 2 (J9, Pin 2)	J9, Pin 4
3 (right)	UART 3 (J9, Pin 1)	J9, Pin 4

Table C.3: *The three servo controllers connection to the interface print.*

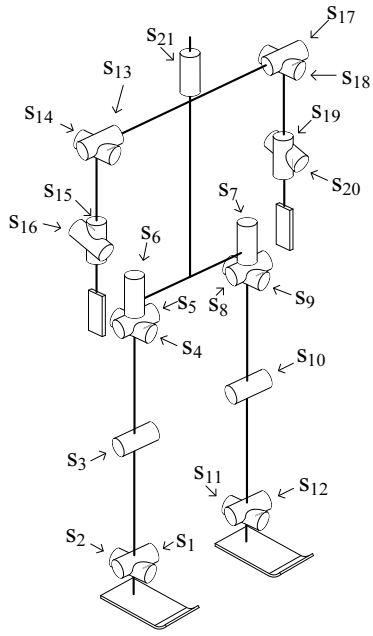


Figure C.3: *Definition of servo names, where S is short servo.*

The logic level serial input on the servo controller board is used to avoid having to implement a level converter circuit. The reset pins are all connected to the same connector on the interface print, to make it possible to reset all servo controllers at once. Each servo controller is connected to their own serial port to increase the update rate of each servo. In [Christensen et al., 2006] it was concluded that the maximum update rate for each servo, when having 21, was limited to 37 Hz due to the limited baud rate on the servo controller. By using three serial ports, it is possible to update the servo position with a frequency of 50 Hz, which is the internal update frequency of the servo controller board.

C.3 Wiring of Sensors

This section describes how the following sensors are connected on the biped robot:

- 12 potentiometers; one from each servo in the legs.
- 6 strain gauges; three on each foot.
- 1 IMU with five DoF.
- 2 battery voltage monitors.

C.3.1 Potentiometer Signals

Inside the 12 servo motors in the legs, the potentiometer signal is taken directly and then sent into an active low-pass filter. The filter is described in detail in Section B.2 on page 178. Table C.4 shows where each potentiometer signal is connected on the interface print.

	Pin 1	Pin 2	Pin 3	Pin 4
J1	Servo 7	GND	Servo 8	GND
J2	Servo 9	GND	Servo 10	GND
J3	Servo 11	GND	Servo 12	GND
J4	Servo 1	GND	Servo 2	GND
J6	Servo 3	GND	Servo 4	GND
J5	Servo 5	GND	Servo 6	GND

Table C.4: *Table showing how the potentiometer signals from the 12 strong servos, located in the legs, are connected to the interface print.*

C.3.2 Strain Gauge

The strain gauge signals are routed into the strain gauge print where the common mode noise is rejected. The output from this print, is then routed into the connectors on the interface print. A detailed description of the strain gauge print can be found in Appendix D on page 191.

Table C.5 shows how the strain gauges are connected to the strain gauge print.

	Left foot print	Right foot print
Pin 1	L_SG_1	R_SG_1
Pin 2	L_SG_2	R_SG_2
Pin 3	L_SG_3	R_SG_3

Table C.5: *Table showing how the strain gauges are connected to the strain gauge prints.*

The signals from the strain gauge prints are routed to the interface print as shown in Table C.6.

	J10: STRAIN_L	J12: STRAIN_R	J7
Pin 1	L_Force_1	R_Force_1	Vcc (+3.3 V)
Pin 2	L_Force_2	R_Force_2	
Pin 3	L_Force_3	R_Force_3	

Table C.6: *Table showing how the strain gauge prints are connected to the interface print.*

C.3.3 Inertia Measurement Unit

The Inertia Measurement Unit (IMU) is connected directly to the interface print as shown in Table C.7. A description of the IMU is found in Section 5.2.3 on page 36.

	Pin 1	Pin 2	Pin 3	Pin 4
J7	3.3 V	GND	x_{rate}	y_{rate}
J8	V_{ref}	x_{acc}	y_{acc}	z_{acc}

Table C.7: *Table showing how the IMU is connected to the interface print.*

The 3.3 V is the voltage supply for the accelerometer and gyroscope. The x_{rate} and y_{rate} refers to the angular velocities around the x- and y-axis measured by the gyroscope,

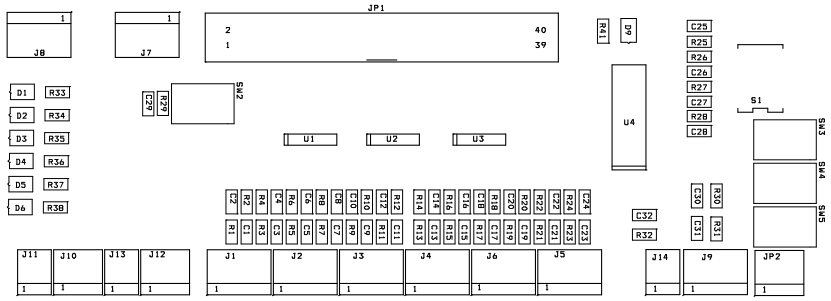


Figure C.4: *Silk screen of the PCB showing the placement of connectors*

Connector	Purpose
JP1	Main connector between the on-board computer and the interface print
JP2	UART1. Serial console for Linux
J1	Potentiometer input from servo motors 7 and 8
J2	Potentiometer input from servo motors 9 and 10
J3	Potentiometer input from servo motors 11 and 12
J4	Potentiometer input from servo motors 1 and 2
J5	Potentiometer input from servo motors 5 and 6
J6	Potentiometer input from servo motors 3 and 4
J7	Power to IMU and input from gyro scope
J8	Input from accelerometer
J9	Tx from UART1, UART2, and UART3, to the servo controller boards
J10	Signals from the strain gauges on the right foot
J11	Power connector for the strain gauge print on the right foot
J12	Signals from the strain gauges on the left foot
J13	Power connector for the strain gauge print on the left foot

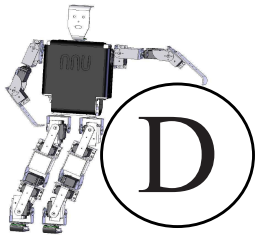
Table C.8: *What the connectors on the interface print are connected to.*

the IDG-300 [InvenSense Inc, 2006]. The signal named V_{ref} is a constant voltage of $1.23\ V$ that can be used for calibration. The three signals, x_{acc}, y_{acc}, and z_{acc}, refers to the acceleration in the x, y, and z-direction measured by the accelerometer, the ADXL330 [Devices, 2006].

C.4 Silk Screen and Schematics

Figure C.4 shows a silk screen of the interface print, where the placement of the connectors can be seen. Table C.8 shows what is connected to the interface print. The complete print layout for the interface print can be found on the enclosed CD-ROM in `hardware/schematics/interface_print.pdf`.

Strain Gauges Print



D

D.1 Measurement Circuit

To make an accurate measurement of the weight distribution on the feet, the strain gauge is placed in a Wheatstone bridge. An instrumental amplifier is used to amplify the difference of the two legs of the bridge as the instrumental amplifier has a high common mode rejection. A high common mode rejection ensures that the noise that is injected in the Wheatstone bridge is removed. The schematic for this circuit is shown in Figure D.1. As seen it has been chosen to implement two strain gauges in the bridge. R_{SG1} is placed on top of the bendable plate, while R_{SG2} is placed below. The result is that when the plate bends, one of the strain gauges will increase in resistance whereas the other will decrease. This will double the fluctuation in the output voltage of the bridge.

The AMP04 instrumental amplifier [Analog Devices, 2000] is chosen because of its high common mode rejection and because it only requires a single supply. The AMP04 has a reference input which offsets the reference. Since the AMP04 operates on a single supply the negative part of the noise can not be removed when the bridge is balanced.

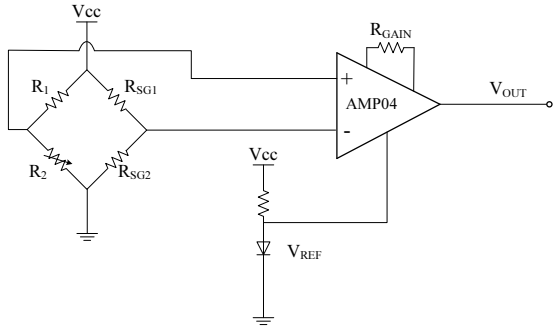


Figure D.1: *The circuit that measures the change in resistance in the strain gauge.*

To avoid this the reference input is used. Using a diode the reference is set to $0.7\,\mathrm{V}$ which is considered to be enough to cover injected noise.

The strain gauges, that will be used in the force distribution sensing system (FDSS), have a nominal resistance of $120\,\Omega$ and in order to balance the bridge R_1 and R_2 are set to $120\,\Omega$. However it can not be guaranteed that the strain gauges keep their nominal resistance after they have been attached to a strain plate. To remedy this, R_2 is made adjustable so the bridge can be balanced after assembly of the foot. Supplying the bridge with $5\,\mathrm{V}$, like with the on-board computer, it will draw approximately $42\,\mathrm{mA}$. Each foot will have three off these circuits, and the task of determining the force exerted on one foot therefore requires a minimum $126\,\mathrm{mA}$. This is considered too much and the bridge and AMP04 is instead supplied with $3.3\ \mathrm{V}$ and the required current is reduced to $3(3.3\,\mathrm{V}/120\Omega) = 82.5\,\mathrm{mA}$ for each foot. Decreasing the current flowing through the bridge also decreases the sensitivity, as the voltage difference between the two sides of the bridge is proportional to the current. This can be remedied by increasing the amplification in the AMP04. Determining the amplification requires that the mass to voltage change is known, which means determining the change in resistance of the strain gauges when a mass is placed on the strain plate. Figure D.2 shows the dimensions of the strain plate, on which a test is carried out. In the test a piece of spring steel, $1\,\mathrm{mm}$ thick, with the same dimensions as shown in Figure D.2 is mounted such that it is able to bend freely. A strain gauge is attached to the plate and a mass is placed on the pressure point and the change in resistance is recorded. In the test a weight of $0.5\,\mathrm{kg}$ was placed on the pressure point which gave a change in resistance of $\Delta R_{0.5\,\mathrm{kg}} = 0.073\,\Omega$. The slope $\Delta\Omega/\Delta m_{\mathrm{load}}$ is considered constant and determining the maximum change in resistance can be found to:

$$\Delta\Omega_{\mathrm{max}} = \frac{m_{\mathrm{min,meas}}}{0.5\,\mathrm{kg}} R_{0.5\,\mathrm{kg}} \quad = \quad 0.38\,\Omega \tag{D.1}$$

where $m_{\mathrm{min,meas}}$ is the maximum weight that the strain plate should be exposed to, which was determined in Section 4.5 on page 27 to be $2.5\,\mathrm{kg}$. Equation (D.2) stems from [Serway and Beichner, 2000, pp. 889] and can be used to determine the voltage difference of the Wheatstone bridge when $\Delta\Omega_{\mathrm{max}}$ is present in the circuit.

$$\Delta V_{\mathrm{max}} \quad = \quad \Big(\frac{R_{SG1}}{R_{SG2}+R_{SG1}} - \frac{R_2}{R_1+R_2}\Big) V_{\mathrm{s}} \tag{D.2}$$

$$\Delta V_{\mathrm{max}} \quad = \quad \Big(\frac{119.62\,\Omega}{119.62\,\Omega+120.38\,\Omega} - \frac{120\,\Omega}{120\,\Omega+120\,\Omega}\Big) 3.3\,\mathrm{V} = -5.19\,\mathrm{mV} \tag{D.3}$$

As it is the difference that is amplified the output of the AMP04 can not exceed what is the largest difference between the legs of the bridge which is $V_{\mathrm{max}} = V_{\mathrm{s}} - V_{\mathrm{ref}} = 2.6\,\mathrm{V}$. This means that the gain that should be used in the AMP04 can be determined to:

$$\mathrm{G} \quad = \quad \Big|\frac{V_{max}}{\Delta V_{\mathrm{max}}}\Big| \approx 500 \tag{D.4}$$

From the data sheet [Analog Devices, 2000] for the AMP04 the gain can be set by the following relation:

$$\mathrm{G} \quad = \quad \frac{100\,\mathrm{k}\Omega}{R_{\mathrm{GAIN}}} \tag{D.5}$$

From the above equation R_{GAIN} is determined to $200\,\Omega$.

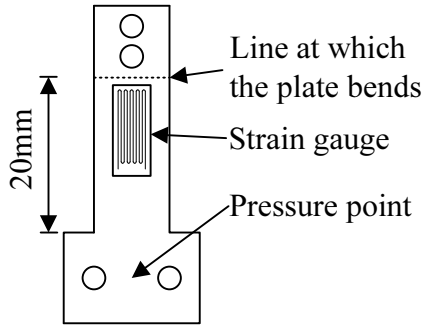


Figure D.2: *The middle strain plate on the foot.*

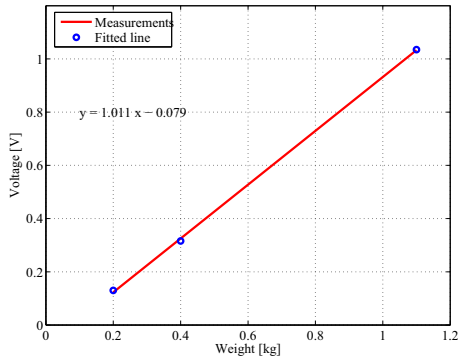


Figure D.3: *The measured data from the FDSS along with the fitted line.*

D.2 Testing the FDSS

The FDSS is subjected to a series of different masses in order to determine the transfer function. Figure D.3 shows the measured data, the line that has been fitted to the data and the transfer function which has be estimated.

The maximum weight that can be measured can be determined to:

$$m_{\text{max,meas}} = \frac{V_{\text{max}}}{1.0112} = 2.57\,\text{kg} \tag{D.6}$$

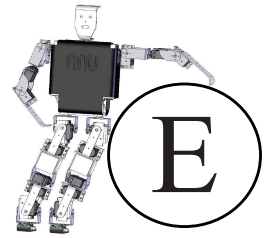
$m_{\text{max,meas}}$ is larger than the required minimum, $m_{\text{min,meas}}$, that the FDSS was designed to measure. In the previous section the gain for the AMP04 was determined using a test setup where a strain gauge was attached to $1\,\text{mm}$ spring steel. The strain gauges have been attached manually meaning that the strain gauges will not have the same characteristics after assembly. This is properly the cause of the deviation.

D.3 Implementing the FDSS

A circuit was designed which implements the FDSS. The circuit is small enough to fit on the foot and the signal can sampled directly by the ADC on the on-board computer. Tests showed that when the FDSS was not mounted on the robot the noise never exceeded $\pm 25\,\text{mV}$ and additional filtering of the signal was deemed unnecessary. How-

ever when the FDSS was implemented on the robot the noise reached $\pm 275\,\mathrm{mV}$ at a frequency of $91\,\mathrm{kHz}$. This is most likely caused by the power electronics in the servo motors since the signal wires from the FDSS run alongside the servo motors in the legs. Implementing a filter should therefore be considered.

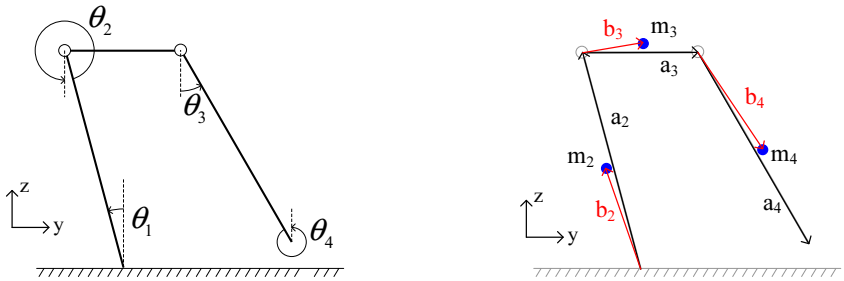
Motivating Examples for Model



This appendix includes motivating examples, to illustrate how the equations are used. As an example a 4 DoF biped robot seen in two dimensions is used.

E.1 Motivating Example for Kinematics

The fictional 4 DoF biped robot is illustrated in Figure E.1. In Figure E.1(a) the joint angles are defined and in Figure E.1(b) the link parameters are defined.



(a) Joint angles (b) Link parameters

Figure E.1: *Definition of the joint angles and link parameter for the simple fictional 4 DoF biped robot. Note that this 4 DoF model is only used for illuminative clarification.*

Utilizing Equation (7.11) and Equation (7.14) on page 77, the rotation matrices can be found for the biped robot. The orientation of the first link frame is:

$$ {}^{0}_{1}\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}_1 & -\mathrm{s}_1 \\ 0 & \mathrm{s}_1 & \mathrm{c}_1 \end{bmatrix} \tag{E.1} $$

where c_i is short for $\cos(\theta_i)$ and s_i is short for $\sin(\theta_i)$. The orientation of the second link frame is:

$$ {}^{0}_{2}\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_1c_2 - s_1s_2 & -c_1s_2 - s_1c_2 \\ 0 & s_1c_2 - c_1s_2 & -s_1s_2 + c_1c_2 \end{bmatrix} \tag{E.2} $$

and of orientation of the third link frame is:

$$ {}^{0}_{3}\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & (c_1c_2 - s_1s_2)c_3 - (c_1s_2 + s_1c_2)s_3 & (s_1s_2 - c_1c_2)s_3 - (c_1s_2 + s_1c_2)c_3 \\ 0 & (s_1c_2 - c_1s_2)c_3 - (s_1s_2 - c_1c_2)s_3 & (c_1s_2 - s_1c_2)s_3 - (s_1s_2 - c_1c_2)c_3 \end{bmatrix} \tag{E.3} $$

The generalized position vector can now be found if the rotation matrices and link parameters are combined as stated in Equation (7.20) and Equation (7.21). Since no movement in the x-direction is present in the motivating example no contribution in this direction is included in $\boldsymbol{q}$,$\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$:

$$ \boldsymbol{q} = \begin{bmatrix} c_1b_{2_y} - s_1b_{2_z} \\ -s_1a_{2_z} + c_{1+2}b_{3_y} - s_{1+2}b_{3_z} \\ -s_1a_{2_z} + c_{1+2}a_{3_y} + c_{1+2+3}b_{4_y} - s_{1+2+3}b_{4_z} \\ s_1b_{2_y} + c_1b_{2_z} \\ c_1a_{2_z} + s_{1+2}b_{3_y} + c_{1+2}b_{3_z} \\ c_1a_{2_z} + s_{1+2}a_{3_y} + s_{1+2+3}b_{4_y} + c_{1+2+3}b_{4_z} \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} \tag{E.4} $$

where there has be further simplifications in form of c_{1+2} which is equal to $\cos_{\theta_1+\theta_2}$. This simplification will be used in this example where necessary. Now the generalized velocity can be found from Equation (7.22):

$$ \dot{\boldsymbol{q}} = \begin{bmatrix} -s_1\dot{\theta}_1b_{2_y} - c_1\dot{\theta}_1b_{2_z} \\ -c_1\dot{\theta}_1a_{2_z} - s_{1+2}b_{3_y}\dot{\theta}_1 - s_{1+2}b_{3_y}\dot{\theta}_2 - c_{1+2}b_{3_z}\dot{\theta}_1 - c_{1+2}b_{3_z}\dot{\theta}_2 \\ -c_1\dot{\theta}_1a_{2_z} - s_{1+2}a_{3_y}\dot{\theta}_1 - s_{1+2}a_{3_y}\dot{\theta}_2 - s_{1+2+3}b_{4_y}\dot{\theta}_1 - s_{1+2+3}b_{4_y}\dot{\theta}_2 \\ -s_{1+2+3}b_{4_y}\dot{\theta}_3 - c_{1+2+3}b_{4_z}\dot{\theta}_1 - c_{1+2+3}b_{4_z}\dot{\theta}_2 - c_{1+2+3}b_{4_z}\dot{\theta}_3 \\ c_1\dot{\theta}_1b_{2_y} - s_1\dot{\theta}_1b_{2_z} \\ -s_1\dot{\theta}_1a_{2_z} + c_{1+2}b_{3_y}\dot{\theta}_1 + c_{1+2}b_{3_y}\dot{\theta}_2 - s_{1+2}b_{3_z}\dot{\theta}_1 - s_{1+2}b_{3_z}\dot{\theta}_2 \\ -s_1\dot{\theta}_1a_{2_z} + c_{1+2}a_{3_y}\dot{\theta}_1 + c_{1+2}a_{3_y}\dot{\theta}_2 + c_{1+2+3}b_{4_y}\dot{\theta}_1 + c_{1+2+3}b_{4_y}\dot{\theta}_2 \\ +c_{1+2+3}b_{4_y}\dot{\theta}_3 - s_{1+2+3}b_{4_z}\dot{\theta}_1 - s_{1+2+3}b_{4_z}\dot{\theta}_2 - s_{1+2+3}b_{4_z}\dot{\theta}_3 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix} \tag{E.5} $$

Also the generalized acceleration can be found from Equation (7.22):

$$
\ddot{\boldsymbol{q}} = \begin{bmatrix}
-c_1\dot{\theta}_1^2 b_{2_y} - s_1\ddot{\theta}_1 b_{2_y} + s_1\dot{\theta}_1^2 b_{2_z} - c_1\ddot{\theta}_1 b_{2_z} \\
\\
s_1\dot{\theta}_1^2 a_{2_z} - c_1\ddot{\theta}_1 a_{2_z} - b_{3_y}\dot{\theta}_1^2 c_{1+2} - b_{3_y}\ddot{\theta}_1 s_{1+2} - 2\,b_{3_y}\dot{\theta}_1\dot{\theta}_2 c_{1+2} - b_{3_y}\dot{\theta}_2^2 c_{1+2} \\
-b_{3_y}\ddot{\theta}_2 s_{1+2} + b_{3_z}\dot{\theta}_1^2 s_{1+2} - b_{3_z}\ddot{\theta}_1 c_{1+2} + 2\,b_{3_z}\dot{\theta}_1\dot{\theta}_2 s_{1+2} + b_{3_z}\dot{\theta}_2^2 s_{1+2} - b_{3_z}\ddot{\theta}_2 c_{1+2} \\
\\
-b_{4_y}\dot{\theta}_1^2 c_{1+2+3} - b_{4_y}\ddot{\theta}_1 s_{1+2+3} - b_{4_y}\dot{\theta}_2^2 c_{1+2+3} - b_{4_y}\ddot{\theta}_2 s_{1+2+3} \\
-b_{4_y}\ddot{\theta}_3 s_{1+2+3} - b_{4_y}\dot{\theta}_3^2 c_{1+2+3} - a_{3_y}\dot{\theta}_1^2 c_{1+2} - a_{3_y}\ddot{\theta}_1 s_{1+2} - a_{3_y}\dot{\theta}_2^2 c_{1+2} - a_{3_y}\ddot{\theta}_2 s_{1+2} \\
-2\,a_{3_y}\dot{\theta}_1\dot{\theta}_2 c_{1+2} + 2\,b_{4_z}\dot{\theta}_3\dot{\theta}_2 s_{1+2+3} + 2\,b_{4_z}\dot{\theta}_3\dot{\theta}_1 s_{1+2+3} + 2\,b_{4_z}\dot{\theta}_1\dot{\theta}_2 s_{1+2+3} \\
+b_{4_z}\dot{\theta}_2^2 s_{1+2+3} + b_{4_z}\dot{\theta}_1^2 s_{1+2+3} - b_{4_z}\ddot{\theta}_1 c_{1+2+3} - b_{4_z}\ddot{\theta}_2 c_{1+2+3} \\
+b_{4_z}\dot{\theta}_3^2 s_{1+2+3} - 2\,b_{4_y}\dot{\theta}_1\dot{\theta}_2 c_{1+2+3} - 2\,b_{4_y}\dot{\theta}_3\dot{\theta}_2 c_{1+2+3} - 2\,b_{4_y}\dot{\theta}_3\dot{\theta}_1 c_{1+2+3} \\
-b_{4_z}\ddot{\theta}_3 c_{1+2+3} + s_1\dot{\theta}_1^2 a_{2_z} - c_1\ddot{\theta}_1 a_{2_z} \\
\\
-s_1\dot{\theta}_1^2 b_{2_y} + c_1\ddot{\theta}_1 b_{2_y} - c_1\dot{\theta}_1^2 b_{2_z} - s_1\ddot{\theta}_1 b_{2_z} \\
\\
-c_1\dot{\theta}_1^2 a_{2_z} - s_1\ddot{\theta}_1 a_{2_z} - b_{3_y}\dot{\theta}_1^2 s_{1+2} + b_{3_y}\ddot{\theta}_1 c_{1+2} - 2\,b_{3_y}\dot{\theta}_1\dot{\theta}_2 s_{1+2} - b_{3_y}\dot{\theta}_2^2 s_{1+2} \\
+b_{3_y}\ddot{\theta}_2 c_{1+2} - b_{3_z}\dot{\theta}_1^2 c_{1+2} - b_{3_z}\ddot{\theta}_1 s_{1+2} - 2\,b_{3_z}\dot{\theta}_1\dot{\theta}_2 c_{1+2} - b_{3_z}\dot{\theta}_2^2 c_{1+2} - b_{3_z}\ddot{\theta}_2 s_{1+2} \\
\\
-2\,b_{4_y}\dot{\theta}_3\dot{\theta}_1 s_{1+2+3} - 2\,b_{4_y}\dot{\theta}_1\dot{\theta}_2 s_{1+2+3} - b_{4_y}\dot{\theta}_2^2 s_{1+2+3} \\
-b_{4_y}\dot{\theta}_1^2 s_{1+2+3} - 2\,b_{4_y}\dot{\theta}_3\dot{\theta}_2 s_{1+2+3} - b_{4_y}\dot{\theta}_3^2 s_{1+2+3} + b_{4_y}\ddot{\theta}_2 c_{1+2+3} + b_{4_y}\ddot{\theta}_1 c_{1+2+3} \\
+b_{4_y}\ddot{\theta}_3 c_{1+2+3} - 2\,b_{4_z}\dot{\theta}_1\dot{\theta}_2 c_{1+2+3} - 2\,b_{4_z}\dot{\theta}_3\dot{\theta}_1 c_{1+2+3} - b_{4_z}\ddot{\theta}_1 s_{1+2+3} \\
-b_{4_z}\dot{\theta}_2^2 c_{1+2+3} - 2\,a_{3_y}\dot{\theta}_1\dot{\theta}_2 s_{1+2} - b_{4_z}\ddot{\theta}_2 s_{1+2+3} - b_{4_z}\ddot{\theta}_3 s_{1+2+3} - b_{4_z}\dot{\theta}_3^2 c_{1+2+3} \\
-2\,b_{4_z}\dot{\theta}_3\dot{\theta}_2 c_{1+2+3} - c_1\dot{\theta}_1^2 a_{2_z} - s_1\ddot{\theta}_1 a_{2_z} - a_{3_y}\dot{\theta}_1^2 s_{1+2} + a_{3_y}\ddot{\theta}_1 c_{1+2} \\
-a_{3_y}\dot{\theta}_2^2 s_{1+2} + a_{3_y}\ddot{\theta}_2 c_{1+2} - b_{4_z}\dot{\theta}_1^2 c_{1+2+3} \\
\\
\ddot{\theta}_1 \\
\ddot{\theta}_2 \\
\ddot{\theta}_3 \\
\ddot{\theta}_4
\end{bmatrix} \tag{E.6}
$$

As seen in Equation (E.4), (E.5) and (E.6) the expressions quickly becomes very large, even for this simple example-robot with 3 links. Because of this rapidly expansion it was chosen to show the derivation of the kinematic model, with this example, while it also has be conducted for the real model.

The deviation of $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ is done in Maple. Afterward Maple can generate MATLAB code, and write to a m-file.

E.2 Motivating Example for Dynamics

To illustrate the approach used to calculate the dynamical model for the 21 joint biped robot, a motivating example is given, using a simple 3 link and 4 joint robot previously explained in Section E.1 on page 195. The example is calculated with right foot as base frame and thereby the torque $\boldsymbol{\tau}_{\mathrm{R}}$ is calculated. The calculations for $\boldsymbol{\tau}_{\mathrm{L}}$ are not shown. As the motivating example from Section E.1 on page 195 does not contain any foot links, the linear force, described in Section 7.4.3 on page 83, which needs to be added in the DSP has no link to be attached to. Therefore a virtual foot link is attached to the 4 DoF biped robot example. This virtual link has zero mass, zero length and therefore zero inertia. The position of the virtual foot links are shown in Figure E.2.

Because of the two extra links, $\boldsymbol{q}$ changes dimensions from 1×10 to 1×14 thereby making the biped robot a 5 link 4 joint robot. Now the calculations for the example can be initiated and first the energy of the 4 DoF biped robot has to be calculated to derive the Lagrangian. The kinetic energy is given by Equation (7.35) using the CoM of each link (found in the motivating example in Section E.1 on page 195)and the angular

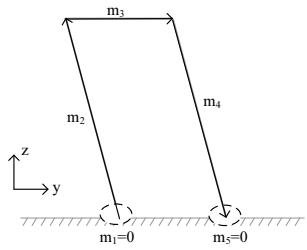


Figure E.2: *Virtual links of the simple 4 DoF biped robot.*

velocity given by Equation (7.36). The calculated kinetic energy is see from Equation (E.7):

$$K_{\mathrm{R}} = \frac{1}{2} m_2 \left(\dot{y}_2^2 + \dot{z}_2^2\right) + \frac{1}{2} \dot{\theta}_1^2 \mathrm{I}_{2_{\mathrm{x}}} + \frac{1}{2} m_3 \left(\dot{y}_3^2 + \dot{z}_3^2\right) + \frac{1}{2} \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 \mathrm{I}_{3_{\mathrm{x}}} + \frac{1}{2} \left(\dot{y}_4^2 + \dot{z}_4^2\right) m_4 + \frac{1}{2} \left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right)^2 \mathrm{I}_{4_{\mathrm{x}}} \tag{E.7}$$

Now the potential energy P is calculated using Equation (7.37).

$$P_{\mathrm{R}} = -m_2 \mathrm{g} z_2 - m_3 \mathrm{g} z_3 - m_4 \mathrm{g} z_4 \tag{E.8}$$

The Lagrangian is calculated as $L_{\mathrm{R}} = K_{\mathrm{R}} - P_{\mathrm{R}}$. The Lagrangian is differentiated with respect to the generalized coordinates $\boldsymbol{q} = [\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\theta}]^{\mathrm{T}}$ and this yields:

$$\frac{\partial L_{\mathrm{R}}}{\partial \boldsymbol{q}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -m_2 \mathrm{g} \\ -m_3 \mathrm{g} \\ -m_4 \mathrm{g} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{E.9}$$

To get the second term of Equation (7.33) the Lagrangian is differentiated with respect to the generalized velocities $\dot{\boldsymbol{q}} = [\dot{\boldsymbol{y}}, \dot{\boldsymbol{z}}, \dot{\boldsymbol{\theta}}]^{\mathrm{T}}$, which yields:

$$\frac{\partial L_{\mathrm{R}}}{\partial \dot{\boldsymbol{q}}} = \begin{bmatrix} 0 \\ 0 \\ m_2 \dot{y}_2 \\ m_3 \dot{y}_3 \\ m_4 \dot{y}_4 \\ 0 \\ 0 \\ m_2 \dot{z}_2 \\ m_3 \dot{z}_3 \\ m_4 \dot{z}_4 \\ 0 \\ \dot{\theta}_1 \mathrm{I}_{2_{\mathrm{x}}} + \left(\dot{\theta}_1 + \dot{\theta}_2\right) \mathrm{I}_{3_{\mathrm{x}}} + \left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right) \mathrm{I}_{4_{\mathrm{x}}} \\ \left(\dot{\theta}_1 + \dot{\theta}_2\right) \mathrm{I}_{3_{\mathrm{x}}} + \left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right) \mathrm{I}_{4_{\mathrm{x}}} \\ \left(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3\right) \mathrm{I}_{4_{\mathrm{x}}} \\ 0 \end{bmatrix} \tag{E.10}$$

Then the derivative of the Lagrangian with respect to the generalized velocities is differentiated with respect to time, which yields:

$$\frac{\partial}{\partial t}\frac{\partial L_{\mathrm{R}}}{\partial \dot{\boldsymbol{q}}} = \begin{bmatrix} 0 \\ m_2\ddot{y}_2 \\ m_3\ddot{y}_3 \\ m_4\ddot{y}_4 \\ 0 \\ 0 \\ m_2\ddot{z}_2 \\ m_3\ddot{z}_3 \\ m_4\ddot{z}_4 \\ 0 \\ \ddot{\theta}_1 \mathrm{I}_{2_{\mathrm{x}}} + \left(\ddot{\theta}_1 + \ddot{\theta}_2\right)\mathrm{I}_{3_{\mathrm{x}}} + \left(\ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3\right)\mathrm{I}_{4_{\mathrm{x}}} \\ \left(\ddot{\theta}_1 + \ddot{\theta}_2\right)\mathrm{I}_{3_{\mathrm{x}}} + \left(\ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3\right)\mathrm{I}_{4_{\mathrm{x}}} \\ \left(\ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3\right)\mathrm{I}_{4_{\mathrm{x}}} \\ 0 \end{bmatrix} \tag{E.11}$$

Now the generalized forces in Cartesian coordinates can be calculated by taking Equation (E.9) and then subtract Equation (E.11). This yields:

$$\boldsymbol{F}_{\mathrm{R}} = \begin{bmatrix} 0 \\ -m_2\ddot{y}_2 \\ -m_3\ddot{y}_3 \\ -m_4\ddot{y}_4 \\ 0 \\ 0 \\ -m_2\mathrm{g} - m_2\ddot{z}_2 \\ -m_3\mathrm{g} - m_3\ddot{z}_3 \\ -m_4\mathrm{g} - m_4\ddot{z}_4 \\ 0 \\ -\ddot{\theta}_1 \mathrm{I}_{2_{\mathrm{x}}} - \left(\ddot{\theta}_1 + \ddot{\theta}_2\right)\mathrm{I}_{3_{\mathrm{x}}} - \left(\ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3\right)\mathrm{I}_{4_{\mathrm{x}}} \\ -\left(\ddot{\theta}_1 + \ddot{\theta}_2\right)\mathrm{I}_{3_{\mathrm{x}}} - \left(\ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3\right)\mathrm{I}_{4_{\mathrm{x}}} \\ -\left(\ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3\right)\mathrm{I}_{4_{\mathrm{x}}} \\ 0 \end{bmatrix} \tag{E.12}$$

To map the generalized forces to joint space, the Jacobian has to be calculated. By using Equation (7.41) the Jacobian can be calculated, and this yields:

$$\boldsymbol{J}_{\mathrm{R}} = [\boldsymbol{J}_{\mathrm{R1}}, \boldsymbol{J}_{\mathrm{R2}}] \tag{E.13}$$

Where $\boldsymbol{J}_{\mathrm{R1}}$ and $\boldsymbol{J}_{\mathrm{R2}}$ are the components of $\boldsymbol{J}_{\mathrm{R}}$, which has been split.

$$\boldsymbol{J}_{\mathrm{R1}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\mathrm{s}_1 b_{2_{\mathrm{y}}} - \mathrm{c}_1 b_{2_{\mathrm{z}}} & 0 \\ -\mathrm{c}_1 a_{2_{\mathrm{z}}} - \mathrm{s}_{1+2} b_{3_{\mathrm{y}}} - \mathrm{c}_{1+2} b_{3_{\mathrm{z}}} & -\mathrm{s}_{1+2} b_{3_{\mathrm{y}}} - \mathrm{c}_{1+2} b_{3_{\mathrm{z}}} \\ -\mathrm{c}_1 a_{2_{\mathrm{z}}} - \mathrm{s}_{1+2} a_{3_{\mathrm{y}}} - \mathrm{s}_{1+2+3} b_{4_{\mathrm{y}}} - \mathrm{c}_{1+2+3} b_{4_{\mathrm{z}}} & -\mathrm{s}_{1+2} a_{3_{\mathrm{y}}} - \mathrm{s}_{1+2+3} b_{4_{\mathrm{y}}} - \mathrm{c}_{1+2+3} b_{4_{\mathrm{z}}} \\ -\mathrm{c}_1 a_{2_{\mathrm{z}}} - \mathrm{s}_{1+2} a_{3_{\mathrm{y}}} - \mathrm{c}_{1+2+3} a_{4_{\mathrm{z}}} & -\mathrm{s}_{1+2} a_{3_{\mathrm{y}}} - \mathrm{c}_{1+2+3} a_{4_{\mathrm{z}}} \\ 0 & 0 \\ \mathrm{c}_1 b_{2_{\mathrm{y}}} - \mathrm{s}_1 b_{2_{\mathrm{z}}} & 0 \\ -\mathrm{s}_1 a_{2_{\mathrm{z}}} + \mathrm{c}_{1+2} b_{3_{\mathrm{y}}} - \mathrm{s}_{1+2} b_{3_{\mathrm{z}}} & \mathrm{c}_{1+2} b_{3_{\mathrm{y}}} - \mathrm{s}_{1+2} b_{3_{\mathrm{z}}} \\ -\mathrm{s}_1 a_{2_{\mathrm{z}}} + \mathrm{c}_{1+2} a_{3_{\mathrm{y}}} + \mathrm{c}_{1+2+3} b_{4_{\mathrm{y}}} - \mathrm{s}_{1+2+3} b_{4_{\mathrm{z}}} & \mathrm{c}_{1+2} a_{3_{\mathrm{y}}} + \mathrm{c}_{1+2+3} b_{4_{\mathrm{y}}} - \mathrm{s}_{1+2+3} b_{4_{\mathrm{z}}} \\ -\mathrm{s}_1 a_{2_{\mathrm{z}}} + \mathrm{c}_{1+2} a_{3_{\mathrm{y}}} - \mathrm{s}_{1+2+3} a_{4_{\mathrm{z}}} & \mathrm{c}_{1+2} a_{3_{\mathrm{y}}} - \mathrm{s}_{1+2+3} a_{4_{\mathrm{z}}} \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{E.14}$$

$$\boldsymbol{J}_{\mathrm{R2}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\mathrm{s}_{1+2+3}b_{4_\mathrm{y}} - \mathrm{c}_{1+2+3}b_{4_\mathrm{z}} & 0 \\ -\mathrm{c}_{1+2+3}a_{4_\mathrm{z}} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \mathrm{c}_{1+2+3}b_{4_\mathrm{y}} - \mathrm{s}_{1+2+3}b_{4_\mathrm{z}} & 0 \\ -\mathrm{s}_{1+2+3}a_{4_\mathrm{z}} & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{E.15}$$

The torque in joint space is now calculated using Equation (7.40) and this yields:

$$\boldsymbol{\tau}_{\mathrm{R}} = \begin{bmatrix} -\mathrm{I}_{3_\mathrm{x}}\ddot{\theta}_1 - \mathrm{I}_{3_\mathrm{x}}\ddot{\theta}_2 - \mathrm{I}_{4_\mathrm{x}}\ddot{\theta}_2 - \mathrm{I}_{4_\mathrm{x}}\ddot{\theta}_3 - \mathrm{I}_{4_\mathrm{x}}\ddot{\theta}_1 + m_4\ddot{y}_4\mathrm{s}_{1+2}a_{3_\mathrm{y}} - m_3\mathrm{c}_{1+2}b_{3_\mathrm{y}}\mathrm{g} \\ +m_3\ddot{y}_3\mathrm{s}_{1+2}b_{3_\mathrm{y}} + m_3\ddot{y}_3\mathrm{c}_{1+2}b_{3_\mathrm{z}} + m_2\ddot{y}_2\mathrm{s}_1b_{2_\mathrm{y}} + m_2\ddot{y}_2\mathrm{c}_1b_{2_\mathrm{z}} + m_3\ddot{y}_3\mathrm{c}_1a_{2_\mathrm{z}} \\ +m_4\ddot{y}_4\mathrm{c}_1a_{2_\mathrm{z}} - \mathrm{c}_1b_{2_\mathrm{y}}m_2\mathrm{g} - \ddot{\theta}_1\mathrm{I}_{2_\mathrm{x}} - m_4\mathrm{c}_{1+2+3}b_{4_\mathrm{y}}\mathrm{g} - m_4\mathrm{c}_{1+2+3}b_{4_\mathrm{y}}\ddot{z}_4 + m_4\mathrm{s}_{1+2+3}b_{4_\mathrm{z}}\mathrm{g} \\ +m_4\mathrm{s}_{1+2+3}b_{4_\mathrm{z}}\ddot{z}_4 + m_4\ddot{y}_4\mathrm{s}_{1+2+3}b_{4_\mathrm{y}} + m_4\ddot{y}_4\mathrm{c}_{1+2+3}b_{4_\mathrm{z}} - m_3\mathrm{c}_{1+2}b_{3_\mathrm{y}}\ddot{z}_3 + m_3\mathrm{s}_{1+2}b_{3_\mathrm{z}}\mathrm{g} \\ +m_3\mathrm{s}_{1+2}b_{3_\mathrm{z}}\ddot{z}_3 - m_4\mathrm{c}_{1+2}a_{3_\mathrm{y}}\mathrm{g} - m_4\mathrm{c}_{1+2}a_{3_\mathrm{y}}\ddot{z}_4 - \mathrm{c}_1b_{2_\mathrm{y}}m_2\ddot{z}_2 + \mathrm{s}_1b_{2_\mathrm{z}}m_2\mathrm{g} + \mathrm{s}_1b_{2_\mathrm{z}}m_2\ddot{z}_2 \\ +\mathrm{s}_1a_{2_\mathrm{z}}m_3\mathrm{g} + \mathrm{s}_1a_{2_\mathrm{z}}m_3\ddot{z}_3 + \mathrm{s}_1a_{2_\mathrm{z}}m_4\mathrm{g} + \mathrm{s}_1a_{2_\mathrm{z}}m_4\ddot{z}_4 \\ \\ -\mathrm{I}_{3_\mathrm{x}}\ddot{\theta}_1 - \mathrm{I}_{3_\mathrm{x}}\ddot{\theta}_2 - \mathrm{I}_{4_\mathrm{x}}\ddot{\theta}_2 - \mathrm{I}_{4_\mathrm{x}}\ddot{\theta}_3 - \mathrm{I}_{4_\mathrm{x}}\ddot{\theta}_1 + m_4\ddot{y}_4\mathrm{s}_{1+2}a_{3_\mathrm{y}} - m_3\mathrm{c}_{1+2}b_{3_\mathrm{y}}\mathrm{g} \\ +m_3\ddot{y}_3\mathrm{s}_{1+2}b_{3_\mathrm{y}} + m_3\ddot{y}_3\mathrm{c}_{1+2}b_{3_\mathrm{z}} - m_4\mathrm{c}_{1+2+3}b_{4_\mathrm{y}}\mathrm{g} - m_4\mathrm{c}_{1+2+3}b_{4_\mathrm{y}}\ddot{z}_4 + m_4\mathrm{s}_{1+2+3}b_{4_\mathrm{z}}\mathrm{g} \\ +m_4\mathrm{s}_{1+2+3}b_{4_\mathrm{z}}\ddot{z}_4 + m_4\ddot{y}_4\mathrm{s}_{1+2+3}b_{4_\mathrm{y}} + m_4\ddot{y}_4\mathrm{c}_{1+2+3}b_{4_\mathrm{z}} - m_3\mathrm{c}_{1+2}b_{3_\mathrm{y}}\ddot{z}_3 + m_3\mathrm{s}_{1+2}b_{3_\mathrm{z}}\mathrm{g} \\ +m_3\mathrm{s}_{1+2}b_{3_\mathrm{z}}\ddot{z}_3 - m_4\mathrm{c}_{1+2}a_{3_\mathrm{y}}\mathrm{g} - m_4\mathrm{c}_{1+2}a_{3_\mathrm{y}}\ddot{z}_4 \\ \\ m_4\ddot{y}_4\mathrm{s}_{1+2+3}b_{4_\mathrm{y}} + m_4\ddot{y}_4\mathrm{c}_{1+2+3}b_{4_\mathrm{z}} - m_4\mathrm{c}_{1+2+3}b_{4_\mathrm{y}}\mathrm{g} - m_4\mathrm{c}_{1+2+3}b_{4_\mathrm{y}}\ddot{z}_4 \\ +m_4\mathrm{s}_{1+2+3}b_{4_\mathrm{z}}\mathrm{g} + m_4\mathrm{s}_{1+2+3}b_{4_\mathrm{z}}\ddot{z}_4 - \mathrm{I}_{4_\mathrm{x}}\ddot{\theta}_1 - \mathrm{I}_{4_\mathrm{x}}\ddot{\theta}_2 - \mathrm{I}_{4_\mathrm{x}}\ddot{\theta}_3 \\ \\ 0 \end{bmatrix} \tag{E.16}$$

Notice in Equation (E.16), which is only valid in SSP-R, that the torque on the fourth joint is always zero. This is an effect of the virtual joint added, which has no mass. Now the model for the SSP-R is given by Equation (E.16), and the model for the DSP phase is calculated. For simplification only the term $\rho_{\mathrm{R}}\boldsymbol{J}_{\mathrm{R}}^{\mathrm{T}}\left(\boldsymbol{f}_{\text{SSP-R}} + \zeta_{\mathrm{R}}\rho_{\mathrm{R}}m_{\mathrm{tot}}\mathrm{g}\right)$ of Equation (7.45) is calculated as the calculations for when the left foot is base frame is identical.

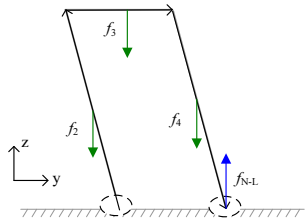


Figure E.3: *Ground reaction force added to the virtual link.*

First the force $\boldsymbol{f}_{\text{N-L}}$, depicted in Figure E.3, needs to be calculated. This yields:

$$\boldsymbol{f}_{\text{N-L}} = \rho_{\mathrm{R}}m_{\mathrm{tot}}\mathrm{g} = \rho_{\mathrm{R}}\mathrm{g}\left(m_2 + m_3 + m_4\right) \tag{E.17}$$

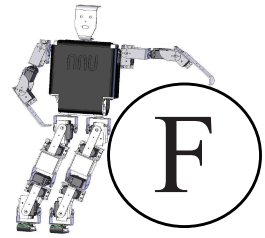
Now ρ_{L} is calculated using Equation (7.52). The distance Δy_{f} between the feet in the frontal direction can be calculated using the generalized positions calculated in the

motivating example in Section E.1 on page 195. The distance Δx_{f} between the feet in the sagittal plane is zero, but could otherwise be determined also by the generalized positions. Now since the distance between the feet in the sagittal plane is zero the first term $\frac{\Delta y_{\mathrm{f}}}{\Delta y_{\mathrm{f}}+\Delta x_{\mathrm{f}}}$ is one, the ZMP in the sagittal plane becomes zero and ρ_{R} can be calculated as:

$$\rho_{\mathrm{R}} = \frac{y_{\mathrm{ZMP,m}}}{\Delta y_{\mathrm{f}}} = \frac{\left(\frac{\mathrm{g}(m_2 y_2 + m_3 y_3 + m_4 y_4) - (m_2 \ddot{y_2} z_2 + m_3 \ddot{y_3} z_3 + m_4 \ddot{y_4} z_4)}{\mathrm{g}(m_2+m_3+m_4)}\right)}{-s_1 a_{2_z} + \mathrm{c}_{1+2} a_{3_{\mathrm{y}}} - \mathrm{s}_{1+2+3} a_{4_{\mathrm{z}}}} \tag{E.18}$$

Now combining Equations (E.12), (E.13), (E.18) and (E.17), the last part of Equation (7.45) concerning the view from the right foot can be calculated. Repeating all the previously step for left foot as base frame will yield the remaining part and the torque in the DSP, $\boldsymbol{\tau}_{\mathrm{DSP}}$, can be calculated. What is worth noticing about the motivating example is that the calculations is done for a 4 DoF robot, which can only move in the frontal plane. Furthermore is the example limited to two dimensions, since all movements in x-direction has been removed. In the actual system, the model takes all three dimensions into consideration, which leads to very high complexity and dimensions of the entire 21 DoF robot.

MODELING VERIFICATION



F.1 Introduction to Model Verification

In this section the verification of all sub-models is described. First a short introduction to the approach used to verify the models is given. Then each model is verified individually and last a verification of the complete model is given.

F.1.1 Model Verification Approach

The goal of the model verification is to verify the complete model, consisting of a number of sub-models. In order to do so, each sub-model has to be verified individually. To conduct a verification of a model, real data must be available for comparison. As the inputs and outputs of all sub-models can not be measured in the real system, some models are used in the verification of other sub-model. This way inputs that can be applied to the systems and outputs that can be measured are achieved. The model dependency is seen from Figure F.1, where the order of verification is also seen. Next a short introduction to the different verification steps is given.

First the verification of the servo motor model is conducted. This is done by applying different loads to a real servo motor, introducing a step on the input and then by reading the potentiometer output, giving the position of the servo. This measurement can then be compared with the servo motor model when applying equal load and input reference. A more thorough description of the test is given in Section F.2 on page 205, where the results of the test also can be seen.

Next the foot model is verified. This is done in a test setup were different weights can be mounted on the actual foot. Then the position of the weights is altered to apply different torques and forces to the foot. The output of the strain gauges is then measured. Applying the same torques and forces to the foot model, should then give the same output. This test can be seen in Section F.3 on page 207.

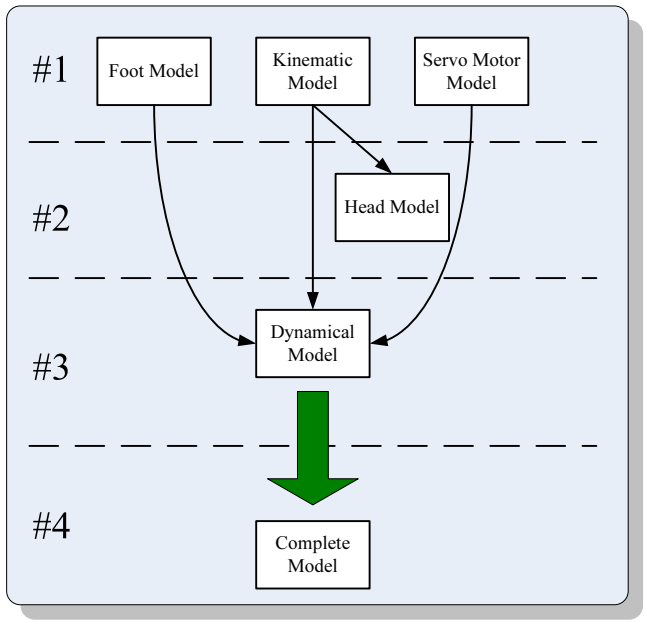


Figure F.1: *The verification of some models depend on others, which is illustrated in this figure together with the order of the test verification.*

Afterward the kinematic model is verified. To do so, extra test output from the real system has to be constructed. These are achieved by mounting an extra accelerometer in each foot, which are only used for test purpose. Now an input to the robot is given, such that each leg by turn moves the foot up and away from zero position. The input is constructed in such a way that the influence from the dynamics of the system is minimized. The acceleration in all three axis is then measured. The same input is given to the model, and the output acceleration from the model is compared to the output of the accelerometers from the robot. As the acceleration is the double differentiated of the position, the kinematic model is considered verified if the accelerations are consistent. A further description of the test and the results can be seen in Section F.4 on page 212.

As the kinematic model can be considered verified at this stage, this can be used to verify the model of the head. An input with the same characteristics as used for the kinematic model verification is constructed, and thereby minimizing the dynamical influence on the system. Then the angular velocity and acceleration of the IMU located in the head are measured and compared with the output of the head model. This can be seen in Section F.5 on page 218.

The last sub-model to be verified is the dynamical model. This test is conducted last as it relies on the verification of the servo motor model, the foot model and the kinematic model. The test is divided into two parts, one verifying the dynamics of the legs and one verifying the dynamics of the upper body including the arms. For the first test, concerning the legs, an input is constructed which emphasize the dynamics of the systems. Even though this input should reveal the system dynamics it must still be constructed in such a way that it can be applied to the real system and still maintain a stable position. As the real system is most stable in DSP, this phase is used to test

the dynamical model of the legs. Furthermore, the DSP model is a superposition of the two SSP models, thus both models will be tested. The input is then constructed such that the upper body is moved in frontal direction and the weight is thereby shifted from the left to the right foot. This input is applied to the robot and the output from the strain gauges is measured. These measurements are then compared to the output of the foot model. If a match between the measurements and the simulated output is found, then the dynamical model of the legs is considered verified, as all the other models where verified in the previous tests. The last part of the dynamical model verification is the verification of the dynamics of the upper body, including the arms. An input to the robot is constructed, such that only the arms are moved, while the robot is in DSP. The output of the strain gauges is then compared with the output of the foot model, as described before. A more full description of these two tests and the results are given in Section F.6 on page 221.

As all the sub-models are verified, the complete model can now be verified at this point. The success criteria of this test is that when the same input applied to both the robot and the model, all the sensor outputs of the robot should match the outputs of the model. An input is constructed were the robot is in DSP and all servos are moved such that all inputs are tested. Now all sensor outputs are measured and compared with the output of the model. This verification is seen in Section F.7 on page 225.

All measurements are compared with model outputs using plots of the data. Further the R^2-value given by Equation (F.1) [Shanmugan and Breipohl, 1988] is used as a benchmark for how good the fit is:

$$R^2 = 1 - \frac{\int \left(y_{\text{real}}(t) - y_{\text{model}}(t)\right)^2 dt}{\int \left(y_{\text{real}}(t) - \mu_{\text{real}}\right)^2 dt} \tag{F.1}$$

where y_{real} is the measurement, y_{model} is the simulated output and μ_{real} is the mean value of the measurement. A fit of one corresponds to a complete match between the simulated output and the measured data.

F.2 Verification of the Servo Motor Model

In this section the verification of the servo motor model is described. Figure F.2 shows the inputs and outputs of the servo motor model.

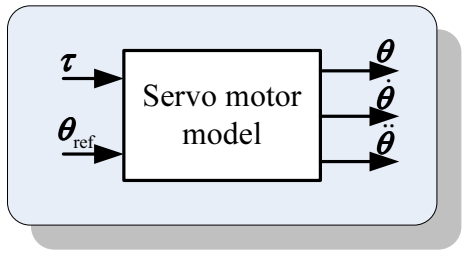


Figure F.2: *The I/O of the servo motor model.*

As seen the servo motor model takes a vector containing references to all servos, $\boldsymbol{\theta}_{\text{ref}}$, and a vector denoting the load torque on each servo, $\boldsymbol{\tau}$, as input. It returns the angular position, $\boldsymbol{\theta}$, angular velocity, $\dot{\boldsymbol{\theta}}$, and angular acceleration $\ddot{\boldsymbol{\theta}}$ of each servo.

F.2.1 Servo Motor Model Test Setup

To verify the model, a number of test is performed. The servo motor is exposed to step from $0\,\text{rad}$ to $2.1\,\text{rad}$, and is exposed to a load given as:

$$\tau_i = m_i\, g\, l \cos\left(\theta_i - 1.05\right) \tag{F.2}$$

where l is the length of the arm, which is $19.5\,\text{mm}$, θ_i is the angle of the servo motor, m_i is a mass suspended in a wire in the end of the arm, and g is the gravity acceleration. Three tests are performed, where different masses are chosen $m_1 = 0$, $m_2 = 2.383\,\text{kg}$ and $m_3 = 4.557\,\text{kg}$. This corresponds to a load torque of:

$$\tau_1 = 0\,\text{Nm} \tag{F.3}$$
$$\tau_2 = 0.456\sin(\theta_i - 1.05)\,\text{Nm} \tag{F.4}$$
$$\tau_3 = 0.873\sin(\theta_i - 1.05)\,\text{Nm} \tag{F.5}$$

It should be noted that two different type of servo's are located on the robot, and a model is derived for both of them. Therefore, this test will include a verification of both models.

F.2.2 Servo Motor Model Test Results

In Figure F.3 the raw measurements from these three tests are plotted.

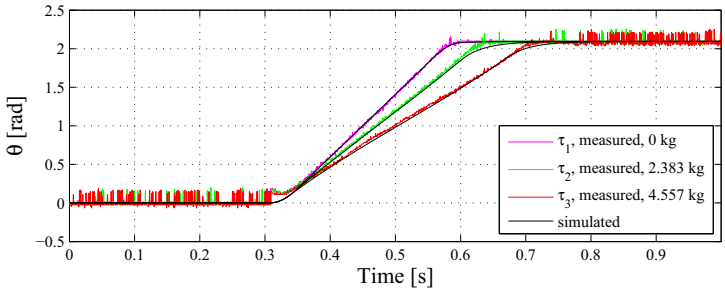


Figure F.3: *Comparison between measurements and simulated data for three different loads, on the HSR-5995TG*

Furthermore a simulation is performed on the implemented servo motor model, which is exposed to the same three loads. The result of this simulation can be seen in Figure F.3 as the three black lines. The same procedure is taken for the HS-645MG servo motor, and similar result are obtained which can be seen in Figure F.4.
The normed error can be seen in Table F.1.

F.2.3 Servo Motor Model Test Discussion

It can be seen that the output of the simulation fits the measured data, with a 0.924 to 0.999 fit of the R^2-value, for both servos. The measurement in Figure F.4 is more

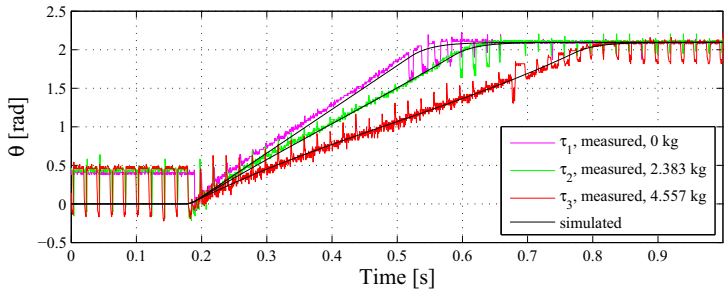


Figure F.4: *Comparison between measurements and simulated data for three different loads, on the HS-645MG.*

	$R^2_{\tau_1}$	$R^2_{\tau_2}$	$R^2_{\tau_3}$
HSR-5995TG	0.999	0.998	0.997
HS-645MG	0.936	0.937	0.924

Table F.1: *Calculated R^2-value of the servo motor model test.*

noisy than that of Figure F.3, this is because the HS-645MG is weaker, and τ_3 is close to the maximum load for this servo.

F.2.4 Servo Motor Model Test Conclusion

Due to a model fit lying between 0.924 and 0.937 for the HS-645MG and between 0.997 and 0.999 for the HSR-5995TG, it is concluded that both models are sufficiently accurate for the purpose of this thesis. The models are now considered verified.

F.3 Verification of the Foot Model

In this section the verification of the foot model documented. The derivation of the model is explained in Section 7.6.

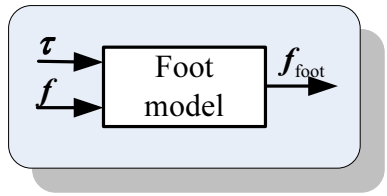
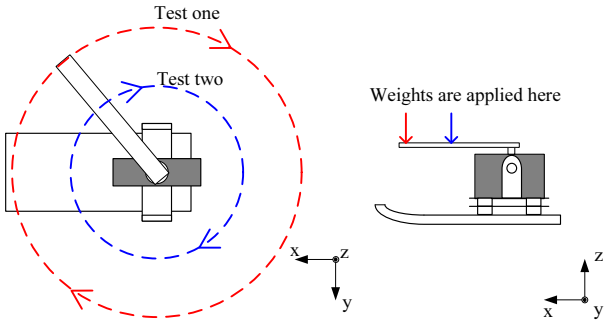


Figure F.5: *Inputs and output of the foot model. Input is the force and torques acting on the foot from the last joint of the leg. Output is the force acting on the strain gauges.*

As it can be seen from Figure F.5 the inputs to the foot model are the torques, $\boldsymbol{\tau}$, and force, $\boldsymbol{f}$, acting in the ankle joint. The output is the forces measured by the three strain gauges, $\boldsymbol{f}_{\text{foot}} = [f_{\text{S1}}\ f_{\text{S2}}\ f_{\text{S3}}]^{\text{T}}$ As the torques and force applied to the foot cannot be measured on the robot, a construction has to made which applies different known torques and forces to the foot.

F.3.1 Foot Model Test Setup

As the foot consists of three different spring steel plates mounted in such a way that the foot forces can be measured in different positions, the test setup is made such that different forces and torques can be applied. The test setup is seen from Figure F.6. It is seen that a rod is connected to a servo motor mounted in a servo bracket. Different weights can be applied with variable distance to the center of rotation of the rod. When the servo motor starts rotating, the position of the weight changes, thereby making different forces and torques acting in the ankle.



(a) Top view (b) Seen the left

Figure F.6: *F.6(a) shows the movement seen from the top, F.6(b) shows the placement of the weights from the left.*

The strain gauges are initially calibrated to zero, when the test setup is mounted, without any weights. This ensures that only the forces and torques from the weight is included in the measurements. Using Newtons second law, the force acting on the foot can be calculated. Knowing the force and the position of the weight, the torque in the frontal and sagittal direction can be calculated. Next is a description of the different tests conducted to verify the foot model:

1. Place a $60\,\mathrm{g}$ weight at the outer-most position of the rod, namely $55\,\mathrm{mm}$. Then rotate the rod, using the servo motor, $360\,\mathrm{deg}$ clockwise until reaching initial position. Then turn the rod counter clockwise $360\,\mathrm{deg}$ until initial position is reached again. This test is repeated for $160\,\mathrm{g}$ and $260\,\mathrm{g}$ weights.
2. The first test are repeated changing only the position of the weight to $41\,\mathrm{mm}$.

The input to the model is the torque around the x and y-axis, τ_x and τ_y , and the force in z-direction, f_z. These are calculated from the weights and the position of the weights, and is shown in Figure F.7.

The numeration of the strain gauges located on the foot is as shown in Figure F.8.

Next the results from the foot model test is described.

F.3.2 Foot Model Test Results

Output of the model and the measured data can be seen from Figure F.9.

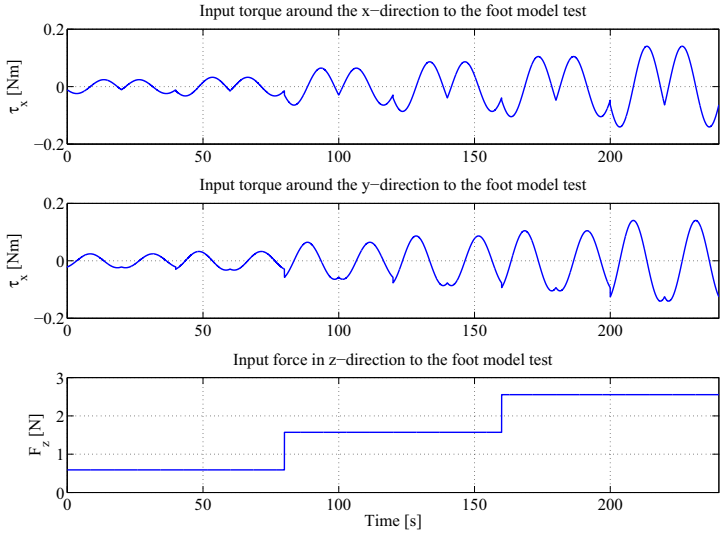


Figure F.7: *Input used in the foot model test.*

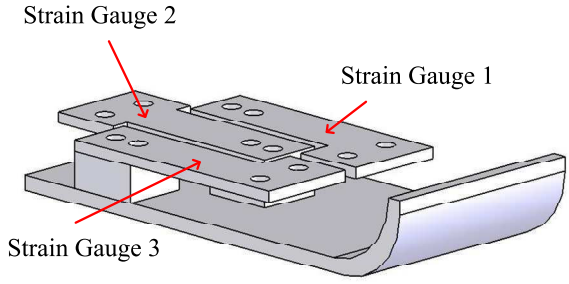


Figure F.8: *Numeration of the strain gauges on the foot.*

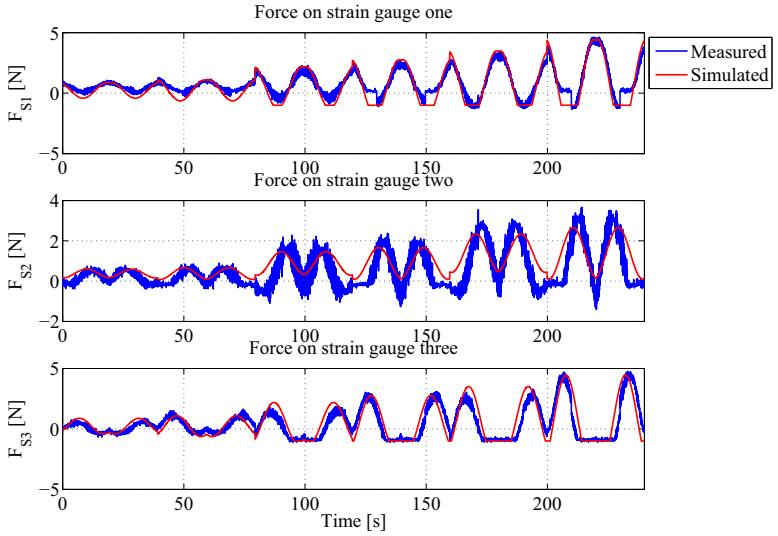


Figure F.9: *Results from the verification of the foot model. The blue lines are the measured output of the strain gauges on the real foot. The red lines are the output of the model. The uppermost figure is the output from strain gauge one, the middle is the output from strain gauge two and the lowest is the output from strain gauge three.*

R^2_{S1}	R^2_{S2}	R^2_{S3}
0.782	0.432	0.652

Table F.2: *Calculated R^2-value of the foot model test.*

The calculated R^2-value, using Equation (F.1), can be seen in Table F.2

F.3.3 Foot Model Test Discussion

As it can be seen from Table F.2 the calculated error percentage reveals a mismatch between the model and the measured data. The calculated error is the largest on strain gauge two. The main reason for the relatively large deviation of the modeling of strain gauge two is the the phase lag between the model and measurements. This phase lag is also seen from the error measured on strain gauge three, but is not as distinct on the error measured on strain gauge one.

From Figure F.9 it is seen that there are several discontinuities in both the measurements and in the output from the model. These discontinuities are caused by the way the inputs are fused together, e.g. when there is a sudden change in the weight applied the force acting on the system change abrupt. This phenomenon is also seen from the created input shown in Figure F.7.

In Figure F.9 it looks like the model do not reach the same amplitudes as the real system. This is because the point where the force and torques are applied is directly above the point where strain gauge two is positioned, in the y-direction. Thereby τ_x has no effect on this strain gauge. This can however never be the case in the real world, since it is not possible to place the point exactly. A problem like this can be amplified fur-

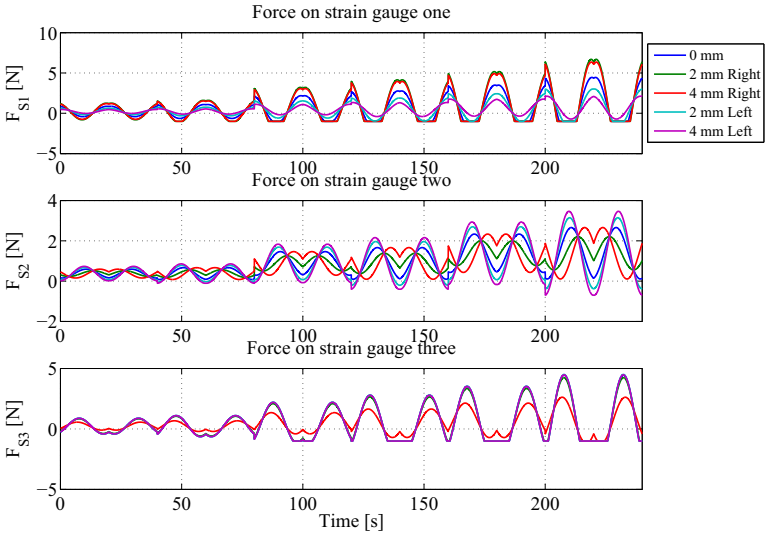


Figure F.10: *The coordinate system located in the foot was moved to the left and right in the y-axis. It is seen that his has a large impact on the model, specially on strain gauge two which is located right above the origin.*

ther, when the spring steel bends, and thereby changing the structure of the system. It is also seen that this has a smaller impact on strain gauge one and three, as these are not located directly above the point where τ_y works on the foot. To investigate how big an impact movement of the origin of the coordinate system has on the model, a plot has been created showing this. The coordinate system was moved $2\,\mathrm{mm}$ and $4\,\mathrm{mm}$ to the left and right in the y-axis. This is seen from Figure F.10. The effect of moving the coordinate origin is outspoken on specially strain gauge two.

Another reason why the amplitudes on strain gauge two does not match that of the model, could be the spring effect in the steel plates where the strain gauges are mounted. As more weight are applied to the spring steel, it will bend more and thereby amplifying the output of the strain gauge. In the model this spring effect not included. The test was conducted with $60\,\mathrm{g}$,$160\,\mathrm{g}$ and $260\,\mathrm{g}$ weights, while the robot weighs $3.7\,\mathrm{kg}$. It would have been the best to conduct the test at the same weight as the robot, but this was not possible due to the construction of the test setup. It is assessed that the amplitude difference between the measured and the model on strain gauge two will become more visible as the weight is increased.

F.3.4 Foot Model Test Conclusion

From Figure F.9 it is seen that the model tracks the input changes in the same manner as the real system. Due to the construction of the foot the R^2-value is best on strain gauge one and three. It should be noted that the width of the foot is small, with only $15\,\mathrm{mm}$ in y-direction between strain gauge one and three, which means that the torque arm is only $7.5\,\mathrm{mm}$. Thus an error in the placement of the weight of $2\,\mathrm{mm}$ would have a large impact on the output of the real system, which also is seen from Figure F.10. It

is therefore valued that the model fits the real system and the foot model is considered as verified.

F.4 Verification of the Kinematic Model

In this section the kinematic model is verified. This verification is conducted by comparing measurements from the real system with output from the kinematic model. The derivation of the model can be found in Section 7.3 on page 71. Inputs and outputs of the kinematic model are seen on Figure F.11.

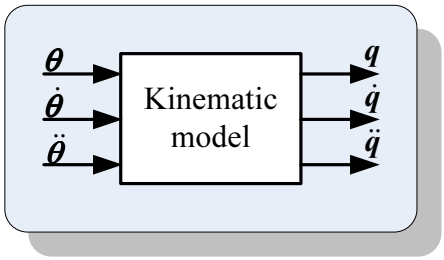


Figure F.11: *Inputs and outputs of the kinematic model. Input is given in joint space coordinates and are converted into generalized coordinates given in Cartesian space.*

It is seen that the model transforms joint space coordinates to generalized coordinates. Next is a description of the test setup, used to verify the kinematic model.

F.4.1 Kinematic Model Test Setup

The kinematics of the biped robot can be seen as four different kinematic chains. Two of these chains are seen from Figure F.12, where the first kinematic chain goes from the supporting foot to the free foot, this is seen in Figure F.12a.An other of the kinematic chain goes from the supporting foot to the left hand, this is seen from Figure F.12b. Only two of the kinematic chains are seen from Figure F.12 as the last two are similar. One of these kinematic chains goes the opposite way compared the one shown in Figure F.12a). The last kinematic chain goes from the supporting foot to the right hand.

As there are four different kinematic chains, four different test are carried out to verify the kinematic model. These are:

1. Move the right leg to a position such that all servos $\theta_1 \ldots \theta_6$ are moved, while maintaining SSP-L.

2. Move the left leg to a position such that all servos $\theta_7 \ldots \theta_{12}$ are moved, while maintaining SSP-R. This movement is animated in Figure F.13.

3. Move the right arm to a position such that all servos $\theta_{13} \ldots \theta_{16}$ are moved, while maintaining DSP. This movements is shown in Figure F.14.

4. Move the left arm to a position such that all servos $\theta_{17} \ldots \theta_{20}$ are moved, while maintaining DSP.

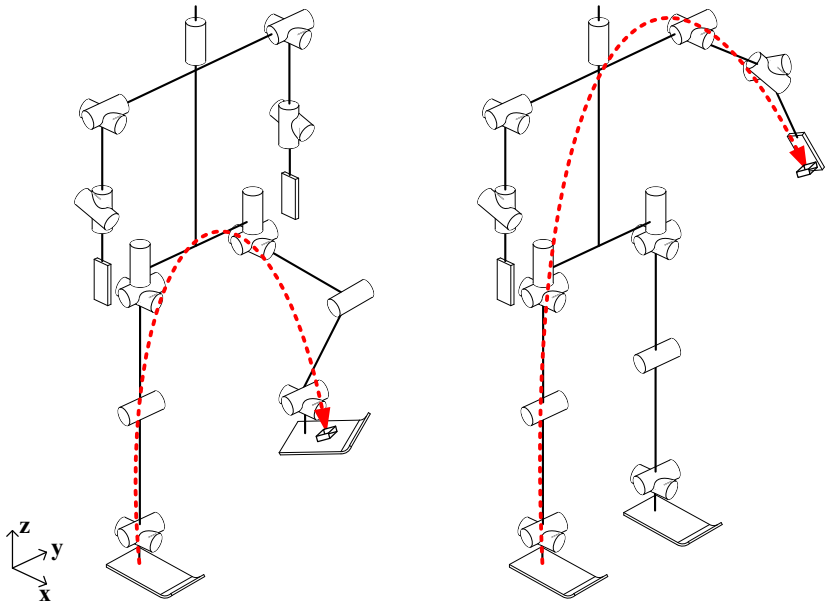


Figure F.12: *Two of the four kinematic chains are seen from this figure.* ***a)*** *The kinematic chain of the legs and the positioning of the extra accelerometer mounted on the feet.* ***b)*** *The kinematic chain of the arms and the positioning of the accelerometer mounted in the hands.*

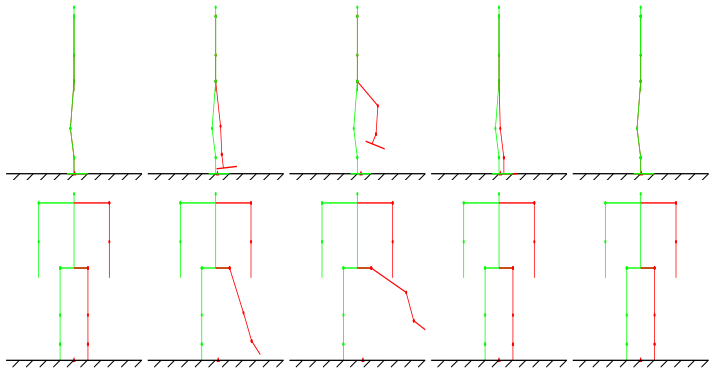


Figure F.13: *Animation of the movements of the left leg used in the verification of the kinematic model. The five figures above show the movements in the sagittal plane, while the five figures below show the movements in the frontal plane. The span of the test is* 10 s

$\Delta x_{a,h}$	$\Delta y_{a,h}$	$\Delta z_{a,h}$	$\Delta x_{a,f}$	$\Delta y_{a,f}$	$\Delta z_{a,f}$
16 mm	20 mm	61 mm	49.5 mm	12 mm	46.2 mm

Table F.3: *Placement of the temporary accelerometers shown in Figure F.15.*

The movements of the left leg used to verify the kinematic model is seen from Figure F.13. The movements are similar, but not equal, for the right leg.

The movements of the right arm used to verify the kinematic model are shown in Figure F.14. The movements for the left arm are equal to that of the right arm.

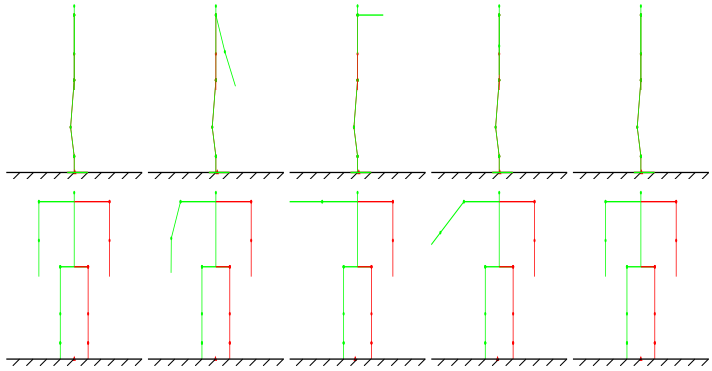


Figure F.14: *Animation of the movements of the left arm used in the verification of the kinematic model. The five figures above show the movements in the sagittal plane, while the five figures below show the movements in the frontal plane. The span of the test is* 10 s.

The movements are designed to have a velocity low enough to minimize the dynamic effect from the system, and thereby only considering the kinematics. As the real system do not have any output in either the hands or feet, which can be used for kinematic comparison with the model, four temporary accelerometers are mounted on the robot. One is located on each foot as shown in Figure F.15(a) and one is located on each hand as shown in Figure F.15(b). It is chosen to insert the accelerometers, since they can provide the necessary information, and since it is not possible to easily implement sensors that measure the Cartesian position.

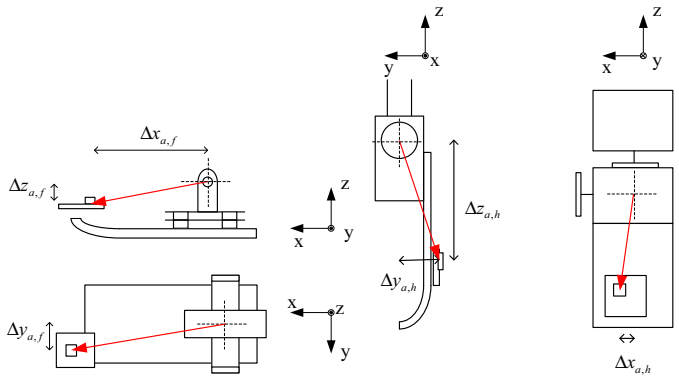
The exact placement of the accelerometers can be seen in Table F.3.

The inputs, which have been applied to the real system, are also applied to the model and a comparison is made. The results can be seen next.

F.4.2 Kinematic Model Test Results

Note that in the following results also the gravity acceleration is included.

In Figure F.16 the results from the kinematic test of the left hand are seen. The three figures contains both the measured and simulated result of the acceleration in the x, y and z-directions, $a_{x,lh}$, $a_{y,lh}$ and $a_{z,lh}$ for the left hand.



(a) Top view (b) Seen the left

Figure F.15: *Positioning of the temporary mounted accelerometers. F.15(a) Position on the left foot. Right foot is the same, where the accelerometer is positioned on the outer right side on the foot. F.15(b) Position of the accelerometer mounted on the right arm. The position of the left arm is similar.*

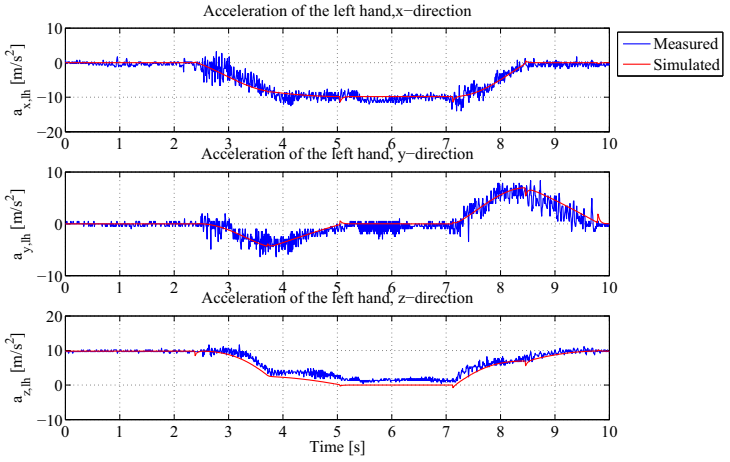


Figure F.16: *Result of the kinematic test of the left hand. The blue line is the measured output of the accelerometer and the red line is the output from the model.*

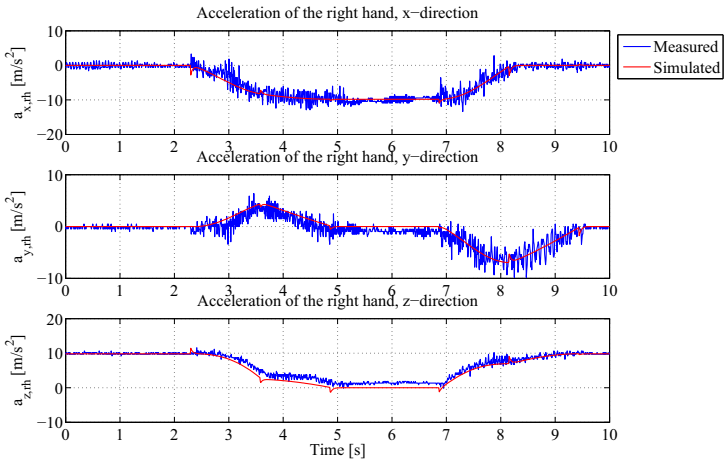


Figure F.17: *Result of the kinematic test of the right hand. The blue line is the measured output of the accelerometer and the red line is the output from the model.*

	$R^2_{a_x}$	$R^2_{a_y}$	$R^2_{a_z}$
Left hand	0.924	0.816	0.864
Right hand	0.916	0.762	0.903
Left foot	0.136	0.821	0.559
Right foot	0.170	0.879	0.672

Table F.4: *Calculated R^2-value of the kinematic model test.*

In Figure F.17 the results from the kinematic test of the right hand are seen. The three figures contains both the measured and simulated result of the acceleration in the x, y and z-directions, $a_{\text{x,rh}}$, $a_{\text{y,rh}}$ and $a_{\text{z,rh}}$ for the right hand.

In Figure F.18 the results from the kinematic test of the left foot are seen. The three figures contains both the measured and simulated result of the acceleration in the x, y and z-directions, $a_{\text{x,lf}}$, $a_{\text{y,lf}}$ and $a_{\text{z,lf}}$ for the left foot.

In Figure F.19 the results from the kinematic test of the right foot are seen. The three figures contains both the measured and simulated result of the acceleration in the x, y and z-directions, $a_{\text{x,rf}}$, $a_{\text{y,rf}}$ and $a_{\text{z,rf}}$ for the right foot.

F.4.3 Kinematic Model Test Discussion

From Table F.4 it can be seen that the calculated R^2 values of the test reveal a good fit between the simulated outputs and the measurements. Only $R^2_{\text{a}_{\text{x,lf}}}$ and $R^2_{\text{a}_{\text{x,rf}}}$, are somewhat smaller than the rest of the calculated fits. The error is however not caused to bad fit, but rather to a bad selection of the input. This is evident from the top graph in Figures F.18 and F.19, where it can be seen that the measured $a_{\text{x,lf}}$ and $a_{\text{x,lf}}$ have smaller amplitudes. The R^2 values tends to be small, if the deflection from the mean is small. This is the case in the mentioned situations.

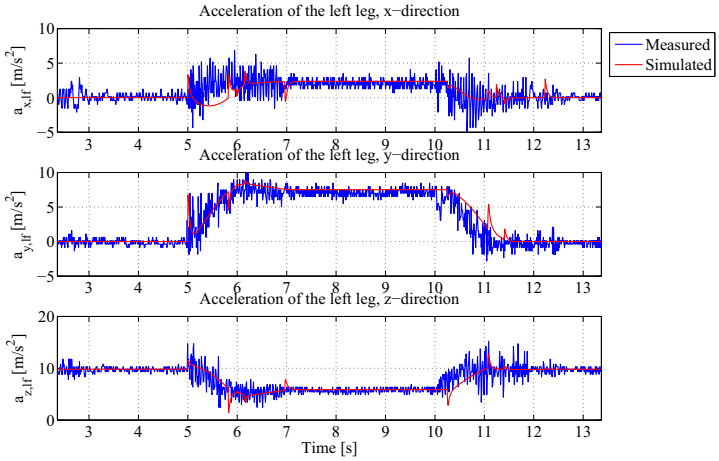


Figure F.18: *Result of the kinematic test of the left foot. The blue line is the measured output of the accelerometer and the red line is the output from the model.*

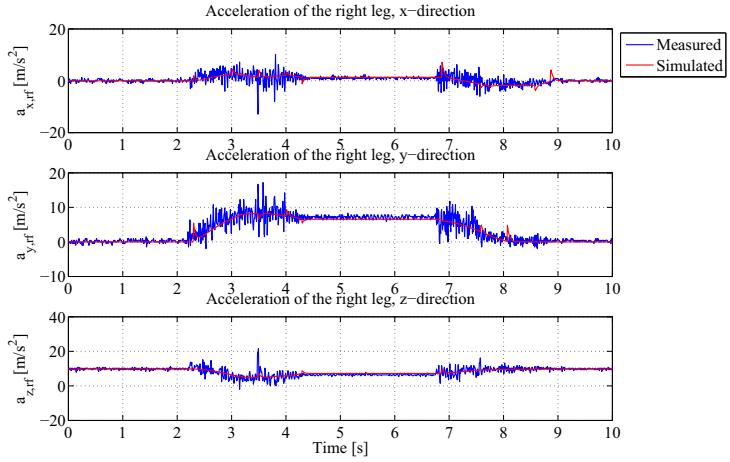


Figure F.19: *Result of the kinematic test of the right foot. The blue line is the measured output of the accelerometer and the red line is the output from the model.*

Furthermore, it is seen from Figure F.16, F.17, F.18 and F.19 that all measurements have relatively large noise contributions in periods. This is most significant during movement, where the movements of the servo motors is to fast which introduces large accelerations, which is not presented in the model. As the internal controller only contains a proportional gain, there is no integration of the error which makes the control of the position imprecise. This is a general problem of the servos and is more evident the heavier they are loaded.

What should be noticed is that the tests shows the acceleration of the limbs, rather than showing the actual position. For two reasons it is assessed that the error in acceleration would be somewhat larger than the actual error in position. The first reason is that the acceleration is the double differentiated of the position, and this differentiation would give errors from measuring the actual position. The second reason is that since the accelerometer measures the acceleration from gravity, a small initial tilt in the positioning of the accelerometer would result in an offset error when the limbs reaches final position. Such an offset error has a large influence on the calculated R^2 values.

F.4.4 Kinematic Model Test Conclusion

From Figure F.16, F.17, F.18, F.19 and the calculated R^2 values shown in Table F.4 it is seen that the kinematic model gives a good estimate of the actual accelerations of the limps of the robot. As it gives a good acceleration fit it can be concluded that the position estimate would be good as well. It can thereby be concluded that the kinematic model estimates the positions and accelerations of the limbs of the robot.

F.5 Verification of the Head Model

In this section the verification of the head model is given. The head model was earlier derived in Section 7.7. This model gives an estimate of the output of the IMU sensor located in the head of the robot. The inputs and outputs of the head model are seen from Figure F.20

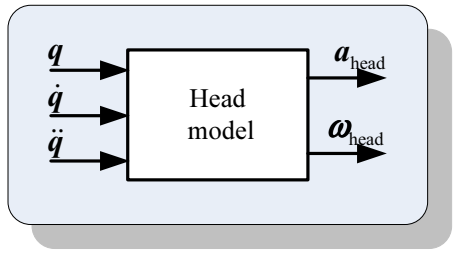


Figure F.20: *Inputs and outputs of the head model.*

The head model returns the linear acceleration, $\boldsymbol{a}_{\text{head}}$, and angular velocity, $\boldsymbol{\omega}_{\text{head}}$, of the head, when given the generalized position vector, $\boldsymbol{q}$, the generalized velocity vector, $\dot{\boldsymbol{q}}$, and the generalized acceleration vector, $\ddot{\boldsymbol{q}}$, as input. As the generalized positions, velocities and accelerations are output from the kinematic model, this model is used to verify the head model.

$R^2_{a_{x,head}}$	$R^2_{a_{y,head}}$	$R^2_{a_{z,head}}$	$R^2_{\omega_{x,head}}$	$R^2_{\omega_{x,head}}$
0.879	0.799	0.214	0.053	–

Table F.5: *Calculated R^2-values of the head model test.*

F.5.1 Head Model Test Setup

To verify the head model, an input has to be constructed which test the acceleration of the head. This means that the upper body should be rotated back, forth, and to both sides. An input as such is constructed and is illustrated in Figure F.21.

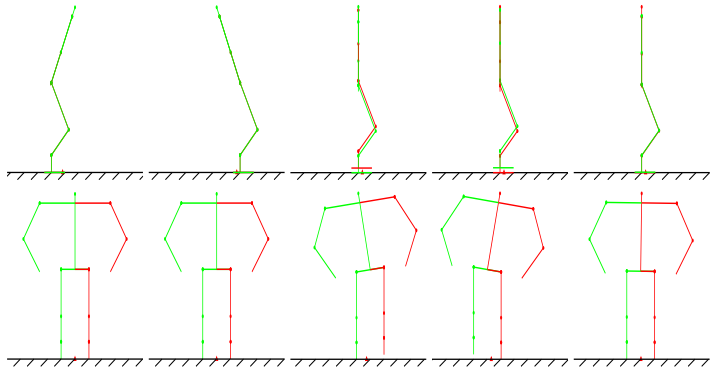


Figure F.21: *Animation of the movements used in the verification of the head model. The five figures above show the movements in the sagittal plane, while the five figures below show the movements in the frontal plane. The span of the test is* 20 s.

Using the constructed input, the torso rotates $20\,\mathrm{deg}$ back and forth, and afterward it rotates from side to side. During these rotations of the torso, an input is applied to the head, which makes it turn from side to side. When the test is conducted the output of the IMU, located on the robot, is compared with the output of the IMU in the model. These results are shown next.

F.5.2 Head Model Test Results

Figure F.22 shows the measured and simulated output of the acceleration of the IMU. The angular velocity of the IMU is seen from Figure F.23.

The calculated R^2-values of the head model test are seen from Table F.5.

F.5.3 Head Model Test Discussion

It is seen from Figure F.22 and Table F.5 that especially $a_{x,head}$ and $a_{y,head}$, have a close relation between the simulation and the measurement. Looking at the results for $a_{z,head}$, the fit is not that significant. The reason for this is that the fluctuation are small compared to the noise contribution. So if better results are desired, it is necessary to create and input that accelerates more in the z-direction. This property of bad scaling between movement at noise becomes even more outspoken for $\omega_{x,head}$ and $\omega_{y,head}$ in Figure F.23. It is not possible to see from the measurement, and the R^2-values, if

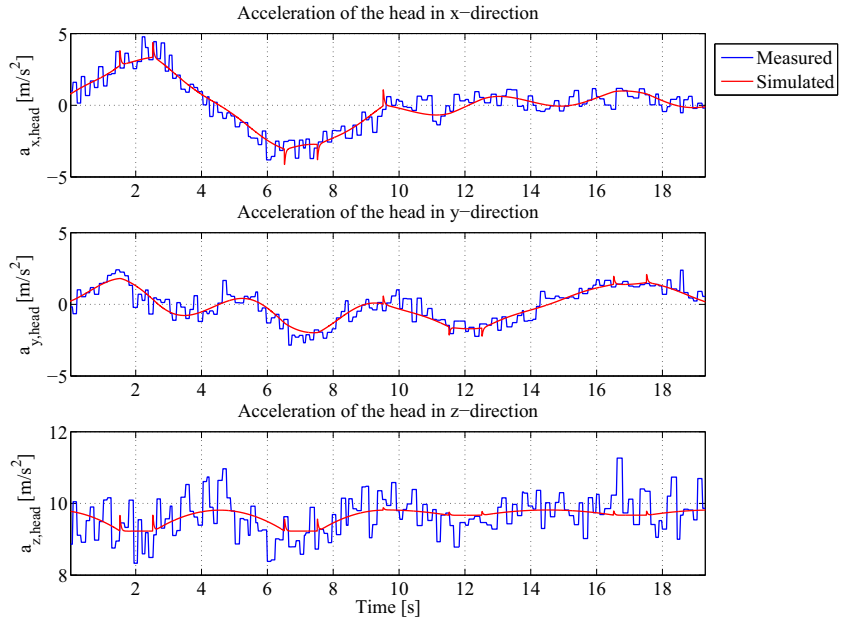


Figure F.22: *The measured and simulated acceleration of the head.*

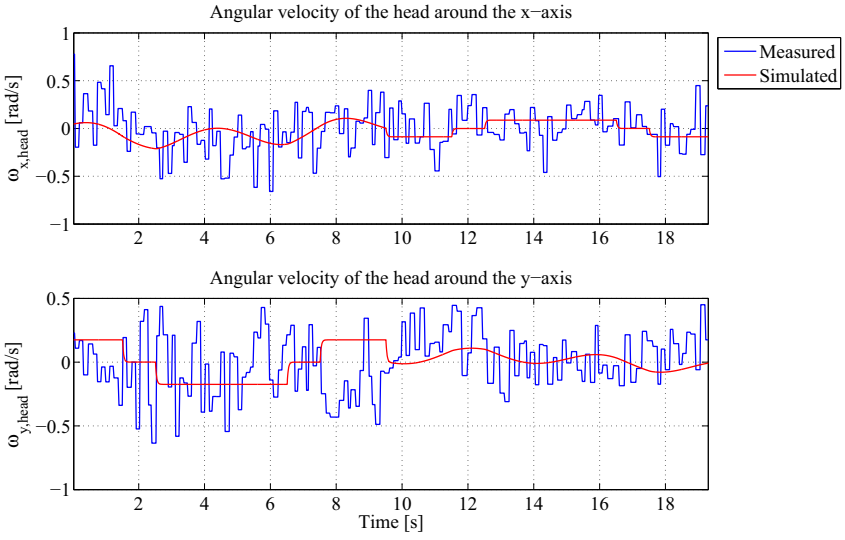


Figure F.23: *The measured and simulated angular velocity of the head.*

the model is correct. The problem with the low signal values is a trade-off, since it is desired to apply mush faster input to the robot, to really stimulate the dynamics of the system. But if this done, the robot would not be able to maintain stability without human interference.

F.5.4 Head Model Test Conclusion

Based on the test results, it is concluded that the head model is correct. This is entirely based on the output of the accelerometer, since it is not possible to conclude anything based on the measurements of the angular velocity. The low R^2-values of the angular velocity is however, assigned a bad input rather than a bad model. Based on the knowledge that $\boldsymbol{\alpha}_{\text{head}}$ and $\boldsymbol{\omega}_{\text{head}}$ are calculated from the the same rotation matrices, it is valued that the entire model is correct. The head model is now considered verified.

F.6 Verification of the Dynamical Model

In this section, the dynamical model is verified. The description of the derivation of the model can be found in Section 7.4 on page 79. The inputs and outputs of the dynamic model is seen from Figure F.24.

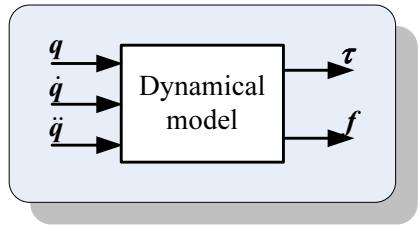


Figure F.24: *Input/output relation of the dynamical model*

As it can be seen,the input to the dynamical model is the generalized coordinate vector, $\boldsymbol{q}$, velocity vector, $\dot{\boldsymbol{q}}$, and acceleration vector, $\ddot{\boldsymbol{q}}$, while the outputs are the torques, $\boldsymbol{\tau}$, and forces, $\boldsymbol{f}$, working in each joint. These forces and torques can not be measured directly on the robot, and the dynamical model therefore has to be validated through other models. As the forces acting on the feet are measured through the strain gauges, these can be used to verify the dynamical properties of the system. Therefore the foot model is included in the verification of the dynamical model. As the input to the dynamical model is the generalized coordinates the kinematic model is included in the verification to transform the joint coordinates. The servo motor model is also included in the test of the dynamical model, since the calculated torque is used for feedback to the servo motors, and thereby having an effect on the dynamical properties.

F.6.1 Dynamical Model Test Setup

The test is divided into two sub-tests, as it is assessed that the dynamical movement of all limbs at one time can be difficult to see on the output of the strain gauges. Therefore the dynamical properties of the arms and of the legs are tested separately. When testing the legs, an input must be constructed in such a way, that the system maintain stability during the test. To obtain the best stability the robot must be in DSP. An input is then

constructed which shifts the weight from left to right foot. The test input is illustrated in Figure F.25. The input is applied to both the model and the real system. Output from the strain gauges is compared.

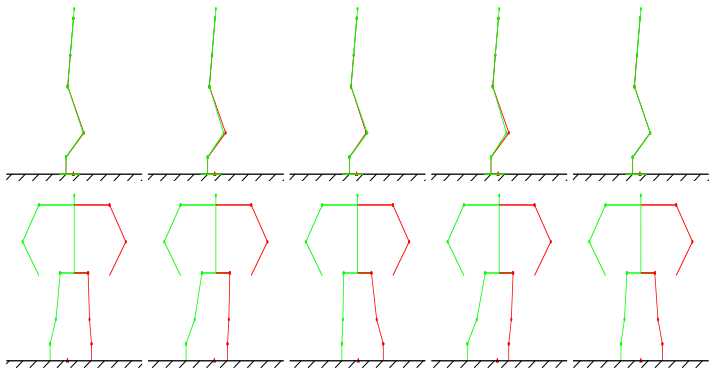


Figure F.25: *Animation of the movements used in the verification of the dynamical model of the legs. The five figures above show the movements in the sagittal plane, while the five figures below show the movements in the frontal plane. The span of the test is* 20 s.

Another input is created, which should verify the dynamical changes in the arms. This input is illustrated in Figure F.26. As for the leg test, the output of the strain gauges is compared for the dynamical test of the arms.

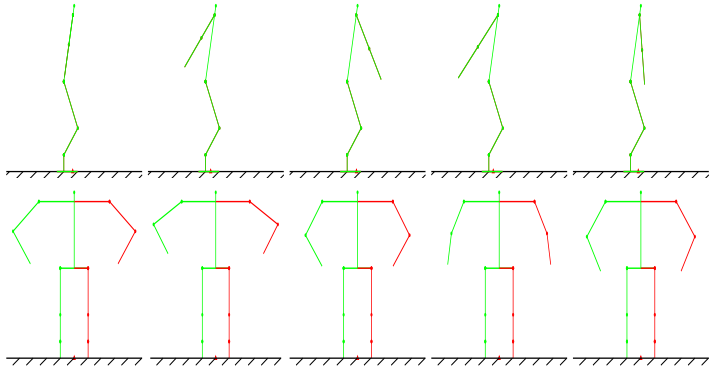


Figure F.26: *Animation of the movements used in the verification of the dynamical model of the arms. The five figures above show the movements in the sagittal plane, while the five figures below show the movements in the frontal plane. The span of the test is* 10 s.

Next the results of the test is given.

F.6.2 Dynamical Model Test Results

Figure F.27 and F.28 show the output of the strain gauges for the dynamical test of the legs.

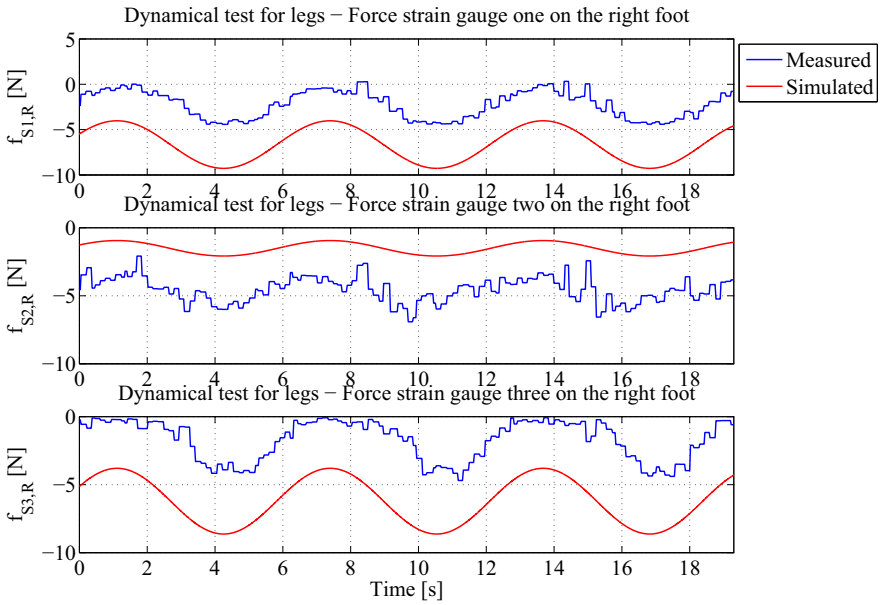


Figure F.27: *Output of the strain gauges on the right foot for the leg test of the dynamical model.*

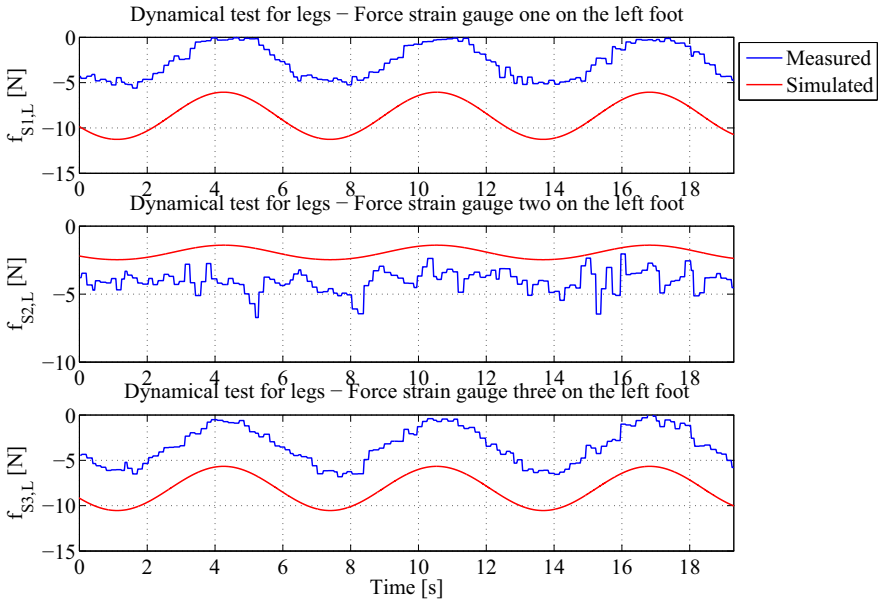


Figure F.28: *Output of the strain gauges on the left foot for the leg test of the dynamical model.*

In Figure F.29 and F.30 the output of the strain gauges is shown for the dynamical test of the arms.

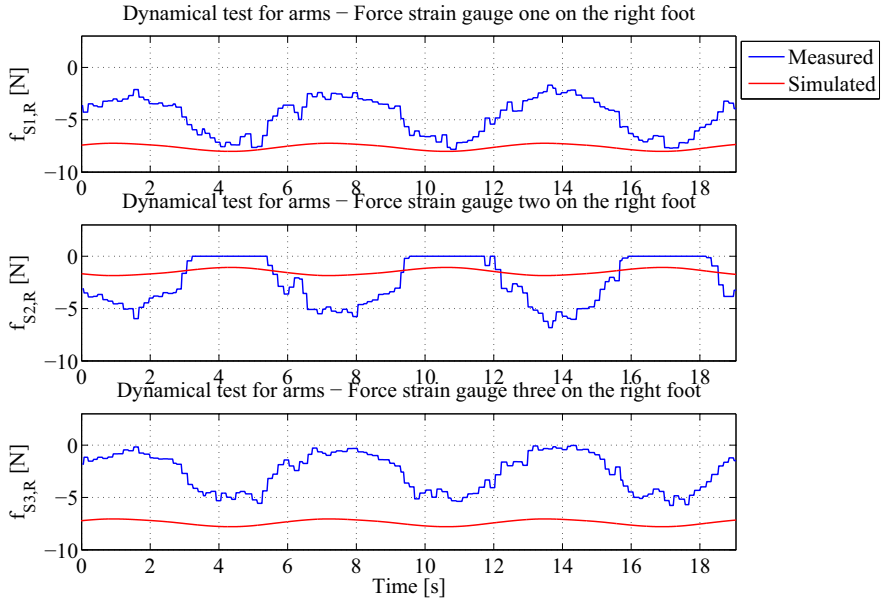


Figure F.29: *Output of the strain gauges on the right foot for the arm test of the dynamical model.*

F.6.3 Dynamical Model Test Discussion

As can be seen in Figure F.29, F.30, F.27 and F.28 the results from the two tests of the dynamical model are not very good. There are some trends which match between the model and the measurements, but generally there is not very good consistency. There are several reasons for this, the first thing to notice is that on some of the measurements there are higher amplitudes than on the output from the model. This larger amplitude is caused by the bending of the spring steel, on which the strain gauges are mounted. As the arms move forward the weight of the robot makes the spring steel bend further, and thereby amplifying the output of the strain gauges. This property was also seen in the test of the foot model described in Section F.3.

Furthermore large ripples are presented throughout the measured signals. The reason for this, is that when moving the robot, the servos get heavier loaded and therefore have a harder time keeping the right position. This is seen as shaking of the robot when moving and have a large impact on the output of the strain gauges, as the shaking is amplified by the spring effect of the strain gauge plates. The levels, or the means values, of the measurements and the model does not fit either, this partly caused by the backlash, which makes the robot turn forward, and thereby completely changing the pressure distribution under the feet. An other reason is the small feet, because the pressure distribution is highly dependent on the placement of the feet. When haven

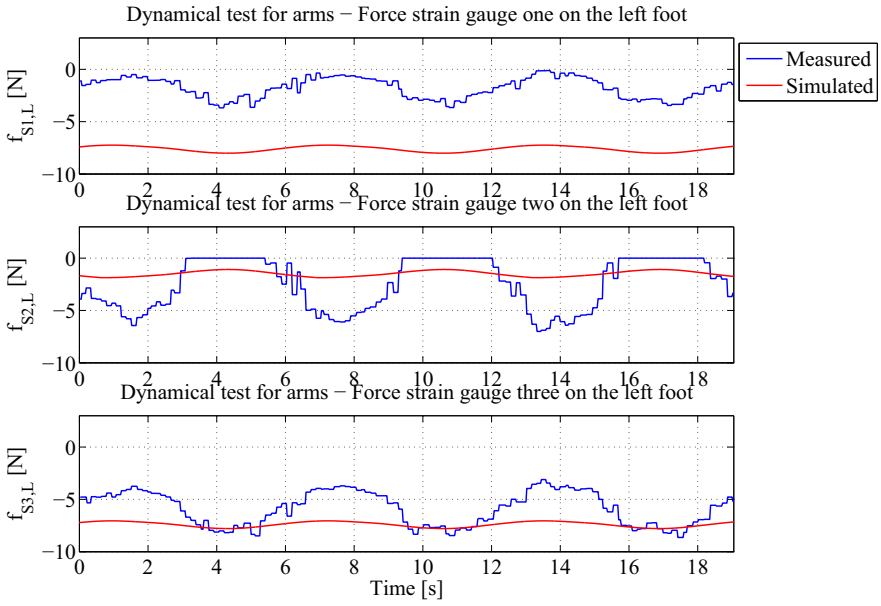


Figure F.30: *Output of the strain gauges of the left foot for the arm test of the dynamical model.*

these dissimilarities in the levels, it makes no sense to calculate the error percentage.

Even though the levels and the fluctuations does not match, it is clearly seen that that there is a relationship between the output of the model and the measurements, they have the same characteristics.

F.6.4 Dynamic Model Test Conclusion

The results of the dynamic model test are varying in success. However having these unmodeled bendable plates under the feet, it is impossible to obtain better results. Based on the fact that the dynamical model is verified through three other models with their own limitations, and taking all the uncertainties into consideration, the dynamical model is considered good for the purpose and scope of this project. But if a exact description of the system is desired, it is clear from this test, that it is necessary to include the bendable plates under the feet in the model.

F.7 Verification of the Complete Model

In this section the complete model is verified. This is done by comparing the output of the complete model with the sensor output of the real system, given the same input. The model contains all sub-models and the input and outputs are as shown i Figure F.31.

Input to the system and model for this test is the positions joint angle vector, $\boldsymbol{\theta}$, and the output are the positions of the servos in the legs, $\boldsymbol{\theta}_{\text{leg}}$, the acceleration and rotation of the head, $\boldsymbol{a}_{\text{head}}$ and $\boldsymbol{\omega}_{\text{head}}$, and the force acting on the strain gauges, $\boldsymbol{f}_{\text{feet}}$. Next the test setup is described.

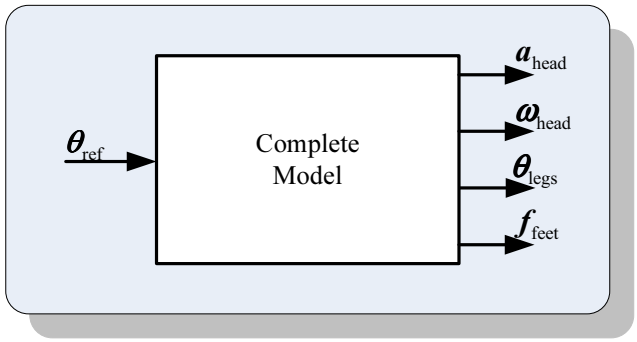


Figure F.31: *Input and outputs of the complete model.*

F.7.1 Complete Model Test Setup

To verify the complete model an input must be created, which ensures that a noticeable fluctuation is seen on all measured output of both the model and the real system. At the same time, the input must be created in such way that the real robot is stable without human interference. This last requirement limits the possible input trajectories to only include those in DSP. Movements in forward direction are limited, as this would cause the robot to fall. Therefore an input is created which moves the torso in the sagittal plane. Further more the arms are moved in all directions while the head is turned from side to side. Thereby are all servos actuated. The test is conducted for 30 s, but the first 5 s is discarded as settle time. The movements of the robot are illustrated in Figure F.32.

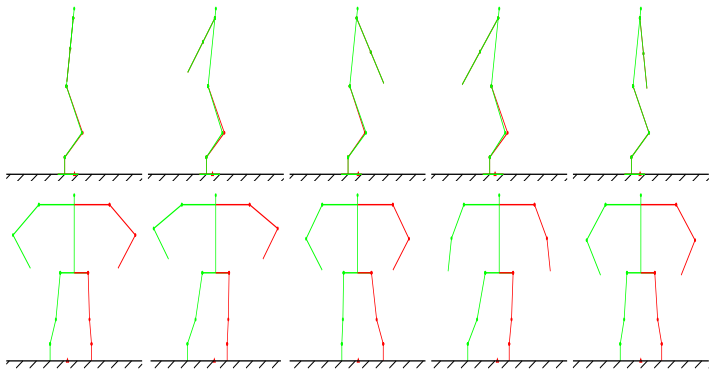


Figure F.32: *Animation of the movements of the robot used in the verification of the complete model. The five figures above show the movements in the sagittal plane, while the five figures below show the movements in the frontal plane. The span of the test is* 20 s*, but here is shown the movements of the first* 10 s *as these are repeated in the last* 10 s.

Next the results of the complete test are shown.

F.7.2 Complete Model Test Results

There are four outputs from the complete test. These are the servo motor positions, the output of the strain gauges, the acceleration of the head and the angular velocity. The

$R^2_{\theta_1}$	$R^2_{\theta_2}$	$R^2_{\theta_3}$	$R^2_{\theta_4}$	$R^2_{\theta_5}$	$R^2_{\theta_6}$
–	0.266	0.445	0.591	–	0.663

$R^2_{\theta_7}$	$R^2_{\theta_8}$	$R^2_{\theta_9}$	$R^2_{\theta_{10}}$	$R^2_{\theta_{11}}$	$R^2_{\theta_{12}}$
0.477	–	–	–	0.299	–

Table F.6: *Calculated R^2-value of the servo motor output in the complete model test. – means that the calculated error is unuseable, due to an offset error in the measurements.*

servo motor positions are shown in Figure F.33 and F.34.

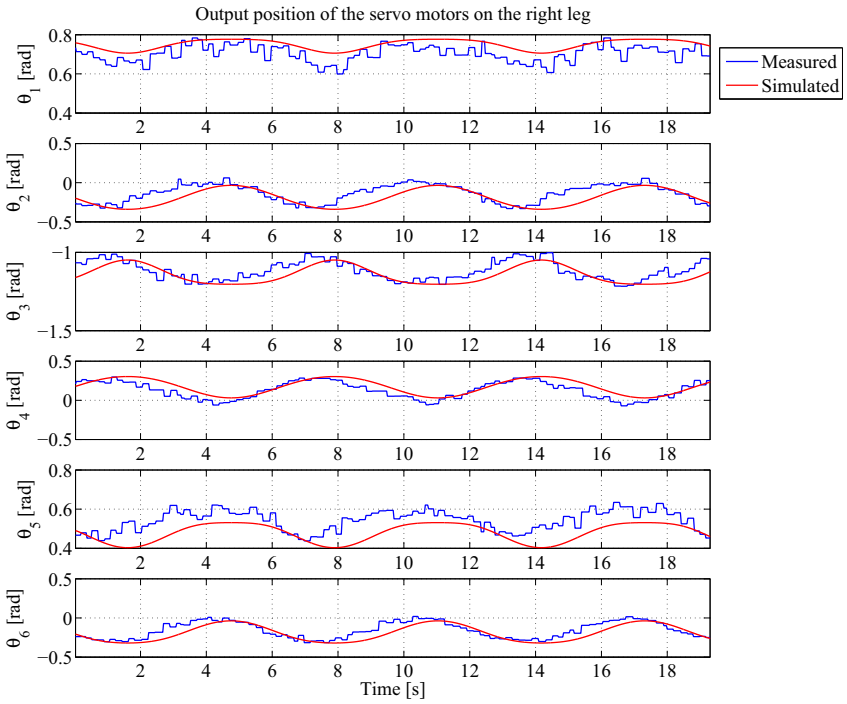


Figure F.33: *Results from the complete model verification, showing the output of the servo motors on the right leg.*

The R^2-value of the servo motor positions are calculated as shown in Equation (F.1), can be seen in Table F.6.

The output of the strain gauges on the right foot is seen in Figure F.35, while the output of the strain gauges on the left foot is seen in Figure F.36.

The output of the IMU is seen from Figure F.37 for the angular velocity and in Figure F.38 for the acceleration.

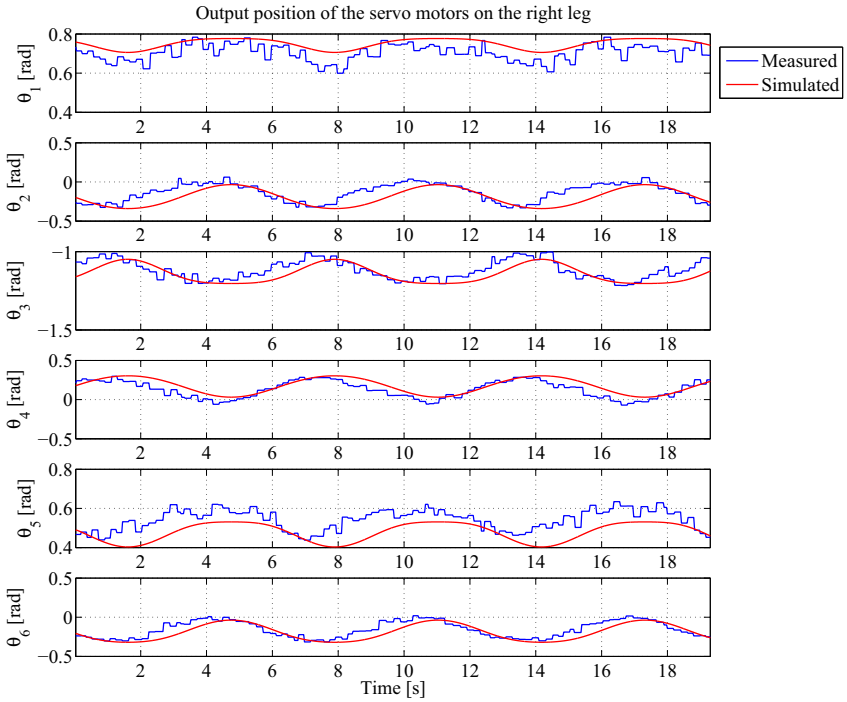


Figure F.34: *Results from the complete model verification, showing the output of the servo motors on the left leg.*

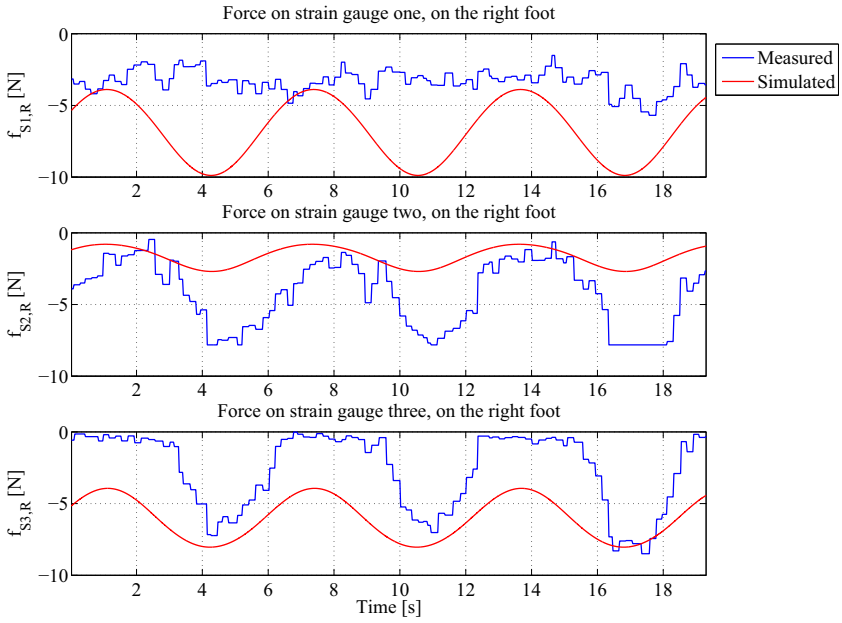


Figure F.35: *Output of the strain gauges on the right foot for the complete model test.*

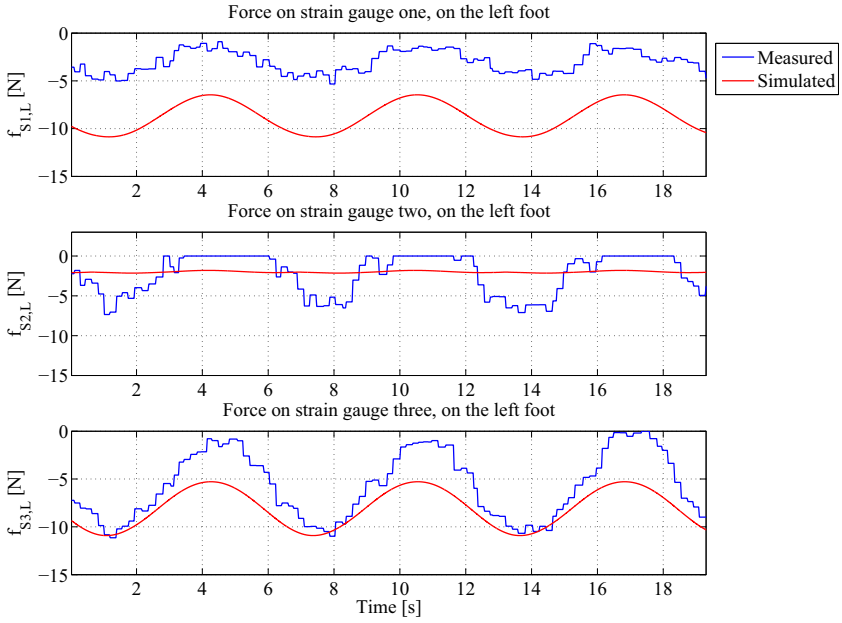


Figure F.36: *Output of the strain gauges on the left foot for the complete model test.*

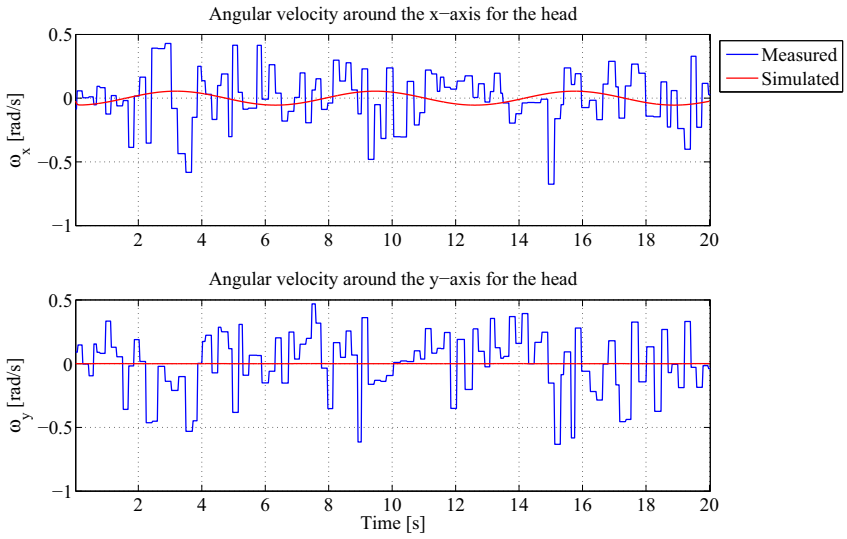


Figure F.37: *Angular velocity output of the IMU in the complete model test.*

F.7.3 Complete Model Test Discussion

It is seen from Figure F.33 and F.34 that the output of the model of the servo motors estimates the real system good, though a little delayed, but the fluctuations and amplitudes in the signals are seen to match.

Looking at the output from the strain gauges in Figure F.35 and F.36, it is seen that the model output and the measurements does not well. This was however expected, based on the results from the verification on the complete model, where it was found that if a precise output of the strain gauges is wanted, it is necessary to include the strain plates in the model. It is however seen, that the strain gauges estimates the trends of the signal. Especially $f_{S2,R}$, $f_{S1,L}$ and $f_{S3,L}$ track the signal very well, though with an offset error. This is caused by the small feet, since the measurements are very dependent on the placement of these. Due to the offset errors it makes no sense to calculate the error percentages.

Looking at Figure F.37 it is seen that the model and the measurements are equal in level, but with the given input the fluctuations in the signal very small compared to the noise. If this output should be evaluated, it is necessary to give a different input, where the angular velocity in the frontal and sagittal plane is much larger. This is however not possible, since the robot is not able to maintain balance if so.

In Figure F.38 the acceleration can be seen, and for $a_{y,head}$, it can be seen that the model tracks the signal very well, but a significant level of noise is however presented on the signal. For $a_{x,head}$ and $a_{z,head}$ the fluctuations in the signal are to small to be seen, but it

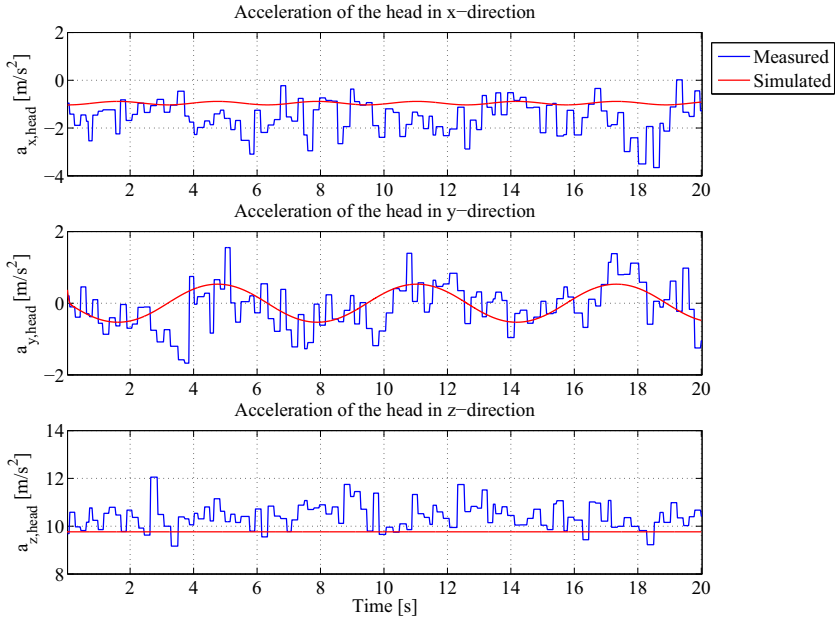


Figure F.38: *Acceleration output of the IMU in the complete model test.*

can however be seen that the level match.

It should be noted that the measured data is raw, unfiltered information. If the signals were filtered it is possible to get a better result in many of the cases. Only the servo motor positions are filtered through the filter implemented in hardware.

F.7.4 Complete Model Test Conclusion

It is not completely clear from the results of this test if the complete model is correct. It is however seen that the model gives the same trends as the system. But it is also clear, that it is necessary to include the bending of the strain plates in the model, if a more precise description is desired. This will however not be done, since a better solution would be to design other feet for the robot, and thereby remove the limit cycles of the system.

Furthermore it is necessary to include the backlash in the model if a precise model is wanted. It is however valued that the model is precise enough for the scope of this project.

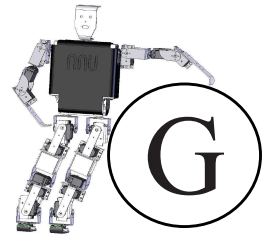
Based on the fact that all sub-models have been verified individually, it is concluded that the developed model is correct, and no further modeling will be done.

F.8 Partial Conclusion to Modeling Verification

In this section was the model verified. All the sub-models was verified individually and the complete model was then verified using the sub-models. The tests showed that the

model overall gave a good estimation of the real system. It was found that the model could be further enhanced by including un-modeled effects such as the spring effect of the strain plates in the foot, as well as backlash in the system.

NUMERICAL SOLUTION TO INVERSE KINEMATICS



G

G.1 Conceptual Knowledge

As stated in Chapter 8 on page 101 the solution to the inverse kinematic is given as:

$$\boldsymbol{\theta} = \boldsymbol{f}^{-1}(\boldsymbol{x}) \tag{G.1}$$

Due to the complexity and nonlinearity of $\boldsymbol{f}$, a closed form solution can often be difficult to establish. In [Goldenberg et al., 1985], [Sciavicco and Siliano, 1988] and [Deo and Walker, 1997] the inverse kinematic model is found from the Jacobian matrix, expressing the following relationship:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\dot{\boldsymbol{\theta}}) \tag{G.2}$$

where $\boldsymbol{J}(\boldsymbol{\theta}) \equiv \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}$. Thus the Jacobian describes the relationship between changes in joint space and the corresponding changes in Cartesian space.

If a robot arm has redundant DoF, the inverse kinematic problem can be divided into sub-problems, where an iterative procedure can be used to determine some of the joint variables, and the rest can be determined to minimize some performance function.

The aim of the inverse kinematic model is to transform a desired position, $\boldsymbol{p}_{\mathrm{T}}$, and orientation, $\boldsymbol{R}_{\mathrm{T}}$, of the torso, into joint coordinates. Furthermore to transform a desired position, $\boldsymbol{p}_{\mathrm{F}}$, and orientation, $\boldsymbol{R}_{\mathrm{F}}$, of the swing foot relative to the torso into joint variables. These position and rotation vectors can be seen from Figure G.1. In the three-dimensional space a robot arm must have at least 6 DoF in order to position the end-effector in an arbitrary position and orientation [Goldenberg et al., 1985] [Craig, 1989]. Thus the mechanical structure of the leg leaves no redundant joints for optimization.

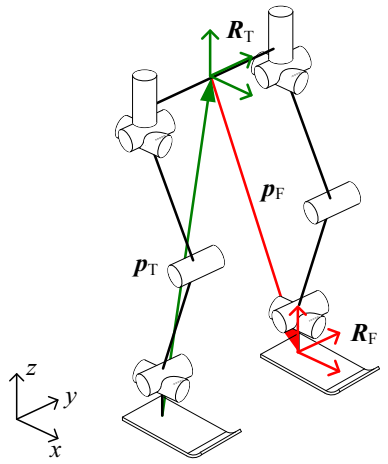


Figure G.1: *The position and rotation vectors used in the numerical solution of the inverse kinematics.*

The desired position of the torso is given as $\boldsymbol{p}_{\mathrm{T}} = [x_{\mathrm{T}}\ y_{\mathrm{T}}\ z_{\mathrm{T}}]^T$, and the orientation as $\boldsymbol{R}_{\mathrm{T}} = [\,\boldsymbol{n}_{\mathrm{T}}\ \boldsymbol{o}_{\mathrm{T}}\ \boldsymbol{a}_{\mathrm{T}}\,]$. Where $\boldsymbol{n}_{\mathrm{T}}\ \boldsymbol{o}_{\mathrm{T}}\ \boldsymbol{a}_{\mathrm{T}}$ are the three column vectors of the rotation matrix, ${}^{6}_{0}\boldsymbol{R}$, explained in Section 7.3.2. ${}^{6}_{0}\boldsymbol{R}$ is the rotation matrix from frame $\{0\}$ to frame $\{6\}$, thus it describes the orientation of the torso relative to the right foot.

Now a residual vector, $\boldsymbol{r}(\boldsymbol{\theta}) \in \mathbb{R}^6$, can be defined:

$$\boldsymbol{r} = \begin{bmatrix} r_{\mathrm{x}} \\ r_{\mathrm{y}} \\ r_{\mathrm{z}} \\ r_{\alpha} \\ r_{\beta} \\ r_{\gamma} \end{bmatrix} = \begin{bmatrix} \boldsymbol{n}_{\mathrm{a}} \cdot (p_{\mathrm{T}} - p_{\mathrm{a}}) \\ \boldsymbol{o}_{\mathrm{a}} \cdot (p_{\mathrm{T}} - p_{\mathrm{a}}) \\ \boldsymbol{a}_{\mathrm{a}} \cdot (p_{\mathrm{T}} - p_{\mathrm{a}}) \\ \frac{1}{2}(\boldsymbol{a}_{\mathrm{a}} \cdot \boldsymbol{o}_{\mathrm{T}} - \boldsymbol{a}_{\mathrm{T}} \cdot \boldsymbol{o}_{\mathrm{a}}) \\ \frac{1}{2}(\boldsymbol{n}_{\mathrm{a}} \cdot \boldsymbol{a}_{\mathrm{T}} - \boldsymbol{n}_{\mathrm{T}} \cdot \boldsymbol{a}_{\mathrm{a}}) \\ \frac{1}{2}(\boldsymbol{o}_{\mathrm{a}} \cdot \boldsymbol{n}_{\mathrm{T}} - \boldsymbol{o}_{\mathrm{T}} \cdot \boldsymbol{n}_{\mathrm{a}}) \end{bmatrix} \tag{G.3}$$

where $\boldsymbol{R}_{\mathrm{a}} = [\boldsymbol{n}_{\mathrm{a}}\ \boldsymbol{o}_{\mathrm{a}}\ \boldsymbol{a}_{\mathrm{a}}\,]$ is the actual orientation of the torso, and $\boldsymbol{p}_{\mathrm{a}} = [x_{\mathrm{a}}\ y_{\mathrm{a}}\ z_{\mathrm{a}}]^{\mathrm{T}}$ is the actual position of the torso. The squared error can now be calculated as:

$$G = \boldsymbol{r}^T \boldsymbol{r} \tag{G.4}$$

The Jacobian that relates the changes in joint coordinates to changes in the residual vector, is defined as:

$$\boldsymbol{J}_{\mathrm{res}} \equiv \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial r_{\mathrm{x}}}{\partial \theta_1} & \cdots & \frac{\partial r_{\mathrm{x}}}{\partial \theta_6} \\ \frac{\partial r_{\mathrm{y}}}{\partial \theta_1} & \cdots & \frac{\partial r_{\mathrm{y}}}{\partial \theta_6} \\ \frac{\partial r_{\mathrm{z}}}{\partial \theta_1} & \cdots & \frac{\partial r_{\mathrm{z}}}{\partial \theta_6} \\ \frac{\partial r_{\alpha}}{\partial \theta_1} & \cdots & \frac{\partial r_{\alpha}}{\partial \theta_6} \\ \frac{\partial r_{\beta}}{\partial \theta_1} & \cdots & \frac{\partial r_{\beta}}{\partial \theta_6} \\ \frac{\partial r_{\gamma}}{\partial \theta_1} & \cdots & \frac{\partial r_{\gamma}}{\partial \theta_6} \end{bmatrix} \tag{G.5}$$

Applying the chain rule to Equation (G.5) gives following relationship:

$$\dot{\boldsymbol{r}} = \boldsymbol{J}_{\mathrm{res}} \dot{\boldsymbol{\theta}} \tag{G.6}$$

An expression for the changes in joint variables for corresponding errors can now be formulated as:

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}_{\text{res}}^{+}\dot{\boldsymbol{r}} \tag{G.7}$$

where $\boldsymbol{J}_{\text{res}}^{+}$ is the pseudo-inverse. The following section describes the developed iterative procedure that, by means of the pseudoinverse Jacobian matrix, transforms a desired position and orientation of the torso into joint variables.

G.2 Iterative Procedure

The main steps of the numerical iterative procedure are produced with inspiration from [Goldenberg et al., 1985], and are as follows (k is the iteration):

1. Initialization:
$$k = 1 \ , \ \hat{\boldsymbol{\theta}}_k = \boldsymbol{\theta}$$

2. Determine the error:
   (a) Find the residual vector, $\boldsymbol{r}(\hat{\boldsymbol{\theta}}_k)$, from Equation (G.3).
   (b) Find the squared error, G_k, from Equation (G.4).

3. Calculate the next step:
   (a) Find the Jacobian matrix, $\boldsymbol{J}_{\text{res}}(\hat{\boldsymbol{\theta}}_k)$, from Equation (G.5)
   (b) Find the pseudoinverse Jacobian matrix, $\boldsymbol{J}_{\text{res}}^{+}(\hat{\boldsymbol{\theta}}_k)$
   (c) Calculate the changes in joint variables corresponding to the error:
$$\boldsymbol{\delta}_k = \boldsymbol{J}_{\text{res}}^{+}(\hat{\boldsymbol{\theta}}_k)\boldsymbol{r}(\hat{\boldsymbol{\theta}}_k)$$

4. Step size control, $\lambda = 1$:
   (a) Calculate $\boldsymbol{r}(\hat{\boldsymbol{\theta}}_k + \lambda\boldsymbol{\delta}_k)$, from Equation (G.3)
   (b) Calculate G_{k+1}, from Equation (G.4).
   (c) Check if the error is reduced $G_{k+1} < G_k$. If not, reduce step size by:
$$\lambda = \frac{1}{2}\lambda$$
   and return to step 4a.

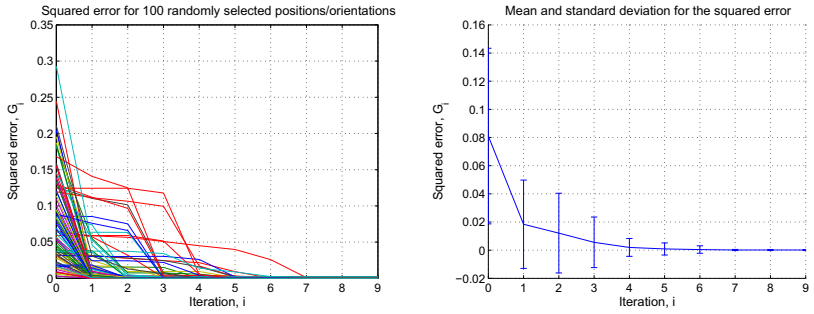
5. Update joint variables: $\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k + \lambda\boldsymbol{\delta}_k$

6. Check if the squared error is less than a given threshold value, ϵ:
   - if $G_{k+1} < \epsilon$,
     then $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}_{k+1}$ and terminate the procedure
   - else,
     $k = k + 1$ and return to step 3

The numerical procedure described above has been implemented in MATLAB as a function that takes the desired position, $\boldsymbol{p}_{\text{T}}$, and orientation, $\boldsymbol{R}_{\text{T}}$, of the torso as an input, and returns the corresponding joint variables, $\boldsymbol{\theta}$. Above procedure can also be used for the left leg.

G.3 Result

The iterative procedure described in previous section has been implemented and tested. In order to test the procedure, a test-function has been developed. This function randomly selects a position and and orientation of the torso. In practice this position and orientation are found by randomly selecting $[\theta_1, \theta_2, ..., \theta_6]$, and then the kinematic model is used to find the corresponding position and orientation of the torso. Thereby it is ensured that the solution is reachable. The inverse kinematic procedure is now called with 100 different input, and the result for the 100 simulations can be seen in Figure G.2(a). In Figure G.2(b) the mean and standard deviation are shown.



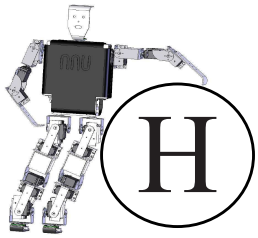
(a) For all 100 random positions.

(b) Mean and standard deviation

Figure G.2: *The squared error after each iteration for 100 randomly chosen positions/orientations.*

As seen in Figure G.2(a) the error converges to zero, and Figure G.2(b) shows that after only one iteration the mean error is reduce by a factor 5. From this it is concluded that the developed procedure serves its purpose. The downside is however, that the simulation time of this test was $289\,\mathrm{s}$, on a $2.8\,\mathrm{GHz}$ P4 computer. This means that each calculation of the angles in average took 2.9_s. It should however be noted, that this is only for one leg, which means that the calculation time for all links will be approximately $10\,\mathrm{s}$. Looking at Figure G.2(a), one could argue that it is enough to run the simulation for one iteration, if the starting error was small. However in the application, the inverse kinematics should be called 50 times per second, and perhaps only seize a maximum of a tenth of the resources on the $200\,\mathrm{MHz}$ CPU mounted on the robot. This means that the runtime should be improved with at least a factor $10,000$. This seems as an impossible task, and another approach must be taken to solve the inverse kinematic problem.

VERIFICATION OF THE INVERSE KINEMATIC MODEL



H

In this appendix the inverse kinematic model is verified. The inverse kinematic model gives a coordinate transformation from generalized coordinates of the end of each limb and of the torso, to joint space. The input and output of the inverse kinematic model can be seen from Figure H.1.

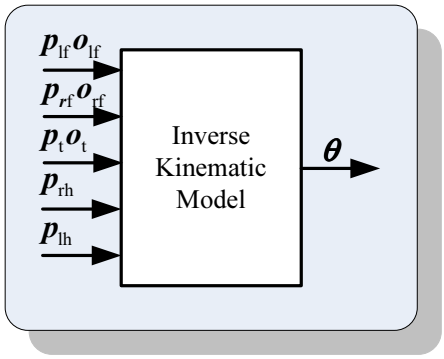


Figure H.1: *Inputs and output relations of the inverse kinematic model. Inputs are positions and orientation, given in Cartesian space, of the limbs and torso. Output is the joint space coordinates.* $\mathbf{p}_{lf}$ *is the position of the left foot.* $\mathbf{p}_{rf}$ *is the position of the right foot.* $\mathbf{p}_t$ *is the position of the torso.* $\mathbf{p}_{rh}$ *is the position of the right hand and* $\mathbf{p}_{lh}$ *is the position of the left hand.*

As the joint coordinates are only measurable for the leg joints, it is not sufficient to test the inverse kinematic model using these. Instead the inverse kinematic model is verified through the kinematic model. As the kinematic model has been verified, this can be used to verify other models. The verification of the kinematic model can be seen in Appendix F in Section F.4. By connecting the inverse kinematic model with the kinematic model a transformation of coordinates can be made from Cartesian

space to joint space and then back to Cartesian space. By comparing the positions and orientations before and after the transformation to joint coordinates it can be seen whether the inverse kinematic model is correct. This connection can be seen from Figure H.2. The kinematic model gives the generalized coordinates as output, in which the position of all links can be found, but it can also be modified to give the orientation of the limbs. This information is inside the model, and is therefore accessible.

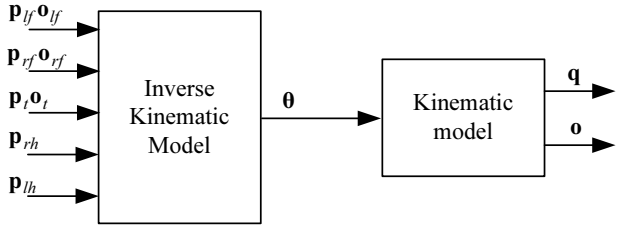


Figure H.2: *The positions and orientations of the limbs are transformed to joint space coordinates and then back to generalized coordinates.*

H.1 Inverse Kinematic Model Test Setup

For this test no physical test setup is used. Instead the test is conducted through the kinematic model, as shown in Figure H.2. An input is constructed which lies inside the reachable space of the kinematics. This input is then transformed to joint space through the inverse kinematic model. Joint coordinates are transformed via the kinematic model, which returns the generalized coordinates of all joints and links. Now the input is compared to the output. An animation of the input used to test the inverse kinematic model is shown in Figure H.3.

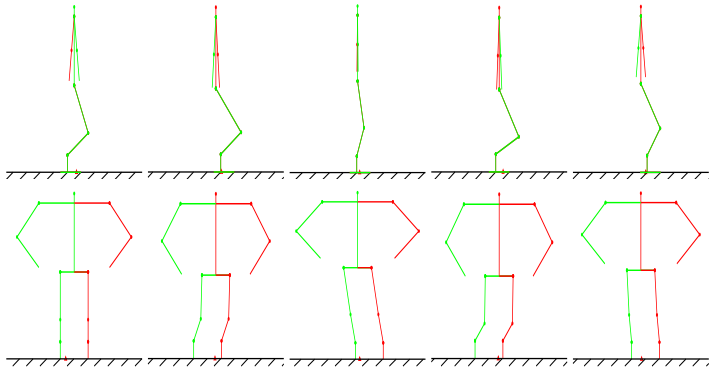


Figure H.3: *Animation of the movements used in the verification of the inverse kinematic model. The five figures above show the movements in the sagittal plane, while the five figures below show the movements in the frontal plane. The span of the test is* 10 s.

Next the results of the inverse kinematic test are given.

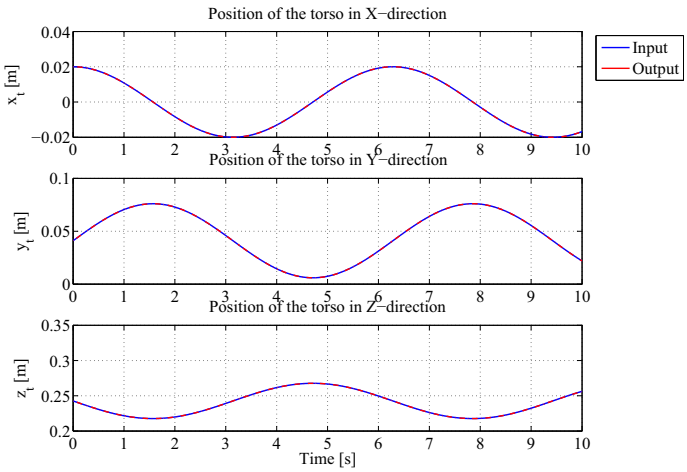


Figure H.4: *The position of the torso before the inverse kinematic model and after the kinematic model.*

	R^2_x	R^2_y	R^2_z	$R^2_{o_x}$	$R^2_{o_y}$	$R^2_{o_z}$
Torso	1	1	1	1	1	1
Right Hand	1	1	1	-	-	-
Left Hand	1	1	1	-	-	-

Table H.1: *Calculated R^2-values of the inverse kinematic test. Orientations of the hands are not possible to set.*

H.2 Inverse Kinematic Model Test Results

The position of the torso before the inverse kinematic model and after the kinematic model is seen from Figure H.4.

The position of the right hand before the inverse kinematic model and after the kinematic model is seen from Figure H.5.

The position of the left hand before the inverse kinematic model and after the kinematic model is seen from Figure H.6.

The orientation of the torso before the inverse kinematic model and after the kinematic model is seen from Figure H.7.

The calculated R^2-values of the inverse kinematic test results are seen from Table H.1.

H.3 Inverse Kinematic Model Test Discussion

As the inverse kinematic model is tested through the kinematic model, there should be a complete match as no noise is introduced. The calculated R^2-values were therefore expected to be 1. The movement of the feet are zero and are therefore not plotted.

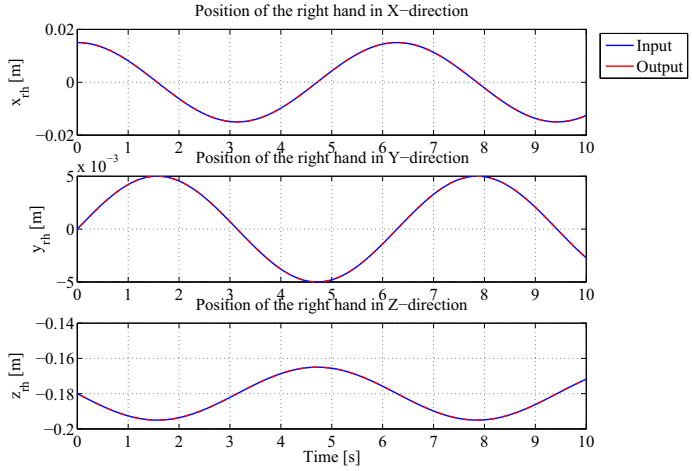


Figure H.5: *The position of the right hand before the inverse kinematic model and after the kinematic model.*

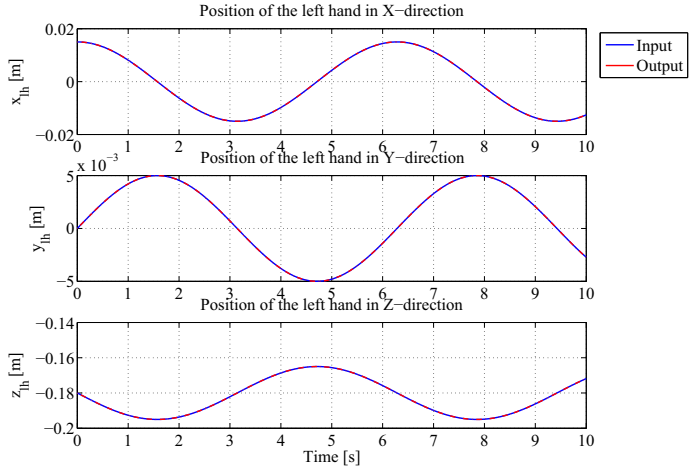


Figure H.6: *The position of the left hand before the inverse kinematic model and after the kinematic model.*

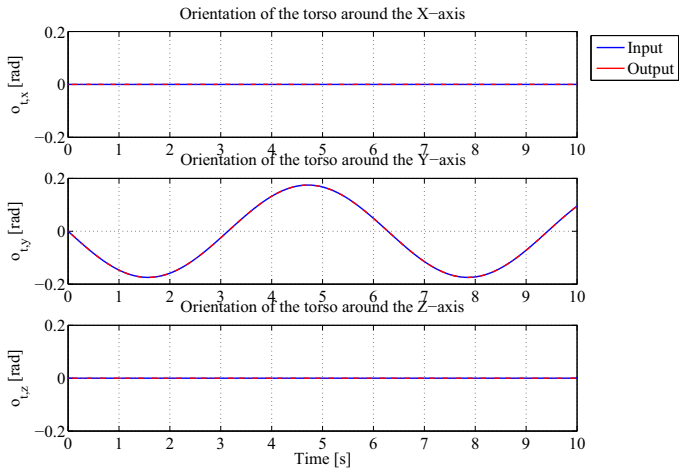
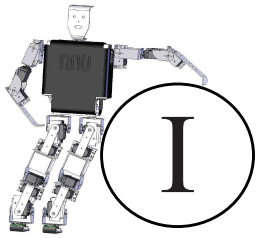


Figure H.7: *The orientation of the torso before the inverse kinematic model and after the kinematic model.*

H.4 Inverse Kinematic Model Test Conclusion

It can be seen from Figure H.4, H.5 and H.6 that there is a complete match in the positioning before the inverse kinematic model and after the kinematic model. This is also the case for the orientation for the torso, as seen in Figure H.7. This is further strengthen by the R^2-values shown in Table H.1. On basis of this, it is concluded that the inverse kinematic model is able to transform desired positions and orientations into joint space coordinates. The inverse kinematic model is therefore considered verified.

TRAJECTORIES



This appendix includes a description of the derivation of the trajectories for the start-up and stop phase, which can be found in Section I.2. In Section I.3 the polynomials for the trajectories found in Chapter 9 are presented. It was found that the biped robot was not able to walk with these trajectories due to the actuators, and it was decided that the walking speed should be decreased. Therefore it was necessary to derive new trajectories, and the derivation of these trajectories are described in Section I.4.

I.1 Establishing the Trajectories for the Start-up Phase

In order to initiate the walk it is necessary to have a start-up phase. The trajectories of the start-up phase should be designed such that the they change the position, velocity and acceleration of the biped, from its initial static position to match the movement at time t_1 for $k = 0$. This is the movement/position of the steady-state walk, which is illustrated in Figure 9.4(a) on page 117 and Figure 9.5(a) on page 120. From this a number of initial conditions can be formulated for the start-up trajectories:

$$\begin{aligned} \mathbf{v}_{\text{a}}(t_{\text{init}}) = \quad \dot{\mathbf{v}}_{\text{a}}(t_{\text{init}}) \quad &= [0\ 0\ 0]^{\text{T}} \\ \mathbf{v}_{\text{t}}(t_{\text{init}}) = \quad \dot{\mathbf{v}}_{\text{t}}(t_{\text{init}}) \quad &= [0\ 0\ 0]^{\text{T}} \end{aligned} \tag{I.1}$$

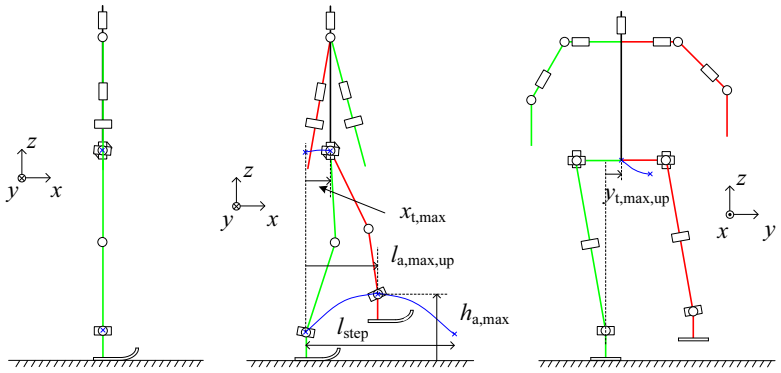
where $\mathbf{v}_{\text{a}}(t_{\text{init}})$ and $\mathbf{v}_{\text{t}}(t_{\text{init}})$ are velocity vectors containing the velocity in all three directions of the ankle and the torso at time t_{init} when the start-up is initiated. Furthermore the biped is placed in its initial position as illustrated in Figure I.1(a).

I.1.1 Start-up Phase for Ankle

For the ankle trajectory in the x-direction the following constraints can be specified:

$$x_{\text{a}}(t) = \begin{cases} 0 & , t = t_{\text{init}} \\ 0 & , t = t_{\text{a}} \\ l_{\text{a,max,up}} & , t = t_{\text{up,max}} \\ l_{\text{step}} & , t = 0 \end{cases} \tag{I.2}$$

where t_{a} is the time at which the foot leaves the ground, $t_{\text{up,max}}$ is the time at which the ankle reaches its maximum as shown in Figure I.1(b) and $x_{\text{a}}(t) = l_{\text{step}}$ at $t = 0$,



(a) Inital: $t = t_{\text{init}}$ (b) SSP: $t = t_{\text{up,max}}$ (c) SSP: $t = t_{\text{y,max,up}}$

Figure I.1: *Illustration of the start-up phase. I.1(a) initial position, in I.1(b) the ankle reaches its maximum point and the torso reaches $x_{t,max}$. I.1(c) shows biped robot at the instant where it reaches its maximum deviation to the right during the start-up phase.*

is the same position as the leading foot of Figure 9.4(a) where the steady-state walk is initiated. For the z-direction following must be fulfilled:

$$z_{\text{a}}(t) = \begin{cases} a_1 & , t = t_{\text{init}} \\ a_1 & , t = t_{\text{a}} \\ h_{\text{a,max}} & , t = t_{\text{up,max}} \\ a_1 & , t = 0 \end{cases} \tag{I.3}$$

Note that the foot does not leave the ground until after time t_{a}, the reason for that is that the CoM has to be moved over the supporting foot prior to initiating a SSP. As for the steady state walk, there is no other requirements to the movement of the foot in the y-direction, than:

$$\dot{y}_{\text{a}}(t) = 0 \tag{I.4}$$

I.1.2 Start-Up Phase for Torso

The trajectories for the torso is designed such that the stability index is minimized. The procedure is all most identical with the one used to obtain the trajectories for the steady-state walk. A number of constraints is formulated to ensure that the torso passes through the wanted positions. For the x-direction the constraints are:

$$x_{\text{t}}(t) = \begin{cases} 0 & , t = t_{\text{init}} \\ x_{\text{t,max}} & , t = t_{\text{up,max}} \\ x_{\text{t}}(t) & , t = 0 \end{cases} \tag{I.5}$$

where $x_{\text{t,max}}$ is the parameter that is varied in order to minimize the stability index and $x_{\text{t}}(0)$ is the position of the torso at the moment where the steady-state walk is initiated, this value can be found from the trajectories of the torso presented in Section 9.4 on page 125, to be $x_{\text{t}}(0) = x_{\text{t,e}} - l_{\text{step}}$. $x_{\text{t,max}}$ is varied in the range of:

$$0 \leq x_{\text{t,max}} \leq 0.6 x_{\text{t}}(0) \tag{I.6}$$

The trajectories in the y-direction should satisfy the following constraints:

$$y_{\text{t}}(t) = \begin{cases} y_{\text{t,mid}} & , t = t_{\text{init}} \\ y_{\text{t,max,up}} & , t = t_{\text{y,max,up}} \\ y_{\text{t,mid}} & , t = 0 \end{cases} \tag{I.7}$$

where $y_{\text{t,mid}}$ is the middle position of the torso as illustrated in Figure 9.5(a), $y_{\text{t,max,up}}$ is the maximum deviation to the right of the torso during the start-up phase, as illustrated in Figure I.1(c) which happens at time $t_{\text{y,max,up}}$. For the y-direction it is chosen to vary on two parameters, namely the value of $y_{\text{t,max,up}}$ and the time at which this point is reached, $t_{\text{y,max,up}}$. The first parameter is varied within the ranges specified i Equation (9.18) on page 121, the second parameter will be varied within:

$$0.3t_{\text{init}} \leq t_{\text{y,max,up}} \leq 0.7t_{\text{init}} \tag{I.8}$$

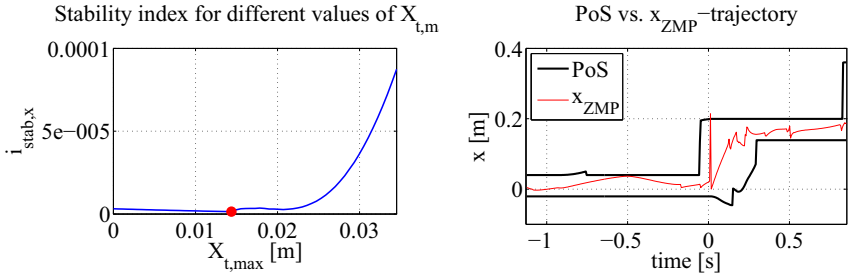
The movement of the torso in the z-direction during the start-up phase is designed exactly like for the steady-state walk.

I.1.3 Finding Parameters for the Start-up Phase

The first parameter to find that describes the start-up phase is $x_{\text{t,max}}$ which is varied within the ranges specified in Equation (I.6) in steps of $0.01(x_{\text{te}} - l_{\text{step}})$, giving a total of 61 simulations. The stability index is calculated using Equation (9.22), and the result of the simulations can be seen in Figure I.2(a), where the red dot marks the most stable trajectory. The value of $x_{\text{t,max}}$ is found to be:

$$x_{\text{t,max}} = 14.4\,\text{mm} \tag{I.9}$$

The ZMP trajectory for this simulation, where the start-up phase and the first step cycle is simulated, is plotted in Figure I.2(b) together with the PoS. It can be seen the ZMP is inside the PoS during the entire start-up phase.

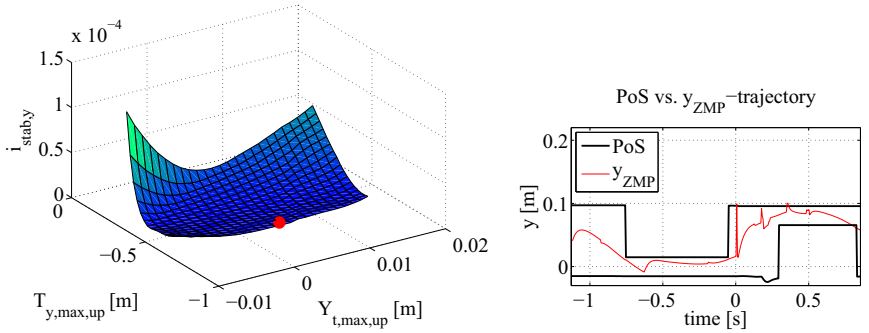


(a) Stability (b) PoS and x_{ZMP}

Figure I.2: *I.2(a) shows the stability index, $i_{stab,x}$, for the start-up phase. The red dot marks the most stable trajectory. I.2(b) shows the trajectory of x_{ZMP} during the start-up phase and the first step cycle, this is plotted together with the PoS.*

Next the values of $y_{\text{t,max,up}}$ and $t_{\text{y,max,up}}$ are found through simulations. The two values are varied within the ranges specified in Equation (9.18) on page 121 and (I.8). $y_{\text{t,max,up}}$ is varied in steps of $0.025y_{\text{t,mid}}$ and $t_{\text{y,max,up}}$ in steps of $0.025t_{\text{init}}$, making a total of 425 simulations. The resulting stability index, $i_{\text{stab,y}}$, of these simulations can be seen in Figure I.3(a), where the red dot marks the trajectory having the smallest stability index, thus the best stability properties. The stability index is calculated using Equation (9.2) on page 115. The values of $y_{\text{t,max,up}}$ and $t_{\text{y,max,up}}$ are found to be:

$$\begin{aligned} y_{\text{t,max,up}} &= 4.1\,\text{mm} \\ t_{\text{y,max,up}} &= -0.635\,\text{s} \end{aligned} \tag{I.10}$$



(a) Stability (b) PoS and y_{ZMP}

Figure I.3: *I.3(a) shows the stability index, $i_{stab,y}$, for the start-up phase. The red dot marks the most stable. I.3(b) shows the trajectory of y_{ZMP} during the start-up phase and the the first step cycle, this is plotted together with the PoS.*

In Figure I.3(b) the ZMP trajectory is plotted together with the PoS, and it can be seen that the ZMP is inside the PoS during the entire start-up phase. At $t = 0$, a large spike can be observed, this spike occurs because there is a phase shift model, where the supporting foot is changed, and is therefore not a spike that will occur in the real system, since there is no sudden phase shifts.

I.2 Establishing the Trajectories for the Stop Phase

The purpose of the stop phase is to bring the robot from the steady state walk to the initial posture, when it is desired to stop walking. The approach is exactly the same as for the start-up phase, a number of parameters describing the trajectories are altered, and a simulation is performed with each value of the parameter, to finally select the value given the best stability. As for the start-up phase, there will be no considerations toward energy consumption, since it is desired to use the trajectories providing the largest stability margin.

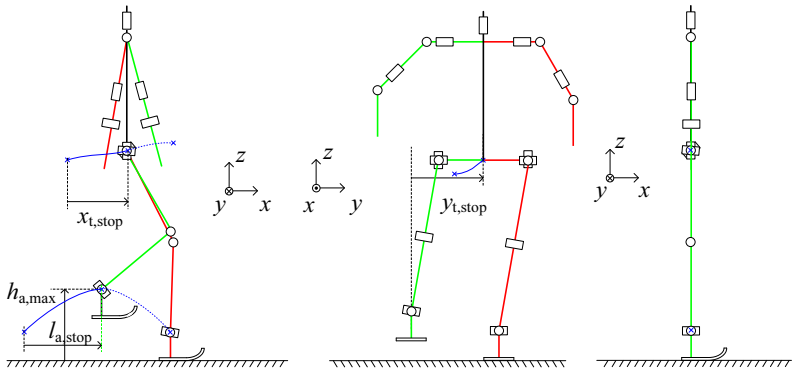
The movement in the sagittal plane during the stop phase is illustrated in Figure I.4(a) and I.4(c), where the last one illustrates the goal position, namely the biped robot placed at rest in its initial posture. Figure I.4(b) illustrates the movement in the frontal plane, where the torso reaches its maximum deviation to the left, $y_{\text{t,stop}}$. As seen the torso passes through the point $(x, y) = (x_{\text{t,stop}}, y_{\text{t,stop}})$ at time $t_{\text{stop,max}}$, and these are the two values that are changed in order to obtain the most stable motion during the stop phase. The two values are changed within the ranges:

$$0.65 l_{\text{step}} \leq x_{\text{t,stop}} \leq 0.95 l_{\text{step}} \tag{I.11}$$

$$1.6 y_{\text{t,mid}} \leq y_{\text{t,stop}} \leq 2.2 y_{\text{t,mid}} \tag{I.12}$$

I.2.1 Finding Parameters for the Stop Phase

The stop phase is described by two parameters, the one is $x_{\text{t,stop}}$ which is varied within the ranges specified in Equation (I.11) in steps of $0.01 l_{\text{step}}$, giving a total of 31 simulations. The stability index is calculated using Equation (9.22) on page 123, and the



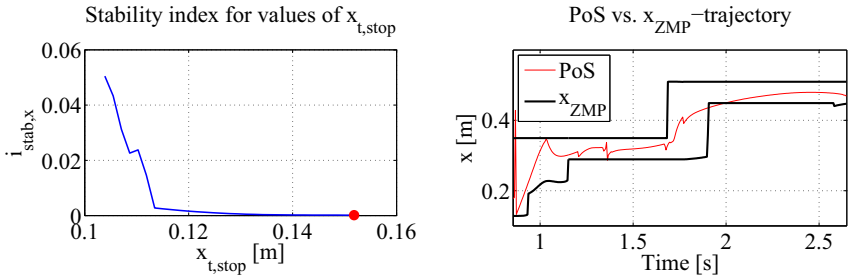
(a) SSP: $t = t_{stop,max}$ (b) SSP: $t = t_{stop,max}$ (c) Initial posture: $t = t_{stop}$

Figure I.4: *Illustration of the stop phase. The biped robot should be moved from steady state walk to the position shown in I.1(b) the ankle reaches its maximum point, $h_{a,max}$, and the torso reaches $x_{t,stop}$. At the same time the torso reaches its maximum deviation to the left, $y_{t,stop}$, as shown in I.4(b). Finally the biped robot reaches its initial posture as shown in I.4(c).*

result of the simulations can be seen in Figure I.5(a), where the red dot marks the most stable trajectory. The value of $x_{t,stop}$ is found to be:

$$x_{t,stop} = 151.8\,\text{mm} \tag{I.13}$$

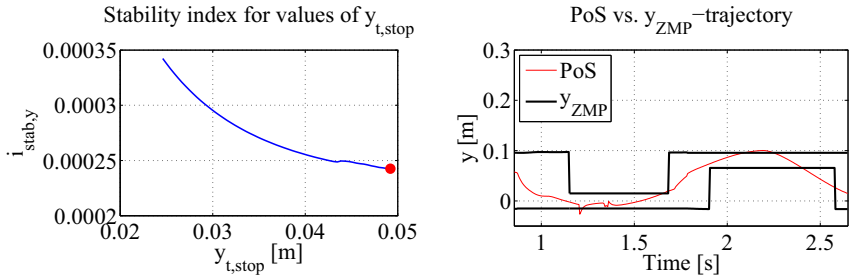
The simulation consisted of a start-up phase, one walking cycle and a stop phase. Note that only the data from the last step and the stop phase is used, when calculating the stability index for the stop phase. This ZMP-trajectory is plotted in Figure I.5(b) together with the PoS.



(a) Stability (b) PoS and x_{ZMP}

Figure I.5: *I.5(a) shows the stability index, $i_{stab,x}$, for the stop phase. The red dot marks the most stable. I.5(b) shows the trajectory of x_{ZMP} during the stop phase and the preceding step, this is plotted together with the PoS.*

Next the value of $y_{t,stop}$ is found through simulations. $y_{t,stop}$ is varied within the ranges specified in Equation (I.12). $y_{t,stop}$ is varied in steps of $0.01 y_{t,mid}$, making a total of 61 simulations. The resulting stability index, $i_{stab,y}$, of these simulations can be seen in Figure I.3(a), where the red dot marks the trajectory having the smallest stability index, thus the best stability properties. $i_{stab,y}$ is calculated using Equation (9.2) on page 115.



(a) Stability (b) PoS and y_{ZMP}

Figure I.6: *I.6(a) shows the stability index, $i_{stab,y}$, for the start-up phase. The red dot marks the most stable. I.6(b) shows the trajectory of y_{ZMP} during the start-up phase and the the first step cycle, this is plotted together with the PoS.*

The value of $y_{\text{t,stop}}$ is found to be:

$$y_{\text{t,stop}} = 90.2\,\text{mm} \tag{I.14}$$

In Figure I.6(b) the ZMP trajectory is plotted together with the PoS.

I.3 Polynomials from first Iteration

In Section 9.3 on page 122 the steady-state walking trajectories for the torso and the feet are found. In Section I.1 they are found for the start-up phase, and in Section I.2 they are found for the stop phase. In this section the polynomial description of the trajectories of the biped robot is presented. It is organized such that all the polynomials for the feet will be presented first, and then the all the polynomials for the torso will be presented.

I.3.1 Polynomials for the Feet

Steady State Walk

The polynomials describing the orientation of the foot are given as:

$$\theta_{\text{a,y}}(t) = \begin{cases} -560.806t^3 + 143.776t^2 & ,t \in [t_1, t_2] \\ 204.375t^3 - 73.355t^2 + 1.400 & ,t \in [t_2, t_3] \\ 0 & ,t \in [t_3, T_{\text{cycle}}] \end{cases} \tag{I.15}$$

and the polynomials describing the trajectories of Figure 9.10 are:

$$x_{\text{a}}(t) = \begin{cases} a_1 \sin(\theta_{\text{a,y}}(t)) + l_{\text{toe}} \sin(\theta_{\text{a,y}}(t)) & ,t \in [t_1, t_2] \\ -8.753t^3 + 4.600t^2 + 0.080 & ,t \in [t_2, t_3] \\ 1.348t^3 - 1.684t^2 + 0.698t + 0.224 & ,t \in [t_3, t_4] \\ 2l_{\text{step}} & ,t \in [t_4, T_{\text{cycle}}] \end{cases} \tag{I.16}$$

$$z_{\text{a}}(t) = \begin{cases} a_1 \cos(\theta_{\text{a,y}}(t)) + l_{\text{toe}}(1 - \cos(\theta_{\text{a,y}}(t))) & ,t \in [t_1, t_2] \\ -5.646t^3 + 2.027t^2 + 0.048 & ,t \in [t_2, t_3] \\ 0.880t^3 - 0.586t^2 + 0.086 & ,t \in [t_3, t_4] \\ a_1 & ,t \in [t_4, T_{\text{cycle}}] \end{cases} \tag{I.17}$$

It should be noted that the polynomials in Equation (I.15), (I.16) and (I.17) are for the right foot, but can be applied on the left foot by time shifting them one step cycle, T_{step}.

The reason that the number of polynomials are not the same in Equation (I.15), (I.16) and (I.17) is that the number of requirements are different.

Start-up Phase

The orientation of the foot is not altered during the start-up, thus:

$$\theta_{\text{a,y}}(t) = 0 \ \ , t \in [t_{\text{start}}, 0] \tag{I.18}$$

and the polynomials describing the position of the ankle are:

$$x_{\text{a}}(t) = \begin{cases} 0 & , t \in [t_{\text{start}}, t_{\text{up}}] \\ -0.365t^3 + 0.530t^2 & , t \in [t_{\text{up}}, t_{\text{upmax}}] \\ -0.398t^3 + 0.080t^2 + 0.250t + 0.064 & , t \in [t_{\text{upmax}}, 0] \end{cases} \tag{I.19}$$

$$z_{\text{a}}(t) = \begin{cases} a_1 & , t \in [t_{\text{start}}, t_{\text{up}}] \\ -0.929t^3 + 0.611t^2 + 0.048 & , t \in [t_{\text{up}}, t_{\text{upmax}}] \\ 0.633t^3 - 0.533t^2 + 0.032t + 0.086 & , t \in [t_{\text{upmax}}, 0] \end{cases} \tag{I.20}$$

Note that Equation (I.18), (I.19) and (I.20) are for the left foot, since the walking cycle is always initiated by moving the left foot. Note that the time index for the start-up phase is negative, this is because the steady-state walking cycle is defined to start at $t = 0$.

Stop Phase

The orientation of the foot is not altered during the stop phase, thus:

$$\theta_{\text{a,y}}(t) = 0 \ \ , t \in [T_{\text{cycle}}, t_{\text{stop}}] \tag{I.21}$$

and the polynomials describing the position of the ankle are:

$$x_{\text{a}}(t) = \begin{cases} -0.077t^3 + 0.326t^2 & , t \in [T_{\text{cycle}}, t_{\text{stop,max}}] \\ -0.693t^3 + 0.217t^2 + 0.255t + 0.0639 & , t \in [t_{\text{stop,max}}, t_{\text{stop}}] \end{cases} \tag{I.22}$$

$$z_{\text{a}}(t) = \begin{cases} -0.744t^3 + 0.524t^2 + 0.048 & , t \in [T_{\text{cycle}}, t_{\text{stop,max}}] \\ 0.744t^3 - 0.524t^2 + 0.086 & , t \in [t_{\text{stop,max}}, t_{\text{stop}}] \end{cases} \tag{I.23}$$

Note that Equation (I.21), (I.22) and (I.23) are for the right foot, since the walking cycle is always ended with moving the right foot, such that the biped robot will stop in initial posture.

I.3.2 Polynomials for the Torso

The polynomials describing the movement of the torso will be presented below.

Steady State Walk

The polynomials describing the chosen trajectories of the torso are given as:

$$x_{\text{t}}(t) = \begin{cases} -4.761t^3 + 0.698t^2 + 0.366t + 0.225 & , t \in [t_1, t_2] \\ 0.553t^3 - 0.436t^2 + 0.187t + 0.125 & , t \in [t_2, t_4] \end{cases} \tag{I.24}$$

$$y_{\text{t}}(t) = \begin{cases} -0.250t^2 + 0.213t + 0.041 & , t \in [t_1, t_4] \end{cases} \tag{I.25}$$

$$z_{\text{t}}(t) = \begin{cases} -0.869t^3 + 0.535t^2 + 0.240 & , t \in [t_1, t_3] \\ 0.684t^3 - 0.456t^2 + 0.270 & , t \in [t_3, t_4] \end{cases} \tag{I.26}$$

Start-up Phase

The polynomials for the torso during the start-up phase are as follows:

$$x_{\text{t}}(t) = \begin{cases} -0.178t^3 + 0.147t^2 & ,t \in [t_{\text{start}}, t_{\text{stop,max}}] \\ 0.673t^3 - 0.173t^2 - 0.016t + 0.014 & ,t \in [t_{\text{stop,max}}, 0] \end{cases} \tag{I.27}$$

$$y_{\text{t}}(t) = \begin{cases} 0.597t^3 - 0.442t^2 + 0.041 & ,t \in [t_{\text{start}}, t_{\text{stop,max}}] \\ 0.248t^3 - 0.068t^2 + 0.005 & ,t \in [t_{\text{stop,max}}, 0] \end{cases} \tag{I.28}$$

$$z_{\text{t}}(t) = \begin{cases} 0.046t^3 - 0.078t^2 + 0.273 & ,t \in [t_{\text{start}}, 0] \end{cases} \tag{I.29}$$

Stop Phase

The polynomials for the torso during the stop phase are as follows:

$$x_{\text{t}}(t) = \begin{cases} 0.170t^3 - 0.435t^2 + 0.368t + 0.058 & ,t \in [T_{\text{cycle}}, t_{\text{stop,max}}] \\ 0.169t^3 - 0.195t^2 + 0.071t + 0.152 & ,t \in [t_{\text{stop,max}}, t_{\text{stop}}] \end{cases} \tag{I.30}$$

$$y_{\text{t}}(t) = \begin{cases} -0.233t^3 - 0.116t^2 + 0.211t + 0.041 & ,t \in [T_{\text{cycle}}, t_{\text{stop,max}}] \\ 0.709t^3 - 0.444t^2 - 0.053t + 0.090 & ,t \in [t_{\text{stop,max}}, t_{\text{stop}}] \end{cases} \tag{I.31}$$

$$z_{\text{t}}(t) = \begin{cases} -0.080t^3 + 0.113t^2 + 0.240 & ,t \in [T_{\text{cycle}}, t_{\text{stop}}] \end{cases} \tag{I.32}$$

I.3.3 Implementation of Trajectories

To ease the workload on the on-board computer, the trajectories are not implemented like the polynomials described above, but are implemented as a look-up table. The values in the look-up table are calculated from the polynomials. The implementation is illustrated in Figure I.7, where it can be seen that when the walk is initiated the data for the start-up phase is executed, then the data for the steady state walk is executed cyclic, until it is chosen to terminate the walk, and then the data for the stop phase is executed.

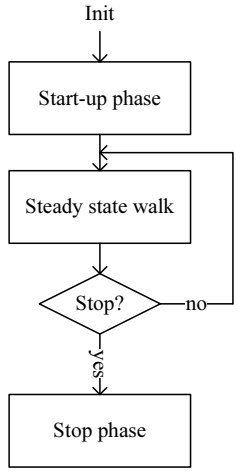


Figure I.7: *The trajectories are implemented as look-up tables, where the data for the start-up phase is executed when the walk is initiated, the data for the steady state walk is used in a cyclic manner, until the it is chosen to stop walking, and at this time the data for the stop phase is used.*

Figure I.7 illustrates why it is important that the trajectories for the start-up phase and the stop phase are designed to fit the steady state walk, and why it is important that the trajectories for the steady state walk are cyclic.

I.4 Derivation of Slower Trajectories

As found in Section 9.5 on page 128 the developed trajectories are not suitable for obtaining dynamical walk, with the given actuators. It is therefore chosen to develop a new set of trajectories, that are more adapted to the servo motors at hand. It is chosen to decrease the walking speed, and in response to this also the step length. Through simulations it is found that the new walking speed and the new step length are:

$$v_{\text{walking}} = 9.35\,\text{cm/s} \tag{I.33}$$

$$l_{\text{step}} = 8\,\text{cm} \tag{I.34}$$

The establishment of the new trajectories are exactly the same as for the first iteration, the same parameters are varied, the same simulations are performed, the same criteria are used to single out the best values of the parameters. Only one change has been made, the reference to the servo motor model are given with a frequency of $40\,\text{Hz}$ as on the real system. This way the unfavourable property is taken into consideration when the parameters are determined. Since the approach is the same as for the first iteration, it will not be examined once again, but only the result will be presented below.

I.4.1 Polynomials for the Feet

Steady State Walk

The polynomials describing the trajectories of the foot are given as:

$$x_{\text{a}}(t) = \begin{cases} 0 & ,t \in [t_1, t_2] \\ -0.732t^3 + 0.769t^2 & ,t \in [t_2, t_3] \\ 0.113t^3 - 0.282t^2 + 0.233t + 0.096 & ,t \in [t_3, t_4] \\ 2l_{\text{step}} & ,t \in [t_4, T_{\text{cycle}}] \end{cases} \tag{I.35}$$

$$z_{\text{a}}(t) = \begin{cases} a_1 & ,t \in [t_1, t_2] \\ -0.704t^3 + 0.506t^2 + 0.048 & ,t \in [t_2, t_3] \\ 0.110t^3 - 0.147t^2 + 0.086 & ,t \in [t_3, t_4] \\ a_1 & ,t \in [t_4, T_{\text{cycle}}] \end{cases} \tag{I.36}$$

Start-up Phase

The orientation of the foot is not altered during the start-up, thus:

$$\theta_{\text{a,y}}(t) = 0 \ \ ,t \in [t_{\text{start}}, 0] \tag{I.37}$$

and the polynomials describing the position of the ankle are:

$$x_{\text{a}}(t) = \begin{cases} 0 & ,t \in [t_{\text{start}}, t_{\text{up}}] \\ 0.025t^3 + 0.003t^2 & ,t \in [t_{\text{up}}, t_{\text{upmax}}] \\ -0.058t^3 + 0.065t^2 + 0.056t + 0.016 & ,t \in [t_{\text{upmax}}, 0] \end{cases} \tag{I.38}$$

$$z_{\text{a}}(t) = \begin{cases} a_1 & ,t \in [t_{\text{start}}, t_{\text{up}}] \\ -0.116t^3 + 0.153t^2 + 0.048 & ,t \in [t_{\text{up}}, t_{\text{upmax}}] \\ 0.079t^3 - 0.133t^2 + 0.016t + 0.086 & ,t \in [t_{\text{upmax}}, 0] \end{cases} \tag{I.39}$$

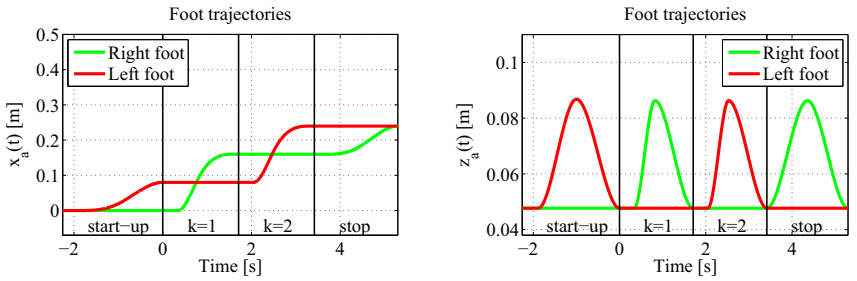
Stop Phase

The polynomials describing the position of the ankle are:

$$x_{\mathrm{a}}(t) = \begin{cases} 0.034t^3 - 0.014t^2 & ,t \in [T_{\mathrm{cycle}}, t_{\mathrm{stop,max}}] \\ -0.082t^3 + 0.081t^2 + 0.064t + 0.016 & ,t \in [t_{\mathrm{stop,max}}, t_{\mathrm{stop}}] \end{cases} \tag{I.40}$$

$$z_{\mathrm{a}}(t) = \begin{cases} -0.093t^3 + 0.131t^2 + 0.048 & ,t \in [T_{\mathrm{cycle}}, t_{\mathrm{stop,max}}] \\ 0.093t^3 - 0.131t^2 + 0.086 & ,t \in [t_{\mathrm{stop,max}}, t_{\mathrm{stop}}] \end{cases} \tag{I.41}$$

The new trajectories for the feet can be seen i Figure I.8.



(a) Trajectories for x-direction

(b) Trajectories for z-direction

Figure I.8: *Trajectories for the right and left foot for start-up, one walking cycle $k = [1, 2]$ and the stop phase. I.8(a) shows the trajectories in the x-direction, and I.8(b) shows the trajectories in the z-direction.*

I.4.2 Polynomials for the Torso

The polynomials describing the movement of the torso will be presented below.

Steady State Walk

The polynomials describing the chosen trajectories of the torso are given as:

$$x_{\mathrm{t}}(t) = \begin{cases} 0.061t^3 - 0.096t^2 + 0.047t + 0.064 & ,t \in [t_1, t_2] \\ -0.525t^3 + 0.154t^2 + 0.126t + 0.104 & ,t \in [t_2, t_4] \end{cases} \tag{I.42}$$

$$y_{\mathrm{t}}(t) = \{ \; -0.062t^2 + 0.106t + 0.041 \quad ,t \in [t_1, t_4] \tag{I.43}$$

$$z_{\mathrm{t}}(t) = \begin{cases} -0.109t^3 + 0.134t^2 + 0.240 & ,t \in [t_1, t_3] \\ 0.086t^3 - 0.114t^2 + 0.270 & ,t \in [t_3, t_4] \end{cases} \tag{I.44}$$

Start-up Phase

The polynomials for the torso during the start-up phase are as follows:

$$x_{\mathrm{t}}(t) = \begin{cases} -0.022t^3 + 0.032t^2 & ,t \in [t_{\mathrm{start}}, t_{\mathrm{stop,max}}] \\ 0.071t^3 - 0.046t^2 - 0.017t + 0.084 & ,t \in [t_{\mathrm{stop,max}}, 0] \end{cases} \tag{I.45}$$

$$y_{\mathrm{t}}(t) = \begin{cases} 0.020t^3 - 0.052t^2 + 0.041 & ,t \in [t_{\mathrm{start}}, t_{\mathrm{stop,max}}] \\ -0.217t^3 + 0.277t^2 - 0.008 & ,t \in [t_{\mathrm{stop,max}}, 0] \end{cases} \tag{I.46}$$

$$z_{\mathrm{t}}(t) = \{ \; 0.006t^3 - 0.020t^2 + 0.273 \quad ,t \in [t_{\mathrm{start}}, 0] \tag{I.47}$$

Stop Phase

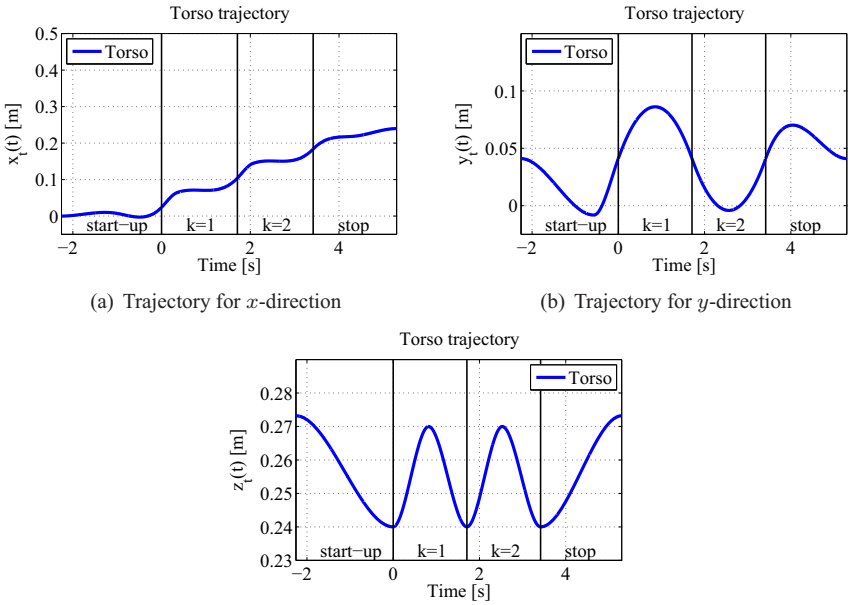
The polynomials for the torso during the stop phase are as follows:

$$x_{\mathrm{t}}(t) = \begin{cases} 0.073t^3 - 0.163t^2 + 0.126t + 0.024 & ,t \in [T_{\mathrm{cycle}}, t_{\mathrm{stop,max}}] \\ -0.035t^3 + 0.043t^2 + 0.013t + 0.059 & ,t \in [t_{\mathrm{stop,max}}, t_{\mathrm{stop}}] \end{cases} \tag{I.48}$$

$$y_{\mathrm{t}}(t) = \begin{cases} 0.030t^3 - 0.112t^2 + 0.105t + 0.041 & ,t \in [T_{\mathrm{cycle}}, t_{\mathrm{stop,max}}] \\ 0.030t^3 - 0.028t^2 - 0.026t + 0.066 & ,t \in [t_{\mathrm{stop,max}}, t_{\mathrm{stop}}] \end{cases} \tag{I.49}$$

$$z_{\mathrm{t}}(t) = \begin{cases} -0.010t^3 + 0.028t^2 + 0.240 & ,t \in [T_{\mathrm{cycle}}, t_{\mathrm{stop}}] \end{cases} \tag{I.50}$$

The new trajectories for the torso can be seen in Figure I.9.



(a) Trajectory for x-direction

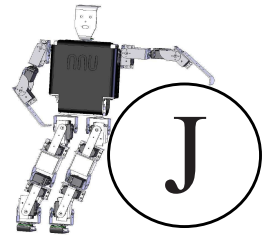
(b) Trajectory for y-direction

(c) Trajectory for z-direction

Figure I.9: *Trajectories for the torso, for start-up phase, the following walking cycle $k = [1, 2]$, and stop phase. I.9(a) shows the trajectory in the x-direction. I.9(b) shows the trajectory in the y-direction. I.9(c) shows the trajectory in the z-direction.*

Comparing the trajectories of this second iteration with those of the first iteration, it can be seen that most of the movement of the torso is performed during the DSP. This is a result from clocking in the trajectories with a frequency of $40\,\mathrm{Hz}$, which makes the ZMP jump back and forth. The largest stability is obtained if the largest movements are made in the DSP.

VERIFICATION OF THE ZMP ESTIMATORS



J

This appendix supports Section 10.2 on page 135 and contains verification of the developed ZMP estimator.

In order to verify the developed ZMP estimators, the model of the robot is used to determine a reference ZMP trajectory that can be compared with the output of the estimator. The test scenario will consist of moving the torso along the x- and y-axis. The test will only be done when in DSP since it was not possible to make the robot balance on one leg. The test has been divided into two sub test:

- **Verification scenario one:** The first scenario moves the torso in the sagittal direction and can be is seen from Figure J.1.
- **Verification scenario two:** The second scenario moves the torso in the frontal direction, which is seen from Figure J.2.

The verification trajectory can be found on the enclosed CD in `Simulink/Test-Trajectories`. When the inputs were chosen it was noted, that even though an input given to the model kept the ZMP within the PoS the same input did not give a stable posture of the real robot. This was caused by backlash, mainly in the hips. The backlash made the actual robot lean its torso further backward than in the model which made it unable to stand. Therefore the inputs were altered such that the robot was able to stand by leaning the torso forward and afterward these inputs were used in a simulation to determine the ZMP trajectory.

Verification scenario one and scenario two are shown in Figure J.3 and J.4 respectively where the estimation of the ZMP is done using the accelerometer. The figures show that the estimate is unreliable. This can not be contributed to the backlash alone. The estimator is sligthly better at estimating a static ZMP, however when trying to estimate

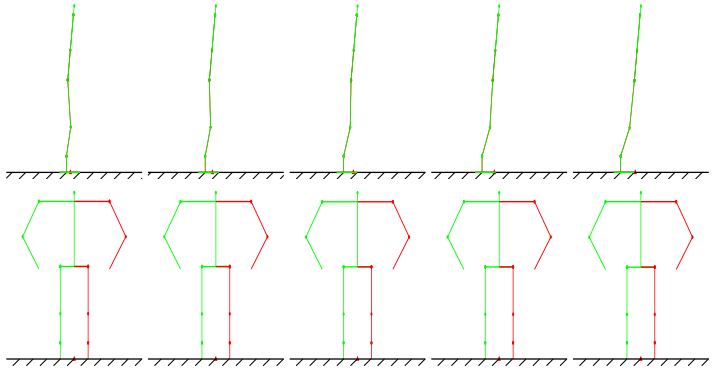


Figure J.1: *Animation of the movements of the robot used in the verification of the ZMP estimator in the sagittal direction. The five figures above show the movements in the sagittal plane, while the five figures below show the movements in the frontal plane.*

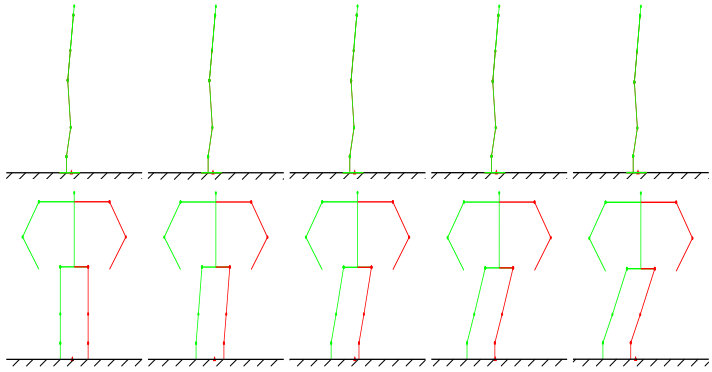


Figure J.2: *Animation of the movements of the robot used in the verification of the ZMP estimator in the frontal direction. The five figures above show the movements in the sagittal plane, while the five figures below show the movements in the frontal plane.*

a moving ZMP the estimator shows a very poor result. The estimator shows better results when used in a simulation of the robot as shown in Figure 10.6, suggesting that the problem does not lie in the approach but in the inputs. Figure J.5 shows the measured angles from the servo motors which are used in Equation (10.10) to calculate the position of the CoM of the torso.

Because of Equation (10.10) the noise becomes highly correlated in the position leading to the poor results seen in Figures J.3 and J.4.

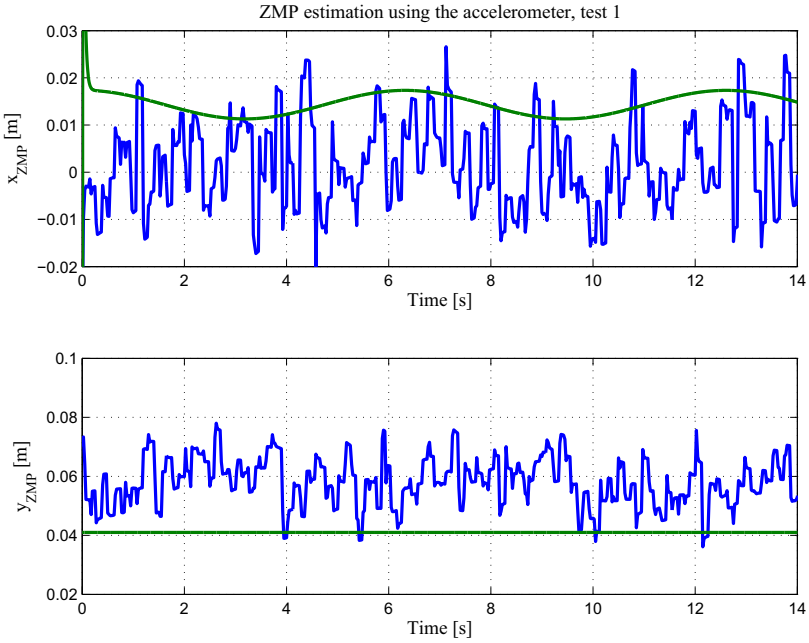


Figure J.3: *The ZMP estimate using the accelerometer in test 1. The green line shows the simulated ZMP and the blue line shows the estimated ZMP.*

Figure J.6 and J.7 show verification scenario one and scenario two where the pressure sensors are used. As expected the ZMP estimator does not yield the same result as the model. The ZMP in the x-axis shows that the estimated ZMP lies approximately $2\,\mathrm{cm}$ behind the simulated ZMP. In the y-axis the estimated ZMP lies about $5\,\mathrm{mm}$ to the left of what was simulated. It should however be noted that the amplitude of the ZMP estimate is almost identical to the simulated ZMP. Since the ZMP estimator uses the pressure measurements from the feet, this estimate will yield the actual position of the ZMP. Therefore the ZMP estimator can be considered accurate and able to detect the deviation between the ZMP from the model and the real ZMP on the robot. The deviation consists in the fact that backlash and bending of the strain plates is not included in the dynamical model.

The ZMP estimator that uses the pressure sensors is only able to measure the ZMP within the PoS and measuring FZMP is therefore not possible. However the ZMP

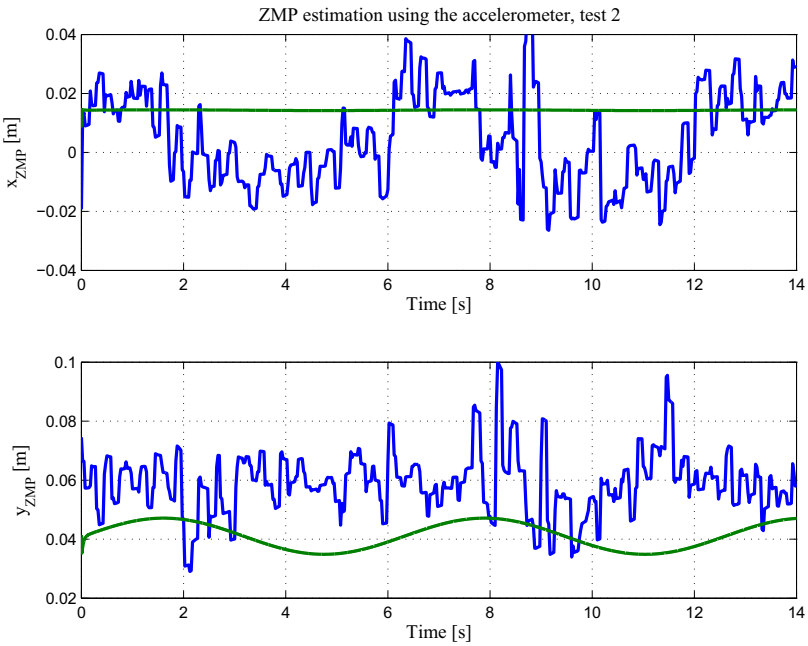


Figure J.4: *The ZMP estimate using the accelerometer in test 2. The green line shows the simulated ZMP and the blue line shows the estimated ZMP.*

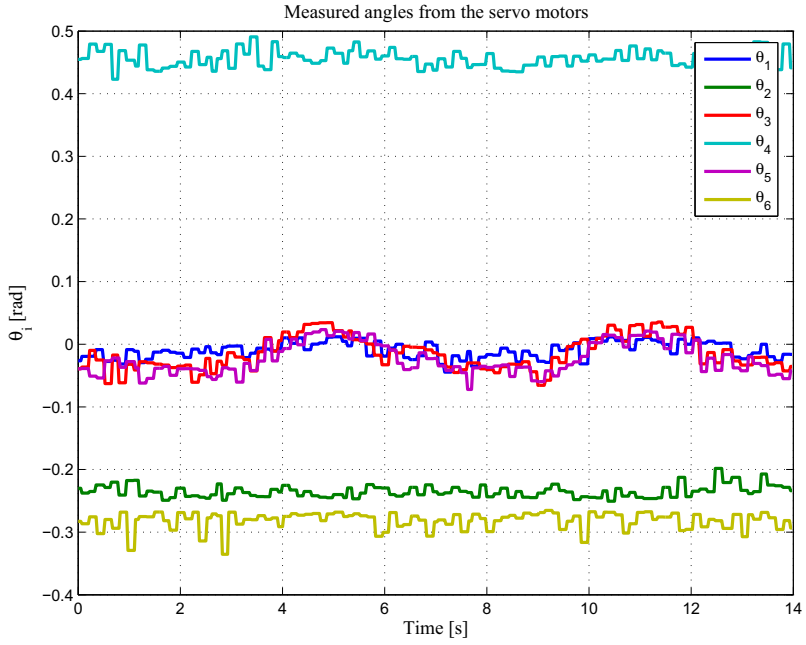


Figure J.5: *The measured angles from the servo motors that are used to estimate the ZMP.*

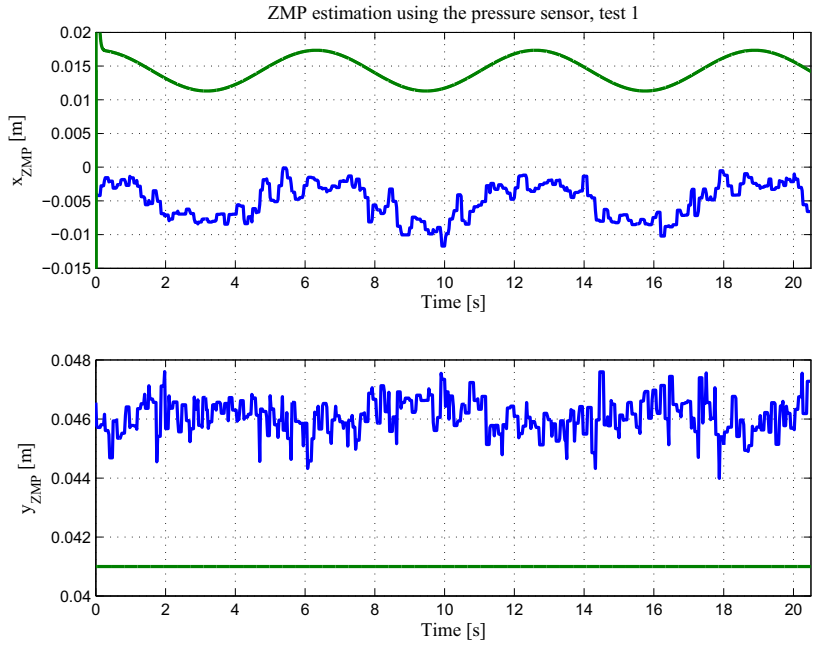


Figure J.6: *The ZMP estimate using the pressure sensors in test 1. The green line line shows the simulated ZMP and the blue line shows the estimated ZMP.*

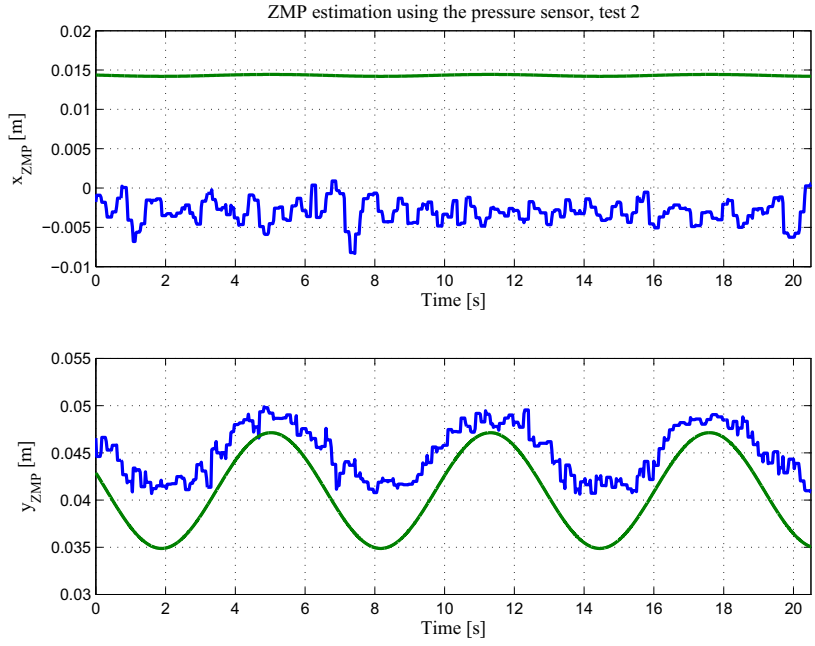
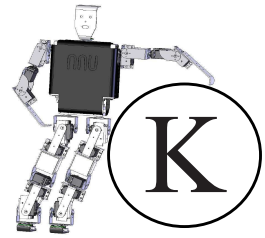


Figure J.7: *The ZMP estimate using the pressure sensors in test 2. The green line line shows the simulated ZMP and the blue line shows the estimated ZMP.*

estimator that uses the accelerometer is able to measure FZMP and it would be advantageous to fuse the two estimates. Because of the reliablity issues regarding the FZMP estimate it is chosen to only use the ZMP estimator that uses the pressure sensors.

CONTROLLER SIMULATION AND VERIFICATION



This Appendix supports Chapter 10 on page 133 where the ZMP controller and hybrid posture controller is designed and implemented. The simulations and tests are described, and the resulting system response is commented after each simulation or test.

K.1 ZMP Controller Simulation in DSP

The controller was implemented in the complete model of the system, and two experiments were conducted. First a $1\,\mathrm{cm}$ step in the ZMP reference is applied along the x-axis. The simulated performance of the controller is seen in Figure K.1.

Next, a $2\,\mathrm{cm}$ step was applied along the y-axis. The response of the system is shown in Figure K.2. The reason for using a larger step along the y-axis is that the PoS is greater than along the x-axis. Overshoot has to be avoided since this will make the robot fall. The cost of this is a slower controller. As it can be seen in Figure K.1 and K.2, the ZMP starts out by moving in the wrong direction. This is caused by the acceleration, and is therefore unavoidable, since it is not possible to move a body, at rest, without acceleration it. However, with the small gains, the effect can be kept to a minimum.

The controller gains were adjusted to minimize the effect of the ZMP going the wrong way the first few samples. When doing this, the stability is increased, but the controller performs much slower.

K.1.1 ZMP Controller Simulation in SSP

To test the ZMP controller in SSP, the robot was positioned in SSP and then the right leg was lifted $2\,\mathrm{cm}$ above the ground. After the robot movements had settled, a step was applied along the x-axis. The same method was used to test the ZMP controller

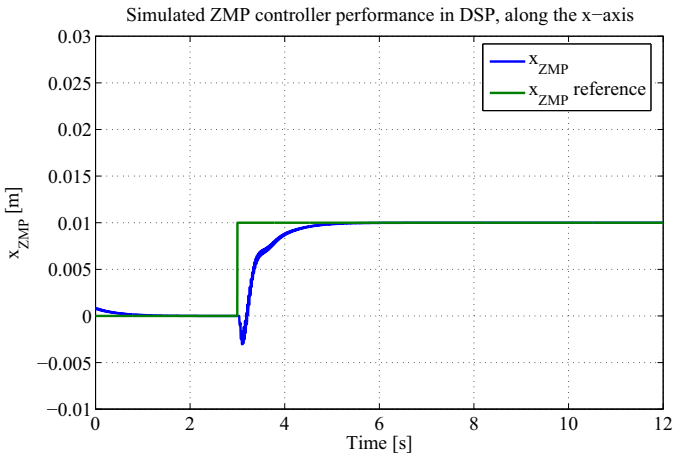


Figure K.1: *Simulating the ZMP controller in DSP by applying a step in x-direction.*

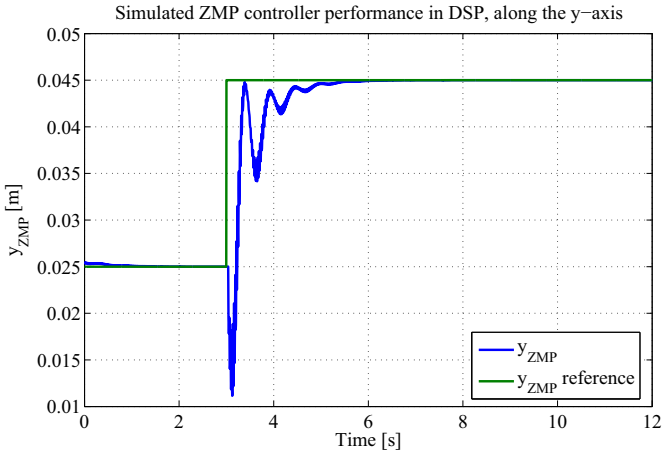


Figure K.2: *Simulating the ZMP controller in DSP by applying a step in y-direction.*

along the y-axis.

First, a $1\,\mathrm{cm}$ step was applied on the reference of the ZMP in the x-axis. The response of the system is shown in Figure K.3. Due to the acceleration, the position of the ZMP

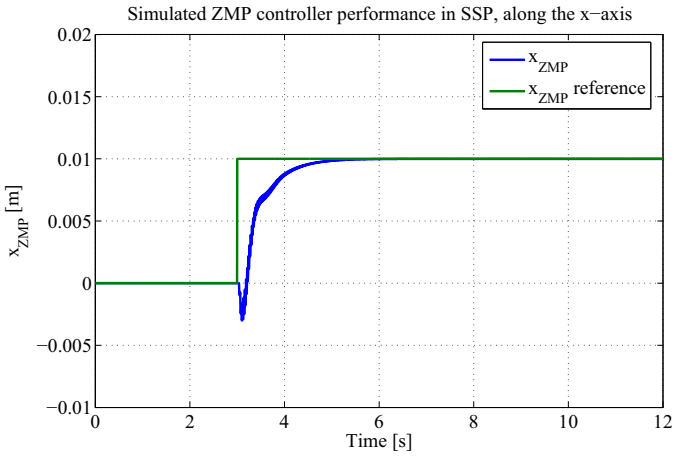


Figure K.3: *Simulating the ZMP controller in SSP by applying a step in x-direction.*

is moved in the opposite direction at first. Then the ZMP is moved in place within two seconds. Overshoot is not wanted for the ZMP controller as this would make the robot fall. At the cost of this, the controller performs slowly.

Next, a $0.5\,\mathrm{cm}$ step was applied along the y-axis. The reason for choosing a smaller step along the y-axis is due to the smaller PoS in SSP. The response of the system is shown in Figure K.4. Once again, overshoot has to be avoided to ensure that the

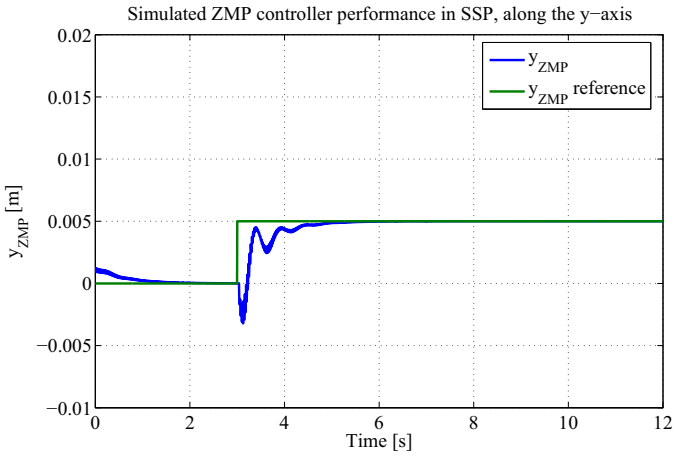


Figure K.4: *Simulating the ZMP controller in SSP by applying a step in y-direction.*

controller does not make the robot fall. The controller gains were increased as much

as possible to gain the fastest response times. It must however be concluded that a sampling frequency of $40\,\mathrm{Hz}$ is too slow for applications like a walking biped robot. It is doubtable that these SSP controllers would work if implemented on a walking biped robot due to their slow response times.

K.2 Implementation and Verification

The ZMP controller was implemented on the actual system and with minor changes to the controller gains ZMP control was realized. The controller was implemented as shown in Figure 10.13. To test the controller, steps were given to the reference position of the ZMP. Two steps were given in two different experiments. First, the desired ZMP position was moved in the x-direction by giving a step in the x-direction. Next, the controller was tested by applying a step in the y-direction.

K.2.1 Testing ZMP Controller in x-direction

In this test, a step of $2.5\,\mathrm{cm}$ on the reference to the ZMP is applied in the x-direction. The corresponding position of the ZMP can be seen in Figure K.5.

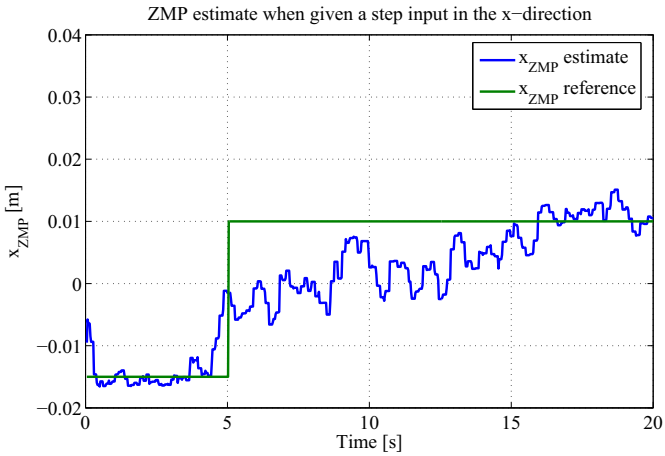


Figure K.5: *Testing the ZMP controller in DSP by applying a step in x-direction.*

It is seen that the ZMP controller manages to move the position of the ZMP to the desired location within $10\,\mathrm{s}$ which is too slow if the controller should be used while the robot is walking. One could argue that by increasing the gains, the controller would perform faster. This is also the case, but due to the limited PoS in the x-direction, the robot tends to fall if the ZMP is moved any faster than shown in Figure K.5. The $2.5\,\mathrm{cm}$ step is the largest step that can be applied to the robot if the robot should maintain balance.

When the step is applied, it is seen that the position of the ZMP jumps approximately $1.5\,\mathrm{cm}$ this is due to backlash in the joints. When the robot is at initial position, the robot can tilt freely around $5\,\mathrm{deg}$ so a sudden jump in the position of the ZMP is to be expected since the starting point of the robot was zero-position. The position of the ZMP is slowly changed, but the position also vary with a higher frequency at

approximately 1 Hz. This frequency is the natural frequency of the robot and occur due to wobbling and backlash in the joints as well as the strain plates in the feet that acts like three steel plate springs. The internal controller in the servo motors also contributes to this swinging of the robot. It has not been possible to retrieve any information regarding the internal controllers from the developing company.

K.2.2 Testing ZMP Controller in y-direction

In this test, a step was applied to the position of the ZMP in the y-direction. The step is applied with a magnitude of $2.5\,\mathrm{cm}$ compared to the starting position. The corresponding position of the ZMP can be seen in Figure K.6.

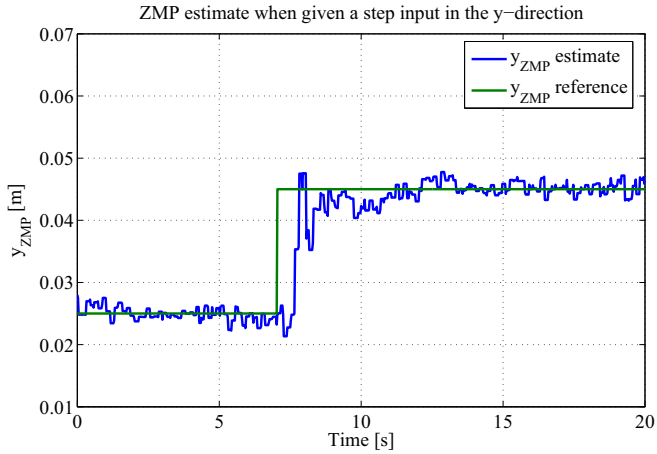


Figure K.6: *Testing the ZMP controller in DSP by applying two steps in y-direction.*

It is seen that the ZMP controller is faster in the y-direction than in the x-direction, and that it reaches the reference position after one second. This is mainly due to the controller gains that are larger than the controller gains used for the controller in the x-direction. This is possible since the PoS is much larger in the frontal plane than in the sagittal plane, when the robot is in DSP. As it can be seen in Table 10.3 the integral gain, is larger than k_{iy} for the other controller, and the proportional gain is likewise increased. It is due to the larger stability in the y-direction that these increased gains can be used.

K.3 Simulating the Posture Controller Performance

The performance of the hybrid posture controller was tested in both DSP and SSP. First, the controller was tested in DSP and steps were given to the inclination of the torso around the x-axis and around the y-axis in two different tests. Next, the posture controller was tested in SSP. To verify that the controller is capable of suppressing disturbances causes by backlash, 0.087 rad is subtracted from the inclination instead of giving a step with the reference. This will simulate of effect caused by backlash. This substitution will only be applied in SSP in around the x-axis, as backlash is most pronounced around this axis.

In all the simulations, it can be seen that the estimated inclination has a spike going the opposite direction than the reference inclination. This happens since the inclination is estimated using the accelerometer data. When accelerating in the opposite direction than the gravity vector, the estimated inclination will have this characteristics. This is not represented in the actual inclination of the torso, but only in the estimated inclination.

K.3.1 Posture Controller Simulation in DSP

In this test, a 0.09 rad step was first applied about the x-axis. This corresponds to approximately 5 deg. The robot was set in zero-position before the test was carried out. The response is seen in Figure K.7.

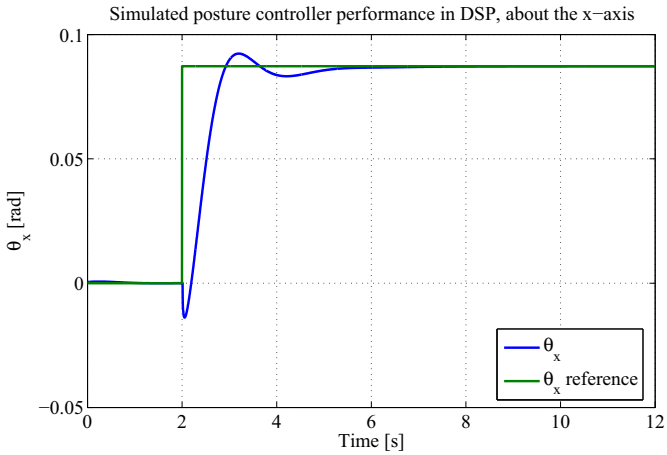


Figure K.7: *Simulating the posture controller in DSP by applying a step about the x-axis.*

It is seen that the robot reaches the reference inclination after one second with a slight overshoot. When tuning the controller, it is a trade-off between having a slight overshoot or a slow controller. It has been decided that the overshoot is acceptable at this level. Next, a 0.09 rad step was applied about the y-axis. In this test, the robot was also set in zero-position before the test was carried out. The response of the system is shown in Figure K.8. The characteristics of the response to this test are similar to the previous. Once again a minor overshoot is detected, but this is considered to be acceptable.

K.3.2 Posture Controller Simulation in SSP

This test differs from the test conducted in DSP, but only for the test regarding step-input about the x-axis. Instead of giving an inclination step input, 0.087 rad is subtracted from the inclination estimate. This simulates the behavior of backlash in the system. The posture controller will then suppress this disturbance by increasing the inclination in both the ankle and the torso as described in the design section. The response is seen in Figure K.9.

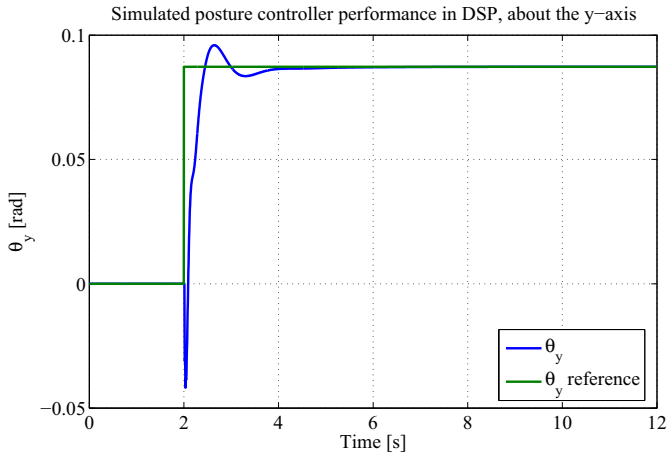


Figure K.8: *Simulating the posture controller in DSP by applying a step about the y-axis.*

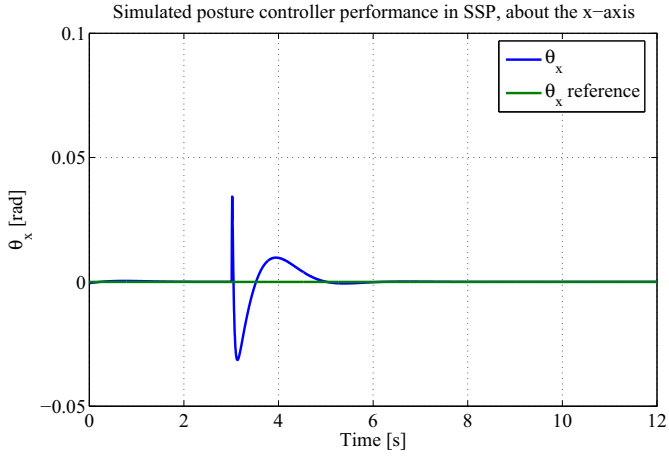


Figure K.9: *Simulating the posture controller in SSP by subtracting* 0.087 *rad from the inclination estimate.*

First, it is seen that the measured inclination increase, event though the error-inclination is subtracted. This is due to the sudden acceleration that occurs when the inclination of the torso is changed rapidly. This acceleration of the torso yields an error in the inclination estimate that is shown in the figure. Next, the inclination is estimated correctly and the controller begins to minimize the error. Once again it has been chosen to accept a minor overshoot to realize a faster settle time.

Finally, a step on 0.087 rad was applied about the y-axis. The response of the system is shown in Figure K.10.

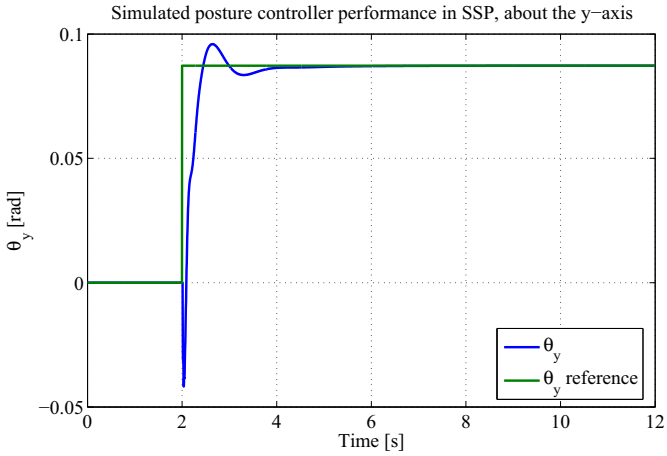


Figure K.10: *Simulating the posture controller in SSP by applying a step about the y-axis.*

This concludes that the posture controller is capable of controlling the model of the biped robot while standing in SSP. Only the controller for SSP-L is tested, since the controller for the right leg is similar, the simulation is not shown.

K.4 Implementation and Verification

This section describes the implementation of the DSP posture controller. The controller will be tested by two experiments, where a step in the inclination is given, in both the sagittal and the frontal plane individually. It was not possible to verify the performance of the SSP controller due to backlash and wobbling. This will be described more detailed in the following.

K.4.1 Testing the DSP Posture Controller About the x-axis

To test the posture controller, the robot was positioned in zero position. A 0.09 rad step was then given on the inclination reference about the x-axis. The measured inclination,$\theta_{\text{x,ref}}$, and reference input is seen in Figure K.11.

In this figure, it is seen that there is a 0.03 rad overshoot. This is primarily due to backlash and wobbling in the joints. It was however also noticed, that the spring plates located in the feet were bended when the torso was tilted, which contributes to the

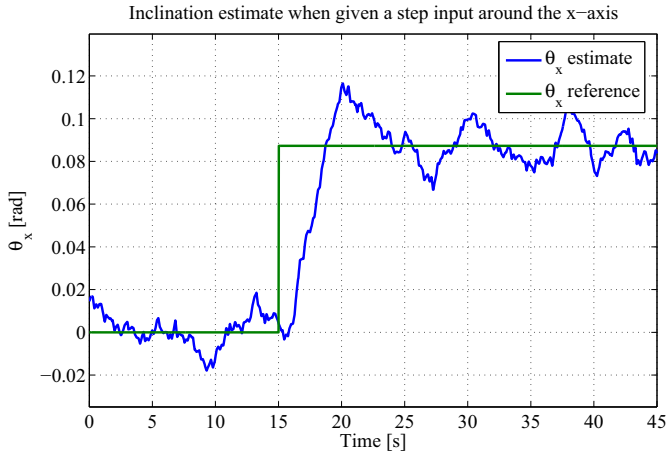


Figure K.11: *A 0.09 rad step input around x-axis is given after 15 seconds.*

overshoot. By adjusting the controller gains, it was not possible to achieve better performance than shown in the figure. If the controller gains were raised the movements became too rapid, and the robot could not maintain its balance. The settling time of the controller is too slow, to realize posture control while the robot is performing dynamic walk and is therefore not suitable for this application. This is mainly due to the slow control signal given by the inclination estimator. The data provided by the accelerometer is dominated by noise resulting in the need of a low-pass filter with a very low cut-off frequency. See Section 10.2.3 on page 140 for details regarding the inclination estimator.

K.4.2 Testing the DSP Posture Controller About the y-axis

To test the posture controller about the y-axis, a 0.09 rad step was given. The biped was placed in zero position with zero inclination. After 15 sec the step was applied. The resulting inclination is seen in Figure K.12.

Similar to the posture controller test around the y-axis, overshoot was seen. The overshoot is measured to 0.015 rad. The overshoot was attempted minimized by decreasing the controller gains with a slower controller at the cost. Similar to the posture controller around the x-axis it must be concluded that this controller is too slow to realize posture control while performing dynamic walk. Once again, backlash, wobbling in the joints and a slow inclination estimator is the main source for these problems.

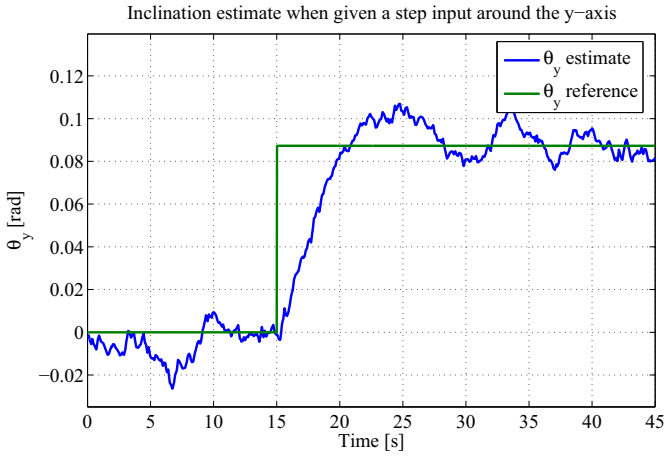


Figure K.12: *A 0.09 rad step input around y-axis is given after 15.*

JOINT LIMITATIONS

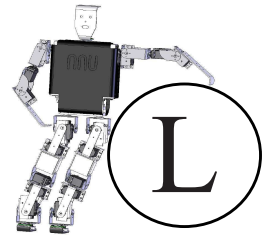
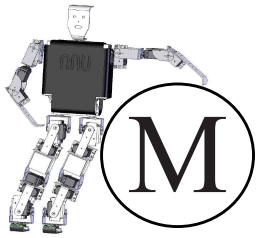


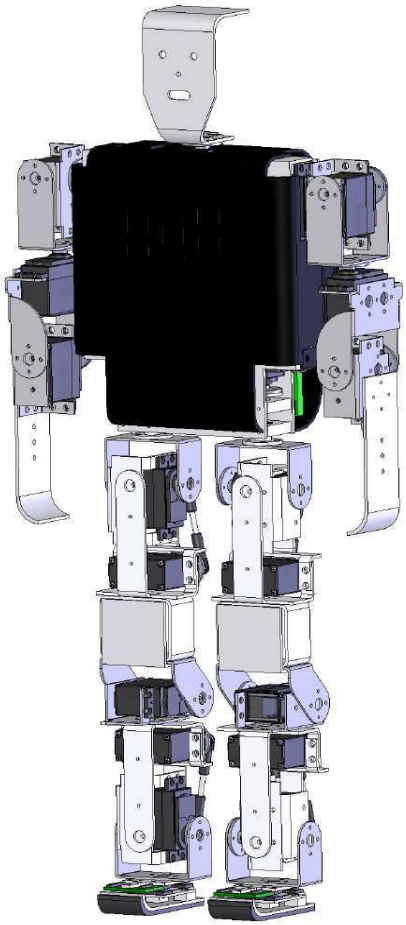
Table L.1 shows the joint limitations. The limitations are chosen such that the robot does not harm itself.

Joint limitations		
$-1.05 \leq$	θ_1	≤ 1.22
$-0.52 \leq$	θ_2	≤ 0.61
$-1.57 \leq$	θ_3	≤ 0.09
$-0.35 \leq$	θ_4	≤ 0.79
$-1.48 \leq$	θ_5	≤ 1.48
$-0.52 \leq$	θ_6	≤ 0.78
$-0.79 \leq$	θ_7	≤ 0.35
$-1.05 \leq$	θ_8	≤ 1.40
$-0.61 \leq$	θ_9	≤ 0.35
$-0.09 \leq$	θ_{10}	≤ 3.4
$-0.7 \leq$	θ_{11}	≤ 0.52
$-1.13 \leq$	θ_{12}	≤ 0.79
$-1.57 \leq$	θ_{13}	≤ 0.52
$-1.39 \leq$	θ_{14}	≤ 1.57
$-1.48 \leq$	θ_{15}	≤ 1.48
$0.17 \leq$	θ_{16}	≤ 1.66
$-1.48 \leq$	θ_{17}	≤ 1.48
$-1.55 \leq$	θ_{18}	≤ 1.40
$-1.40 \leq$	θ_{19}	≤ 1.40
$0.17 \leq$	θ_{16}	≤ 1.66
$-0.79 \leq$	θ_{21}	≤ 0.79

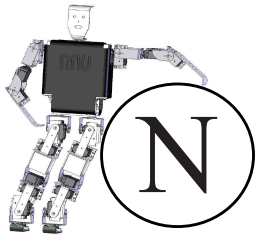
Table L.1: *The joint limitations given in radians.*

Drawing of Complete Robot



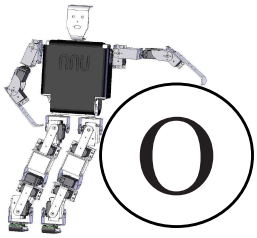


Acronyms



Acronym	Description
ADC	Analogue to Digital Converter
ASM	Assembler
CoM	Centre of Mass
CoP	Centre of Pressure
CPU	Central Processing Unit
CPG	Central Pattern Generator
DC	Direct Current
DoF	Degrees of Freedom
DSP	Double Support Phase
FDSS	Force Distribution Sensing System
FFT	Fast Fourier Transform
FPU	Floating Point Unit
FSR	Force Sensing Resistor
GCC	GNU Compiler Collection
GPIO	General Purpose Input/Output
IMU	Inertia Measurement Unit
LAN	Local Area Network
LiPo	Lithium Polymer
PCB	Printed Circuit Board
PoI	Point of Intersection
PSU	Power Supply Unit
SNR	Signal to Noise Ratio
SSH	Secure SHell
SSP	Single Support Phase
SSP-L	Left Single Support Phase
SSP-R	Right Single Support Phase
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
WLAN	Wireless Local Area Network
ZMP	Zero-Moment Point

CD Contents



Literature Literature, articles and data sheets used in the project. The actual location in the folder can be seen in the Bibliography

Mechanical Design 3D drawings and blueprints from SolidWorks

Electrical Design Drawings, schematics and component lists used for construction of the hardware. Mainly from Orcad

- **Schematics** Hardware schematics

Simulink Simulink files

- **Models** Simulink files for the models
- **Test Trajectories** Simulink files of the test trajectories
- **Controllers** Simulink files of the controllers
- **Interface** Simulink interface for the real system. Including xPC

Maple Maple files

Matlab MATLAB files

- **Models** Files used for modeling
- **Sensor Fusion** Files used for estimation and sensor fusion
- **Trajectories** Files used to find trajectories
- **Measurement Data** Measurement data from the test

Reports The thesis, the report from the study trip and a handout of the project

Videos and Pictures Videos and pictures from the project

Software Software files

- **Drivers** Drivers for the hardware
- **Data logger** Files for logging of data in Simulink