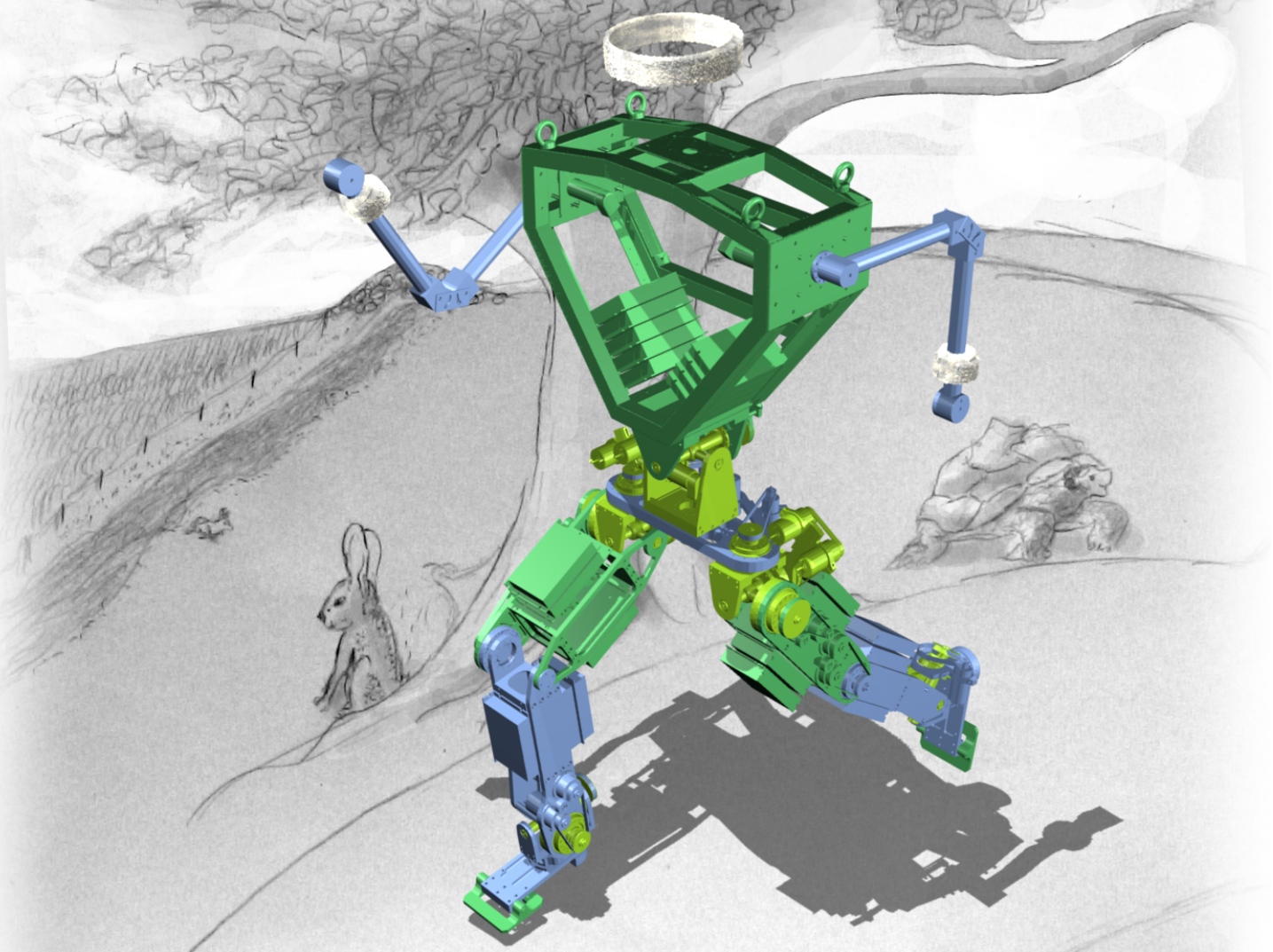
A Step Closer; Towards Dynamic Walking



Group 1036

"To err is human, but to really foul things up you need a computer."

—**Paul Ehrlich**

AALBORG UNIVERSITY
Institute of Electronic Systems
Frederik Bajersvej 7
Telephone 96 35 87 00
Fax 98 15 17 39
http://es.aau.dk/

Title:
One Step Closer; Towards Dynamic Walking on AAU-BOT1

Term:
9th-10th Semester
Intelligent Autonomous Systems
September 1th, 2009- June 2nd, 2010

Project Group:
Intelligent Autonomous Systems 1036

Group Members:
Jes Andersen
Henrik Dalsager Christensen

Supervisors:
Jan Helbo
Jakob Stoustrup
Mads Sølver Svendsen

Copies: 7

Pages: 255

Appendices: 11

Abstract:

The thesis treats the development of a dynamic walk controller for AAU-BOT1. AAU-BOT1 is a biped robot developed at Aalborg University.
A Biped Robot system, can be described as a hybrid system, as long time spans of the gait are governed by continous motions, and the impacts with the ground can be considered discrete events. Planning a trajectory, and applying it on the system must take this into account.
AAU-BOT1 is physically unable to walk statically, this is partly due to kinematics, and weight distribution, and partly due to the combination of small actuators and large frictions in the joints.
It has been suggested numerous times during the development, that a traditional PID regulator can be used as a servo controller, and that all the nonlinear effects, both discrete as well as continuous can be suppressed on the robot. This claim is put to the test in this thesis, as a traditional PID regulator is tuned and applied for the physical system.
The gait is generated on line, based on model prediction, and numeric integration, of a nonlinear inverted 3d pendulum. Applying the trajectory to the robot requires that the implemented PID controller can handle the system sufficiently well, it is however shown that non-linearity's can occur which can destabilize a PID regulator.
Slowing down the trajectory, and keeping the knees stretched counters the problems, and a stable quasi-static gait is achieved.

AALBORG UNIVERSITET
Institut for Electroniske Systemer
Frederik Bajersvej 7
Telephone 96 35 87 00
Fax 98 15 17 39
http://es.aau.dk/

Titel:
Et skridt på vejen, mod dynamisk gang med AAU-BOT1

Periode:
9.-10. Semester
Intelligente Autonome Systemer
1. September - 2. June

Projekt Gruppe:
Intelligent Autonomous Systems 1036

Gruppe Medlemmer:
Jes Andersen
Henrik Dalsager Christensen

Vejledere:
Jan Helbo
Jakob Stoustrup
Mads Sølver Svendsen

Kopier: 7

Sider: 255

Appendiks: 11

Synopsis:

Dette kandidat speciale dokumenterer udviklingen af en controller til at gå dynamisk med AAU-BOT1.
En tobenet robot kan beskrives som et hybrid system, da meget store dele af gangen kan forklares med kontinuerte bevægelses ligninger, og resten, som f.eks. når foden rammer jorden, kan forklares med diskrete opdateringer af states.
AAU-BOT1 er fysisk ude af stand til at opnå statisk gang, delvist pga. begrænsninger i kinematikken og placeringen af vægten på robotten, men også pga. de små motorer og den høje friktion i gear.
Det er tidligere projekter med AAU-BOT1 blevet foreslået at bruge en klassisk PID regulator som servo kontrol, og det er forventet af tidligere grupper at en PID regulator skulle kunne undertrykke de non-lineære effekter i systemet. I dette speciale bliver det forslag testet i praksis, og en PID regulator er brugt som servo kontrol.
Gangen bliver genereret online, på baggrund af en fremskrevet model, og numerisk integration af et nonlineært omvendt 3d pendul. At få robotten til at følge den genererede bevægelse kræver at servo kontrollen kan holde ledene stabile under gang, det er dog vist i praksis at den relativt simple PID regulator ikke kan håndtere alle de nonlineære artefakter, og kan blive ustabil, når f.eks. benene skal strækkes meget hurtigt ud, eller en fod sættes hårdt i jorden.
Hvis gang trajektoriet simplificeres så benene er udstrakte under hele skridtet, og at impacts bliver kørt langsomt, kan quasi statisk gang dog opnåes med en PID regulator som indre løkke.

Acknowledgments

We would like to thank our supervisors, Jan Helbo, Jakob Stoustrup, and Mads Sølver Svendsen, for making the project possible, and for lending out AAU-BOT1 for the duration of the project. Multiple discussions on modelling techniques, and on gaits have provided the insights needed for analyzing the problem.

A special thanks is given to our families, which have supported us through the span of the project, even though they by now must have forgotten what we look like.

Besides of family and spervisors the following people have aided us.

Andy Ruina Theoretical and Applied Mechanics, Mechanical and Aerospace Engineering, Cornell University Ithaca. NY 14850

For providing valuable contacts, and pointers in the early research phase.

Daniel Winther Uhrenholt Craftsman, at mechanical lab at Automation and Intelligent Systems, Aalborg University, Aalborg

For design and fabrication of brackets for the upgraded potentiometers, upgrading the crane frame, and for mounting the IMU, and OBC to the robot.

Mariano Garcia Theoretical and Applied Mechanics, Cornell University Ithaca. NY 14850

For providing MATLAB simulations of McGeers 2d passive dynamics walkers, and for introduction via MATLAB code to the essentials of 3d passive dynamic walking bipeds.

Michael Odgård Kuch Niss Graduated Masters student, Automation and Intelligent Systems, Aalborg University, Aalborg.

For introductions to the existing AAU-BOT1 interfaces, and introduction to the safety features in the laboratory.

Per Kingo Jensen Graduated Masters student, Automation and Intelligent Systems, Aalborg University, Aalborg.

For providing detailed information on the hardware, and sharing experiences with using WeBots, and inspection of CAN interface.

Rasmus Nielsen Undergraduate Student, Electronics and Electrotechnics, Aalborg University, Aalborg.

For operating AAU-BOT1 and video camera during experiments which requires more than two people.

Simon Jensen IT employee, at Automation and Intelligent Systems, Aalborg University, Aalborg

For mounting the potentiometers used for goniometers, improving the PSU choice for the OBC, and reparing damage done to the robot.

Contents

List of Acronyms

AAU Aalborg University

AMPM Angular Momentum Pendulum Model

CAN Controller-Area Network

CMP Centroidal Moment Point

CoG Center of Gravity

CoM Center of Mass

CoP Center of Pressure

DoF Degrees of Freedom

EMG Electromyography

FRI Foot Rotation Indicator

FTS Force Torque Sensor

GCoM Ground projected Center of Mass

GRF Ground Reaction Force

HDG Harmonic Drive Gear

IAS Intelligent Autonomous Systems

IMU Inertia Measurement Unit

iZMP Imaginary Zero Moment Point

FZMP Fictatious Zero Moment Point

LIPM Linear Inverted Pendulum Model

LQG Linear Quadratic Guassian

LQR Linear Quadratic Regulator

LTI Linear Time Invariant

MIMO Multiple Inputs and Multiple Outputs

moCap Motion Capture

MPC Model Predictive Control

MPG Motor Pattern Generator

MPG's Motor Pattern Generators

MPG Motor Pattern Generator

OBC OnBoard Computer

ODE Ordinary Differential Equation

openDE Open Dynamic Engine

PID Proportional, Integral and Differential

PoS Polygon of Stability

PDBR's Passive Dynamics Based Robots

PDBR Passive Dynamics Based Robot

PLL Phase Locked Loop

SISO Single Input and Single Output

UDP User Datagram Protocol

Xode Xml data format for Open Dynamics Engine

ZMP Zero Moment Point

ZRAM Zero Rate of change of Angular Momentum

Definitions

Abduction Moving limps away from the body, and spreading them from their "kinematic siblings" i.e. abducting a leg means to move it outwards in the frontal plane [2].

Adduction Moving limps towards the body, or nearing them to their "kinematic siblings" , i.e. adducting a leg means to move it towards the other leg, if it crosses the center of the body, it is usually named hyper-adduction [2].

Anterior A well defined term for describing the front side, or the forward motions of the human body, and as a consequence, it is also used when describing similar areas in anthromorphic robotics. It is in robotics defined as forward motions, in the sagittal plane [1].

Anthromorphic Is an expression inherited from fiction, here it defines the action of the author when assigning human attributes to animals or objects, in litterature, it is often also spelled **Anthropomorphic**. It is commonly used in robotics to express that a robot behaves somewhat human, in action and/or in appearance[3].

Configuration Space Considers the space a robot can move in, as a space which is multi dimensional and non convex, where each robot DoF contributes with a dimension. The result is a space which defines if a movement of a joint is legal or not, (with relation to collisions with the environment or the robot it self) when all other joint angles are known[37].

Dynamic Walk Is a description of a particular subset of all possible gaits, where the ZMP, CoP, and CoM are allowed to leave the PoS. This corresponds to falling over in at least some time interval, and to allow this it is necessary to later catch the robots balance, if it is to retain a stable walk. Catching the balance can be done either literally, by extending a limb towards the ground and catch the robot, or by moving a lot of mass to force ZMP, CoP and CoM into the PoS [28].

Dorsi Flexion Bending the foot upwards towards the shin, and decrease the relative angle between the foot and the leg. A foot angle less than 90^{o} is said to be dorsi-flexed, [2].

EPOS Is not an abbreviation, but the name that Maxon Precision Motors has given to their series of motor controllers. They are primarily available with position control, which could explain the name chosen, however, they can also control a motor to follow a speed or current reference[23].

Extension Is a rotation of a joint, which "unbends" it, or the relative angle between the links increases towards a reference [2].

Flexion (or Flexing) is a bending movement of a joint, (rotation) in which the relative angle between the links decrease [2].

Gait Is the description of the pattern of motion, of the limbs, during a legged locomotion. i.e. The main four gaits of a horse are walk, trot (jog), canter and gallop[4].

Holonomic See Nonholonomic.

Hyper extension When the extension movements goes beyond the normal range of motion for humans. i.e. A knee which is bended oppositely of the usual direction [2].

Hyper flexion When the flexion movements goes beyond the normal range of motion for humans. i.e. bending the hand, in the direction of the fingers in a fist, towards the arm until the fingers touch the inside of the arm. This is not possible for many humans, but some can hyper flex their wrists [2].

Inferior A well defined motion when describing motions of the human body. It is defined as an downwards motion (negative in z-axis), [1].

Lateral A well defined motion when describing motions of the human body. It is defined as motion away from the center in the frontal plane, [1].

Medial A well defined motion when describing motions of the human body. It is defined as motion towards the center in the frontal plane, [1].

Nonholonomic Also known as an **anholonomic** system. In robotics a system is said to be nonholonomic if the controllable DoF are less than the total DoF. An automobile is an example of a non-holonomic vehicle. The vehicle has three DoF, (its position in two axes, and its orientation relative to a fixed heading). Yet it has only two controllable DoF (rotation of the wheels and the angle of the steering wheel), with which to control its position and orientation. Thus, not every path in phase space is achievable; however, every path can be approximated by a holonomic path, this is called a (dense) homotopy principle. The non-holonomicity of a car makes parallel parking and turning in the road difficult, but the homotopy principle says that these are always possible, assuming that clearance exists[26].

Plantar Flexion Is the movement which the bottom of the foot moves down, and and the angle formed between the shin and the foot increases. It relates to extension in motion, except that the foots reference position is not close to straight continuation of it's kinematic parent. If the ankle has a relative ankle larger than 90^{o} it is plantar flexed [2].

Posterior A well defined motion when describing motions of the human body. It is defined as motion backwards in the sagittal plane [1].

Polygon of Stability Is also often referred to as Support Area, in relating literature. It is the convex hull of the ground support points. As an example consider a tripod stool standing on a floor, then the PoS would be the triangle spanned on the floor by the three point feet[39].

Redundancy Usually encountered in other fields of research as the implementation of on-line backup systems, to avoid a system failure when a component fails. In Robotics, redundancy considers the controllable DoF in relation to the work space DoF. The idea is that if a point in the work space can be reached in at least two combinations of joint angles, the system has redundancy[26].

Static Walk Is a description of a particular subset of all possible gaits, where the ZMP, CoP, and CoM are kept inside the PoS at all times. This ensures that the robot is at all times in balance, and has been applied on many biped robots[39].

Step The basic cycle within a gait. For all gaits which repeats it self in a pattern, the gait can be cut into steps, which represent a complete cycle. For human gait, a step is often representing the motions that collectively allows the feet to be moved to mirrored locations of each others starting point[16].

Superior A well defined motion when describing motions of the human body. It is defined as an upwards motion (positive in z-axis), [1].

Work Space or Task space, is the space in which the robot operates. It is bounded by the objects which the robot must avoid, and is often expressed in 6 dimensions, (x, y, z, yaw, pitch, roll)[26].

1

Prologue

"In the fifties, it was predicted that in 5 years robots would be everywhere.
In the sixties, it was predicted that in 10 years robots would be everywhere.
In the seventies, it was predicted that in 20 years robots would be everywhere.
In the eighties, it was predicted that in 40 years robots would be everywhere..."
—Marvin Minsky

This thesis has been composed during two semesters from September 1st, 2009, to June 3rd 2010 at the Department of Electronic Systems at the Section for Automation and Control, under the Master program Intelligent Autonomous Systems (IAS).

Previously created documentation, and future AAU-BOT developments can be followed at http://www.aaubot.aau.dk, where this thesis is also available as a pdf file.

In this report the ongoing development of a control system for the AAU-BOT1 humanoid robot is documented. In this part of the development, the goal set since the beginning of the project in 2006 of achieving dynamic biped walk in 2010 is considered the primary target.

A controller for AAU-BOT1 is by definition a system that enables the robot to track a reference trajectory. This implies that errors in the output when compared to the reference signal must be suppressed[21]. Typically when a controller is applied to any system, it is applied when the passive system dynamics does not stabilize the output at the reference value by default. As an example consider a segway, see Figure 1.1 for an illustration of a segway, which without a controller would fall flat on the ground instead of moving forward at a stable velocity.

Actuators on mechanical systems are often selected such that they are able to counter the passive system dynamics, and thus they can be used to force a system to follow a reference which is far away from the naturally occurring motions. For the segway this statement can be interpreted as the ability to stop it by suddenly running the wheels much faster than is needed to sustain the velocity. Large actuators allows for a very short stop length, and small actuators require more time to stop the segway. It is obvious that when the actuator is chosen very close to the requirement for maintaining the velocity it might not even be possible to stop the segway at all.

On AAU-BOT1 the actuation system has been designed to be adequately strong to recreate

Figure 1.1: A picture of the X2 Segway, this is the commercial version of a classic control problem of balancing an inverted pendulum on a cart.

the human gait in simulations, but only a small torque margin is left to suppress any errors in the original dynamics model. Some of the actuators are even chosen so that they must be overloaded in a short amount of time[27]. Due to the choice of actuators, trajectories which is very different from human walking can pose problems, and it can be required that the actuators has to do more work than was suspected in the design phase just to counter the naturally occurring passive dynamics. Hence a trajectory is needed for AAU-BOT1 which is either anthromorphic as it was designed for, or if this is not possible, it must minimize the torques needed to track it, such that the torque margin available to suppress tracking errors and disturbances is maximized.

For the trajectory synthesis, the advances done in Passive Dynamics Based Robots (PDBR's) within the last decade is investigated, as it has produced highly efficient, and human-like walking with bipedal robots [6]. Interestingly PDBR's in general are designed to take advantage of the system dynamics to achieve walking, and can usually walk down a slope with no actuation at all. The advancements achieved in this field of research are almost only a product of the refined mechanic designs, and thus as this type of robot achieves dynamic walking on level ground, it does so with very simple control strategies like on/off state machines.[7].

The PDBR's investigated in the research for this thesis are build for high efficiency and achieves anthromorphic walk using simple models, which is promising for on-line generation of a reference trajectory. The Passive Dynamics Based Robot (PDBR) designs studied has

not produced robots that could be programmed to do anything else than walking, hence it is interesting to duplicate their behavior with an active robot, that can. When investigating the PDBR's currently produced, it is revealed that most of the PDBR's are unable to stand still. This is both due to their foot design and due to the simple control strategies applied.

AAU-BOT1 is mechanically not built as the PDBR's. Instead AAU-BOT1 has as a very flexible design with regards to control and motions possible, in theory it can simulate a multitude of gaits, and should be able to do alot more than just walking, i.e. stand still and initiate walking, or sitting down[1], or jumping.

This thesis however focuses on the development of a dynamic walk controller for AAU-BOT1, which harnesses the ideas used for the PDBR's for a trajectory synthesis for AAU-BOT1.

Previous work done on the modelling, instrumentation and simulator implementation is vital to the development of a controller which can achieve dynamic walk. This is mainly due to limited funding, which makes it difficult to repair any damage that the robot might experience, and thus makes testing difficult.

During the analysis phase to this thesis, it was questioned by this group whether the existing simulation model was sufficiently well representing the double stance phases. The main reason for this assessment was that the existing simulator was based on a linear combination of two nonlinear models, which was not a modelling technique familiar to this group.

To investigate if the simulation model was representing the robots dynamics appropriately, the documentation and implementation of it was analyzed in greater detail, and the analysis revealed shortcomings in the existing simulation system with regards to parameter estimation, and verification of the model. Hence it was decided at an early stage to work on verifying the model.

It was eventually decided to implement an alternative simulator model based on a different modelling principle, as it could be used to make the existing simulation plausible, and provide better insight into the dynamics of the system. Also the trajectory generation system and the controller structures developed for retaining balance during static walk is not expected by this group to be suitable for dynamic walking. Hence it will also be re-designed in this thesis.

[1] Sitting down is no longer possible with AAU-BOT1 due to constraints from wiring and EPOS placement.

1.1 Thesis Outline

This section is provided to give an overview of the structure of the thesis.

The thesis is built up in seven major chapters which treats the documentation differently. The Prologue concerns background knowledge, and introduces different ways of achieving bipedal walk which has been researched. The Conceptualization concerns the considerations that was done in general before the system was designed, and also a treatment of the expected challenges in transferring the concept to AAU-BOT1. The Modelling provides an overview of the developed models, and the Controller Design concerns the actual design of the controller. The Implementation presents the implemented systems, and the performance of the implementations is shown in Results. Finally the report ends in an Epilogue which concludes the thesis, and provides final thoughts on the work.

Prologue The thesis is initiated by a prologue. This provides the reader with background information on the AAU-BOT project, and introduces much of the terminology used throughout the thesis.

- ▷ **Thesis Outline** Overview of the content in biblical order.
- ▷ **History of the AAU-BOT1 Project** Provides a short introduction to the evolution of the project of creating a walking biped robot.
- ▷ **Anatomy** Defines the geometry and naming references which is used in the thesis. The definitions are to a large extend compatible with previous AAU-BOT1 documentation.
- ▷ **Balancing Points** Introduces the balance points often used in robotics.
- ▷ **Introduction to Human Walking** Introduces the terminology used when describing human walking, and treats some of the dynamic properties of the human approach to walking.
- ▷ **Bipedal Walking Robots** Introduces other robots that have already achieved bipedal walking, and introduces some of the terminology used to describe the gaits of a bipedal walker.
- ▷ **Analysis of AAU-BOT1 Walking Attempts** Provides an overview of the previous attempts to make the robot walk, and analyzes why the robot has not yet succesfully been walking.
- ▷ **Summary** sums up the knowledge gained from the preliminary research.

Conceptualization The problem is investigated with respect to possible solutions, and the problems which is expected to cause challenges are explained. Some of the tools needed is also defined here as they need to be build from scratch.

- ▷ **Project Outline** To solve tha challenge a path to the solution is determined. The path and the system design is presented in this chapter.
- ▷ **Posture Controller Concepts** Possible choices for a posture controller is listed, and the advantages and challenges for each type is explained.
- ▷ **Introduction to Passive Walkers** introduces the necessary dynamics of the passive dynamic biped walkers, as they are the inspiration for the solution.
- ▷ **Trajectory Generators** Lists and explains the gait generation schemes considered for this thesis.

- ▷ **Trajectory Challenges** Explores the situations where a pure passive dynamics model is insufficient or inappropriate for synthesizing a gait trajectory for AAU-BOT1.
- ▷ **Simulator Model** Describes the concepts that was considered for a simulation model.
- ▷ **Palantir, 3D Visualization Server** Discusses the necessesity of a visualizer, and what it should contain.
- ▷ **Concept Summary** Sums up the chapter, and provides discissions.

Modelling The modelling chapter contains some of the models which is developed for this thesis.

- ▷ **Rigid Body Model for the Simulator** The existing simulator has not bee accepted, and a new is developed. The new modelling principle is based on Rigid Body Dynamics, so this concept is introduced.
- ▷ **Gears and Friction Models** Some of the existing models are compared, and a simple model is chosen.
- ▷ **DC Motor models** The motor locations are presented and modelled for simulation purposes.
- ▷ **Force Torque Sensor Models** The FTS are modelled as linear sensors, and this model is presented

Controller Design The controller developed is presented, it consists of two major parts, being a continous PID regulator, and a discrete state space driven trajectory generator, dubbed a Motor Pattern Generator.

- ▷ **Limit Cycle of Hybrid System** A simple example with a planar walker is setup to illustrate that the problem can be described as a hybrid system, however this method quickly becomes complex to analyze and control.
- ▷ **Posture Controller Design** The method used for PID controller implementation is presented
- ▷ **Motor Pattern Generator** The method used for online trajectory generation is presented, it is a combination of model prediction, and numerical integration, depending on the state of the robot.

Implementation The implementation of the elements developed in the project is presented.

- ▷ **Estimation of CoP** This section describes the implemented transformation from FTS measurements to the CoP location.
- ▷ **Determination of Step Length** A function is developed which estimates the step length.
- ▷ **State Machine** The state machine that governs the discrete changes in behaviour is presented
- ▷ **Simulator Implementation** The simulator implemented in simulink is presented.
- ▷ **Visualization Implementation** Considers the structure of the network based visualizer.

Results The implemented system is tested on the physical AAU-BOT1, and the results are presented.

- ▷ **Verification of the Free Move PID Regulator** The implemented PID regulator is tested in continous operations, to test for stabillity and performance.
- ▷ **Pitch Verification Test of PID Regulator** The PID regulator is tested with dynamically changing loads, to see if it can suppress the dynamics of the system.
- ▷ **Walk Test** A quasi static trajectory is precalculated and applied to the posture controller, to test for stabillity issues at transition surfaces.
- ▷ **Test of Inverse Kinematics** The kinematics are put in the control loop, as they form a crucial part of the trajectory generation scheme
- ▷ **Walk Test with Dynamically Reference** The full system is tested, and it is shown that a fast trajectory can cause problems for the chosen solution.

Epilogue A summary of the thesis, and the conclusion of the thesis is presented.

- ▷ **Conclusion** The intentions of the master thesis is evaluated and interpreted, and a conclusion on the findings is provided.
- ▷ **Future Work** As the next group will continue developement, some areas of the current setup is not functioning optimally, and it is suggested here what to change first.
- ▷ **Proposals for AAU-BOT2** Experience from working with implementation on AAU-BOT1 is condensed to recommendations for the next generation of a biped robot on AAU.

Appendices This part contains all the appendices which further elaborate on some of the subjects treated in the thesis. Appendices for this thesis can in general be read as independent documents, and thus is equipped with short introductions. Still the references are made both between appendices and to and from the main report.

- ▷ **SimMechanics** This appendix treats an example implementation to aid in understanding the simulator implementation.
- ▷ **Derivation of the Passive Walker Dynamics** The model used as reference for the passive walker model used for the development of the state machine controller is derived.
- ▷ **Muscle EMG Patterns During Walking, a Pedagogical Interpretation** Provides insight into the human anatomy and the health science data used for some of the conceptualization done in this thesis.
- ▷ **Impact at Heel Strike** Concerns the complications arising when the impact should be modelled in a simulation environment.
- ▷ **Journals** is actually separate appendices, but the content is quite similar in form, as they describe the experiments done to verify or parameter estimate the models.
- ▷ **Balance Schemes** represents the work done to create a blance controller. Multiple reasons exists why this is not implemented.
- ▷ **Control of a 2d Straight Legged Biped** Very simple control strategy for a 2d walker without knees is illustrated
- ▷ **Palantir Protocol** Describes the protocol implemented for the developed tcp/ip based 3D visualizing tool “Palantir”

- ▷ **Bug Reports** During developments some bugs have occurred, and some of them have been resolved, to ease further development and void surprises, the known bugs are published in this appendix.
- ▷ **Users Manual for AAU-BOT1 Laboratory** Concerns how to interface with the AAU-BOT1 laboratory, and which safety measures that should be taken to avoid damaging the robot and the personnel. Useful for other students or researchers that might work on the robot in the future.

1.2 History of the AAU-BOT Project

The following chapter is intended to provide the reader with background information concerning the AAU-BOT project.

The AAU-BOT project was initiated in 2006 by Professor Jakob Stoustrup, of the Section for Automation and Control, who received a research grant from the Dannin Foundation, for construction of a limping biped robot.

From the viewpoint of Aalborg University (AAU) and the Dannin Foundation, the overall aim of the AAU-BOT project is to close the gab between health sciences and robotics and increase collaboration in these fields. The original goal of the limping biped robot project, was to achieve dynamic and human like walking in 2010, and then afterward apply the limping, and other dysfunctionalities [31].

1.2.1 Designing AAU-BOT1

In 2006 the Masters students Mikkel Melters Pedersen, Allan Agerbo Nielsen and Lars Fuglsang Christiansen, was given the task to design the mechanical structure of the biped, and to choose the appropriate actuators and sensors [27]. They based their kinematic design on Motion Capture (moCap) data of a student walking, designed and manufactured the parts for AAU-BOT1 to be able to handle the torques and forces which would be required to match the gait, (and a few other test cases).

Using the moCap data, or a Linear Inverted Pendulum Model (LIPM) as reference trajectory, they ran simulations with simple Proportional, Integral and Differential (PID) controllers for each kinematic joint. It was concluded that in simulations the robot had enough power to handle the loads during dynamic walk, and it could walk dynamically for at least 4 steps even without balance control.

1.2.2 Instrumentation of AAU-BOT1

In 2007 the Masters students Per Kingo Jensen, Mathias Garbus and Jan Vestergaard Knudsen, was handed over the manufactured parts, and the documentation from the design project [31]. They assembled the robot, and redesigned the instrumentation, afterward they chose the OnBoard Computer (OBC) and installed the internal Controller-Area Network (CAN) networks for communicating with the EPOS's on the AAU-BOT1. Finally they achieved statically stable walking in simulation.

The original simulator was however replaced entirely with a novel approach to biped modelling, rejecting the rigid body approach used by the mechanics group the year before. The reason for this choice was the fact that the original simulator model could not be linearized, and that it was too slow for a controller.

They concluded that with PID control of the actuators and balance control, static walk should be possible with AAU-BOT1, however they also concluded that the kinematic limitations of the

robots ankles would make it nearly impossible in practice if the ankles where not redesigned. Fortunately the robot was never meant to walk statically, so the group did not recommend changes, instead they suggested that focus should be laid on dynamic walking in future projects.

1.2.3 First Attempt on Walking With AAU-BOT1

In 2008 the Masters students Michael Odgård Kuch Niss, and Brian Thorarins Jensen enhanced the dynamic models developed the previous year, especially concerning foot to ground contact, to allow for better simulation and more sophisticated controller designs.

The OBC was also prepared for better controller performance[30], by providing a more direct link with the CAN, and a real time kernel. It was attempted to achieve static walk on the physical robot, however, due to malfunctioning angular sensors and malfunctioning Force Torque Sensor (FTS) in the left ankle, the controller never reached a fully functional implementation, and thus it was never debugged properly.

The group concluded that the robots actuators should be controlled from software, as the built-in PID regulators in the EPOS performed poorly at double actuated joints. Posture controllers based on feedback linearization for setting the torque at each joint was developed, and successful implemented in simulation. However posture control was not achieved on the physical AAU-BOT1, and thus walking was not attempted.

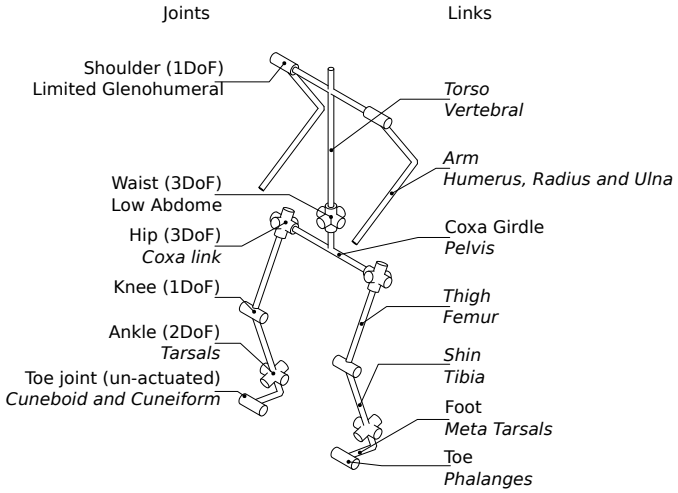


Figure 1.2: Joint and Link names, The stylized drawing illustrates the orientation of the joint axes, and the available DoF which is reduced in relation to humans. However the names are chosen such that they relate closely to the human anatomy.

1.3 Anatomy of AAU-BOT1

AAU-BOT1 is by design chosen to have a geometry that resembles human dimensions. Hence each joint and link can be referred to with human-like name conventions. This is illustrated in Figure 1.2. Also in this figure, the less familiar medical names for the links and joints are defined, as they also occur in this thesis. The reason for this introduction of medical terms, is that the research concerning human anatomy and human walk is often done by physicians, which often uses the medical terms. Names are collected from Grays Anatomy [15, 16].

AAU-BOT1 does not have linear actuators like muscles or pistons, had the robot been equipped with such devices they would have been dubbed in a similar human like fashion. Instead of linear actuators, each joint (except the toes) is actuated by DC motors from Maxon. Each DC motor is powered by a digitally controlled motor control unit, with a CAN interface, the control units are from Maxon and is referred to as an EPOS, more about the EPOS and DC motors can be read in Section 3.3.

A belt connects the motor shaft to a Harmonic Drive Gear (HDG), which transfers the power from the belt to the joint hinge, the joint hinge angles are measured on some of the joints by potentiometers, and the motor shaft angles are measured by build-in tachometers, see Section 3.2 for more details on the power transfer, and Section 3.4 for more details on the sensors on the robot.

In Figure 1.3 the names of commonly used planes and axes are defined. The location of the origin in Figure 1.3 could indicate that the world coordinates origin are placed in the waist, this is not the case for this thesis, instead it is chosen that $z_0$is in fact placed exactly below the feet, when the robot is standing on the ground in zero position. In practice multiple coordinate systems are used in this thesis, and they will be defined when the calculations done in the coordinate system is described, if it is deemed necessesary.

It can be noted that the existing Solidworks model of the robot has origo located just as in Figure 1.3, this is also the case for the developed simulation model before it is initialized. For the simulation model it should also be noted that it is created in a left handed coordinate

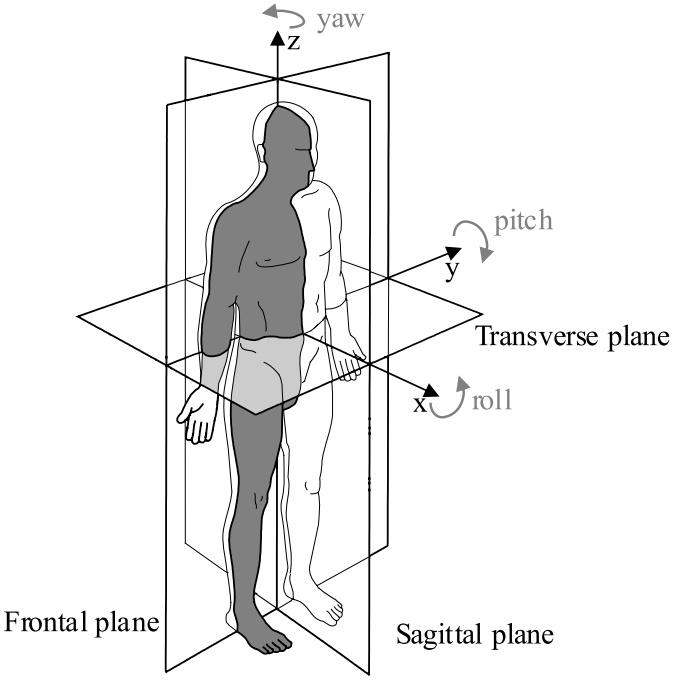


Figure 1.3: The Global axis, and planes with names and orientation of rotations, used throughout this thesis, the figure is inherited from [30].

system with the y axis upwards. However, all interfaces with the simulation have orientation and positions defined as the world coordinates used in the report.

1.4 Balance Points

The task of keeping a biped in balance has been researched for a while. For control engineers, it is often preferred to have a simple trajectory or stability point that can be tracked, as this provides a criterion and a reference for a stabilizing controller.

Many such points has been defined and proposed as a stabilizing reference. They are introduced in this section.

1.4.1 Center of Mass (CoM)

The CoM, is the point in a rigid system of particles at which the system's whole mass can be considered to be concentrated for the purpose of calculations. As rigid bodies defines a volume of distributed particles which are rigidly connected to each other, the CoM can also be found for a system of rigid bodies, which is used as a concept for modelling mechanical systems, see Section 3.1 for more on this modelling principle.

In the special case where all the elements of a body is rigid, the CoM can be found as the average of the position of the particles, weighted by the masses of them [17, 33], see Equation 1.1.

$$CoM = \frac{\sum_i m_i \vec{r_i}}{\sum_i m_i} \tag{1.1}$$

If an object has uniform density then its center of mass is the same as the centroid of its shape.

The CoM of the entire robot can be found in a similar matter, and naturally it is depending on the current position of the joints, as it moves the masses around in relation to each other. However much in the same manner as if the robot was a statue, the CoM behavior it is independent of dynamic effects. See Figure 1.4 for an illustration of the location of the CoM.

1.4.2 Ground projected Center of Mass (GCoM)

The GCoMis the CoM projected on the ground [22]. Hence it is also focusing on the static properties of the robot.

$$GCoM_x = \frac{\sum_i m_i \vec{x_i}}{\sum_i m_i} \tag{1.2}$$

$$GCoM_y = \frac{\sum_i m_i \vec{y_i}}{\sum_i m_i} \tag{1.3}$$

$$GCoM = \begin{bmatrix} CoM_x \\ CoM_y \\ 0 \end{bmatrix}$$

The point is for very slow movements a great indicator for balance. In fact if it is within the Polygon of Stability (PoS) a statue would be statically stable, and never fall. A robot moving very slowly has very small dynamic effects and can thus be considered a statue for all practical purposes.

Alternatively the GCoM can be found as the point fulfilling the following equation.

$$\sum_i ((GCoM - \vec{r_i}) \times m_i \cdot \vec{g}) = 0$$

g is the gravitational acceleration vector: [0 0 -9:82]T [m/s2]

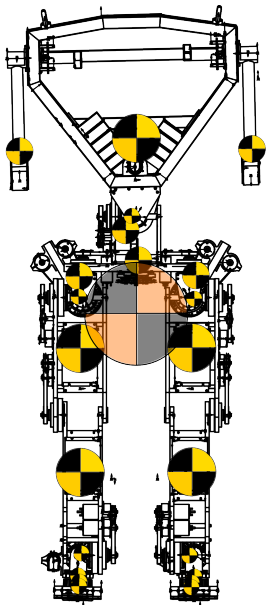


Figure 1.4: Each body part have a CoM, the weighed average of them is the robot CoM. Notice that it might not be located within the geometry, but is always within the mass.

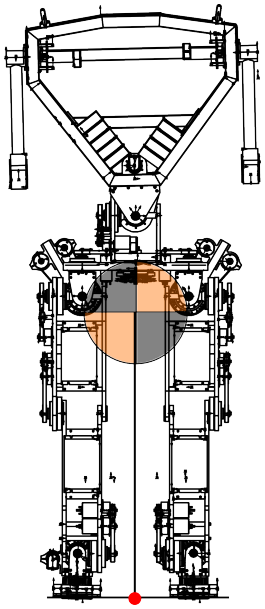


Figure 1.5: The ground projection projection of the CoM, is called GCoM and indicates if the robot is in balance if the robot is static.

1.4.3 Center of Pressure (CoP)

The CoP is a point on the ground that only have meaning when the robot has ground contact [22]. It sums and weights all the forces exerted between the robot surfaces (that has ground contact), and the ground, to create a single point in which, a single linear force, and a single torque, can produce the same motions of the robot, as the entire ground does.

It is a point which includes the dynamics of the robot, as the accelerations of the robot, will produce a change in the distribution of the forces between the robot surface and the ground.

To calculate the CoP it is required that knowledge is obtained about the forces between the robot and the ground at multiple points on the surface, then for each point the ground normal force F can be used to determine the CoP:

$$CoP_x = \frac{\sum_i F_i \vec{x}_i}{\sum_i F_i}$$

$$CoP_y = \frac{\sum_i F_i \vec{y}_i}{\sum_i F_i}$$

Notice that in practice the CoP can be difficult to obtain for complex surfaces, and generally requires many small force sensors on the surface for a proper determination. CoP position, (x, y), is very sensitive to errors in the moment & force components for a small Fz. The CoP can also be very challenging to calculate, as it requires a model which distributes force over the contact surfaces.

If CoP is within the PoS at all times a robot in motion will be statically stable. This feature makes CoP control much more flexible than just using the GCoM for reference, and allows for much quicker reactions. The CoP requires a ground contact to even exist, and it follows from the definition that a CoP that lies on the rim of the PoS indicates that the robot is actually not in balance, and is falling.

The behavior of the CoP can take a while to understand fully. As an example consider Figure 1.6, here the motion of the CoP is animated for a system of 2 rigid bodies. Notice that it first moves away from the masses, due to the acceleration of the upper body, which reacts on the lower body. When the motion has stopped the CoP moves as far towards the GCoM as possible.

1.4.4 Zero Moment Point (ZMP)

While literature on biped walking often states that CoP and ZMP are identical points [22], this is actually only true while the robot retains balance [19, p. 734 - 735]. Still this is a favorable feature as the ZMP is easy to compute, while the CoP is easy to measure.

There exist several definition's of the ZMP, which are all valid. Many of these are presented in [22] and here two of them are presented.

The first one presented is more conceptual, and aids in understanding the ZMP, it states that the ZMP is the point on the ground where the total moment generated due to gravity and inertia equals zero as shown below.

$$\sum M_x = \sum M_y = 0$$

where:

Mx and My are the torques around the x-axis and y-axis respectively [Nm]

The second definition presented, is to explain how the ZMP is computed in practice. The derivation of the calculation presented here assumes flat ground with no inclination, and is found in [8].

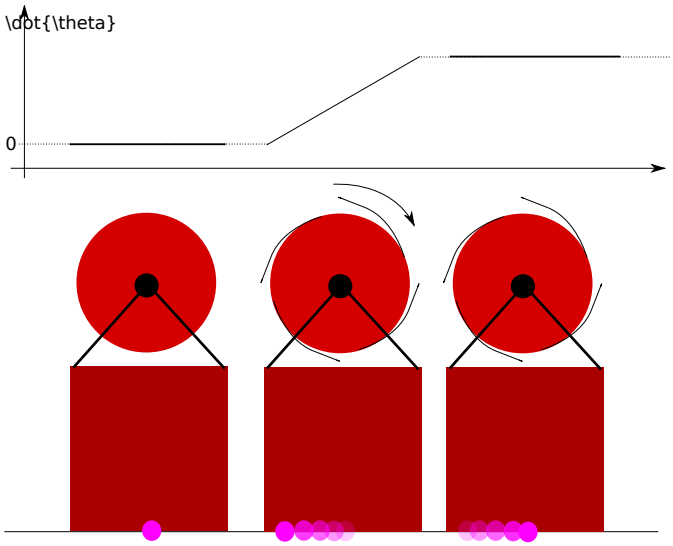


Figure 1.6: The location of the CoP is often at the GCoM, but can divert when accelerations occur on the system. In this illustration the CoP starts by coinciding with the GCoM, and then due to acceleration of the flywheel is forced towards the edge of the PoS. Fortunately it does not land on the edge, as the box would then have turned over. When the flywheel is no longer accelerated, it maintains the angular velocity, and the CoP moves back to the CoM.

$$ZMP_x = \frac{\sum_i [m_i\{x_i(\ddot{z}_i + g) - \ddot{x}_i z_i\} - I_{i,y}\omega_{i,y}]}{\sum_i \{m_i(\ddot{z}_i + g)\}} \tag{1.4}$$

$$ZMP_y = \frac{\sum_i [m_i\{y_i(\ddot{z}_i + g) - \ddot{y}_i z_i\} - I_{i,x}\omega_{i,x}]}{\sum_i \{m_i(\ddot{z}_i + g)\}} \tag{1.5}$$

1.4.5 Imaginary Zero Moment Point (iZMP)

For static gaits where the robot is at all times in balance, the ZMP and CoP will always coincide. For a dynamic gait, the ZMP should often leave the PoS and thus be separated from the CoP. The difference between the two helps in indicating which direction the robot is falling, and about which point on the ground contact surface. In this case the ZMP is often referred to as iZMP, Foot Rotation Indicator (FRI)or Fictatious Zero Moment Point (FZMP).

Some literature and scientists does not distinguish between iZMP and ZMP as they both coincide with CoP when balanced, others explicitly states that the ZMP only has meaning when it is inside the PoS and thus it always coincides with CoP, hence the iZMP is a different point, and not a true (regular) ZMP [14, 35].

The effect of the iZMP in relation to an object is illustrated in Figure 1.7, and in short it can be stated that when the iZMP is not within the PoS it means that an unbalanced moment has occurred which cannot be compensated for by foot reaction forces.

The iZMP is calculated by assuming that the sole of the foot extends the entire ground. This has no physical justification, and the point is often considered a mathematical point. However it remains powerful in diagnostics of the balance when falling, and thus when walking dynamically.

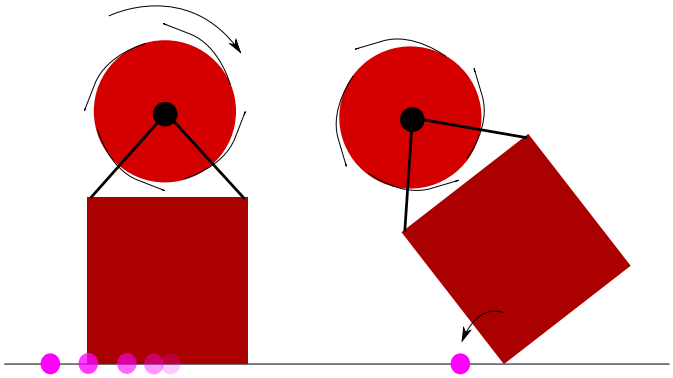


Figure 1.7: The iZMP is located where the CoP would be if it could leave the PoS. Hence if they do not coincide it marks that the object having ground contact is falling. It also hints the direction of the fall. In the illustration the flywheel is spun up too fast, creating torques that overthrows balance.

1.4.6 Centroidal Moment Point (CMP)

Also denoted in some literature Zero Rate of change of Angular Momentum (ZRAM) [22]. The CMP is defined as the point where a line parallel to the ground reaction force, passing through the CoM,intersects with the external contact surface.

The CMP is thus the point where the Ground Reaction Force (GRF) would have to act to keep the horizontal component of the whole-body angular momentum constant. When the moment about the CoM is zero, the CMP coincides with the CoP. However, when the CoM moment is non-zero, the extent of separation between the CMP and CoP is equal to the magnitude of the horizontal component of moment about the CoM, divided by the normal component of the GRF.

Or more simply, it can be stated that when the CMP corresponds with the ZMP, the GRF passes directly through the CoM of the body, satisfying a zero moment or rotational equilibrium condition. Hence, the departure of the CMP from the ZMP is an indication of non-zero CoM body moments, causing variations in whole-body, spin angular momentum.

In some literature the CMP is dubbed "the effective CoP", as it is intuitively what it illustrates. The Figure 1.8 illustrates the CMP movement of a system which is turning over due to some previous actuation.

$$CMP_x = CoM_x - \frac{F_{G.R.x}}{F_{G.R.z}} CoM_z \tag{1.6}$$

$$CMP_y = CoM_y - \frac{F_{G.R.y}}{F_{G.R.z}} CoM_z \tag{1.7}$$

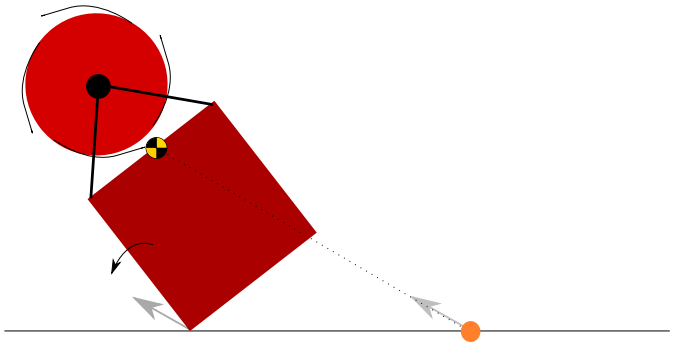


Figure 1.8: The CMP indicates the relation between the GRF, and the CoM. Here the box is tilting creating ground reaction forces that prevents it from sliding. Hence the CMP moves away from the CoP, which indicates that the system is out of balance.

1.5 Introduction to Human Walking

Describing the human gait can be done in a multitude of ways. In robotics, and also in the AAU-BOT project, the focus has often been laid on describing the different phases of foot to ground contact. As this method for describing the human walk, provides insight into the sequence of foot motions necessary, it is also presented in this section. However, as this approach shows little about how the torso and the legs move to achieve the gait, the movements above the ankle is presented after the phases of the foot is explained.

After the traditionally used gait phases are presented, the section continues into a presentation of some of the properties of the human body to provide the reader with a basic understanding of how humans achieve the actuation necessary to walk.

1.5.1 Foot Movement Phases

Traditionally the human gait cycle is divided into eight sequences[10]. The names are somewhat self-descriptive as they are based on the movement on a single foot, an illustration of this can be seen in Figure 1.9. The sequences are by definition a sequence in a loop, but usually a step is divided in two phases (the stance phase, and the swing phase) and in the following order:

Stance Phase

1. **Heel Strike** initiates the gait cycle, it is the moment in time where the heel of the front foot impacts with the ground, and it also represents the point at which the body's center of gravity is at it's lowest position.

2. **Foot-flat** occurs when the leg rolls forward on the foot and the plantar surface of the foot touches the ground.

3. **Midstance** happens when the other foot swings by, the foot in focus is still flat on the ground.

4. **Heel-off** is a result of the upper body now being in front of the rear foot. Ground contact is maintained by lifting the heel and stand on the toes, in this phase the push-off is also initiated via the gastronomicus muscles, which plantar flex the foot. This indicates that

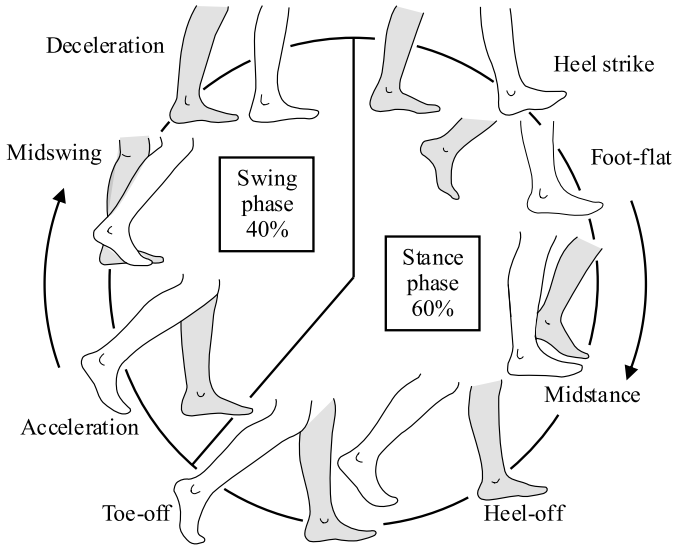


Figure 1.9: The human walk cycle can be divided into a swing phase and a stance phase when described by foot movement phases. This figure is seen from the perspective of the right foot.

a force is provided to keep the angle and the toes in their designated angles. In many recorded gaits the other foot has just experienced heel strike when heel off is initiated, this is however different for each individual, and can sometimes happen during the push-off, or immediately after.

5. **Toe-off** occurs when the foot finally leaves the ground.

Swing phase

1. **Acceleration** begins at the exact moment in time when the foot is lifted from the ground, and indicates that the foot accelerates forward.

2. **Midswing** occurs when the foot passes directly beneath the body, coincidental with midstance for the other foot.

3. **Deceleration** is the last motion in the swing phase, the swinging is slowed down and the knee is extended, and the foot is stopped just in time to have the proper stride length when heel strike occurs.

As a result of this division in phases, where only one foot is considered in the gait, it could be speculated that the two feet and legs can be moved independent of each other, and if individually controlled to stay in the sequence, then achieving this gait is just a matter of phase shifting the gait cycle on the two feet.

However this is not all together true, in fact, when heel strike occurs, the other foot is always entering, or already in the heel-off phase, ready to immediately enter the toe-off sequence afterward, even when changing frequency, and direction (where the stride length is different on the two feet).

The gait is often described using a time division of the shape of the PoS. The shape of the PoS over time relates off course directly to the phases where the feet has ground contact, and

is often referred to as the support phases of the gait. In the following the support phases are defined, and it is chosen to describe them in a sequence which relates to the stance and swing phases of the left foot.

1.5.2 Support Phases

1. **Double Support Phase - Heel Strike and Heel off** when the heel strike occurs of the left foot, the right foot is just entering or still in heel-off, this creates a support polygon which spans both feet, (the toes on the right foot, and the heel on the left, and the area spanned between them)
2. **Single Support Phase - Left Heel** the right foot no longer has ground contact, and the left foot carries the entire robot on the heel. The support polygon only consists of the area where the heel touches the ground, the left foot is rotated downwards towards foot-flat.
3. **Basic Single Support Phase** the right foot is swinging forward, and the left foot is kept flat on the ground. Hence the support polygon spans the convex hull of the left foot.
4. **Single Support Phase - Heel-off** the right foot has been swung so far forward that heel strike will soon occur, in this phase the support polygon only spans the toes on the left foot, this phase does not always occur, especially short stride lengths can be achieved without this phase.
5. **Double Support Phase - Heel Strike Heel off (repeated)** The cycle repeats, but left and right is switched.

Other support phases are also possible, but they are not necessary for achieving regular repetitive human walking, for a graphical interpretation of the support phases see Figure 1.10. The other support phases usually encountered are:

1. **Basic Double Support Phase** both feet are flat on the ground. This happens when standing still, and is not a part of the dynamic walking pattern for humans, but ar often encountered in static walking.
2. **Double Support - Flat foot and Heel Strike** is entered if the heel strike occurs before the stance foot has entered heel-off, this can happen when initializing or ending a walk, or when shortstepping. As initialization of walking is possible in a multitude of ways, it can also be avoided if chosen to. The gait is by definition not stable at initialization, and it can be discussed if initialization is even part of the human walking pattern, even though it off course is necessary to have a initialization sequence to be able to start walking.

During the phases of the feet, and the relating support phases, the keen reader will have observed that the description so far has been strictly two dimensional. A human is naturally moving in three dimensional space, and the geometry of the feet placements can vary in both stride length and step width, even the angle of the feet can be varied without violating the described phases.

While the names of the foot placement geometry parameters are quite self-explanatory, the locations are to avoid possible confusions illustrated in Figure 1.11. There is also a step height, (or ground clearance) parameter when describing the gait, this is not drawn in the figures. It denotes the distance from the lowest part of the swinging foot, to the ground.

It is often considered satisfying to understand the phases and the geometry of the feet placements, as the phases indirectly dictate the relating movements of the legs and hip. However, human walking is not only determined by the way the feet interact with the ground.

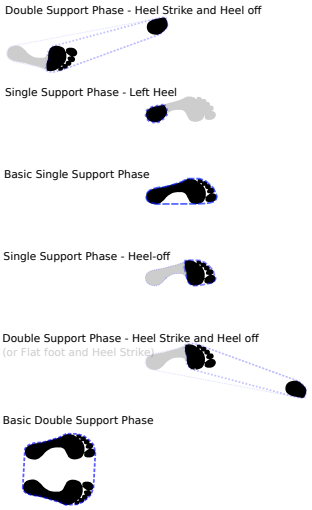


Figure 1.10: Support Phases, illustrated from the Transverse plane. The dotted lines marks the PoS.

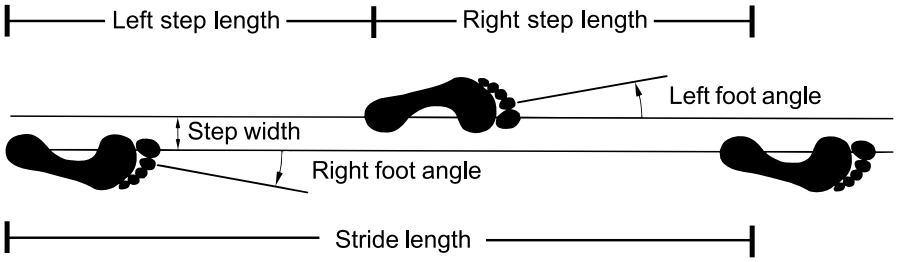


Figure 1.11: Step parameters in the transverse plane, Figure is borrowed from [10].

1.5.3 Legs and Torso Movements

Almost all the parts of the body are in motion during the walk, not just in the obvious forward flowing way, but also in relation to each other. The total sum of all the motions are a well performed and efficient dynamic walk. The movements of the major body parts are briefly introduced in this section.

- **Pelvic list** describes the tendency of the pelvis to list downward in the non-weight bearing side. This results in an angular displacement of approximately 5° in the frontal plane, i.e. around the roll axis. The displacement occurs in the hip joints. Even without the pelvic list, the non-weight bearing leg must contract itself, by flexing the knee, in order to steer free of the ground during the swing phase, but the pelvic list sets up a requirement for a more aggressive contraction of the leg. It is not obvious why the pelvic list occurs, but it could be that it is more cost efficient to contract the leg, than to keep the pelvis level.

- **Knee flexion in stance phase**. Just before heel-strike the knee of the supporting leg is almost straight, but at heel-strike it begins to flex, and this continues until the foot is flat on the ground. This flexion is equivalent to a rotation of approximately 15° in the knee.

The knee flexion both absorbs some of the impact of the heel-strike as well as decreases the elevation of the center of mass and thus it helps in maintaining balance.

- **Sideways displacement of the body.** During the stance phase the entire body shifts slightly over the weight bearing leg. The magnitude of this shifting is about 4-5cm per stride, depending on the step width. This motion is maintained by rotation about the roll axis of the hip and ankle. This motion is very sex dependent
- **Swinging of the arms.** It has been known for some time that the swing of the arms during walking is not simply a passive movement induced by the mechanical displacement of the body, but that the muscles are activated cyclically [29]. Passive dynamics reveals that when the arms swings as a pendulum, oppositely of the swinging leg, it reduces the yaw torque on the feet, and thus increases stability on slippery surfaces.
- **Body rotations** The body also has parts which rotate in the transverse plane.
  - **Pelvic rotation** describes the rotation of the pelvis around the yaw axis. The rotation is for humans approximately 4° to the right and 4° to the left alternately under normal walking conditions. This effect elongates the step length.
  - **Thigh and shin rotation** The lower part of the leg rotates in phase with the pelvis, and the rotation increases from the pelvis to the shin. From the *midswing phase* the leg starts to rotate inwards, until midstance where the outward rotation begins.
  - **Torso rotation** During walk the torso rotates in opposite direction of the pelvis, this also animates the arms passively. Hence when a leg swings forward the opposite arm simultaneously swings forward due to the torso rotation.
  - **Ankle and foot rotation** Mechanisms in the ankle joint and the foot, absorb rotations from the leg while the foot is in contact with the ground in the stance phase.

Human walking, has been described as a series of phases, and some of the major body motions has been presented. However, to understand how the motions are created, can be of interest when designing a controller that is to move a robot in a human-like way.

1.5.4 Actuation's During the Human Gait

The human body in it self has some interesting mechanical properties which influences the behavior while walking. Especially because it has muscles for actuation of the joints, and in relation to robotics, muscles are interesting in their behavior as they are very different from the rotational DC motors often found on a robot.

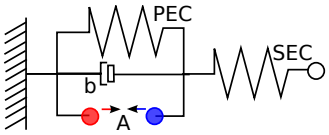


Figure 1.12: Mechanical model of a muscle.

1.5.4.1 Walking Using Muscles

According to [16] the mechanical model made in 1950 by Archibald Vivian Hill still represents a very reasonable model for the mechanical behavior of a muscle. The model is an empirical model, and replicates the behavior of muscles measured in a series of experiments. As only the mechanical muscle behavior is of interest to this thesis, the model is presented here without considering the actual biologic structure of a muscle.

The model can be illustrated as in Figure 1.12, and be expressed as in Equation 1.8.

$$\dot{F} = \frac{K_{SEC}}{b}\left(K_{PEC}\Delta x + b\dot{x} - (1 + \frac{K_{PEC}}{K_{SEC}})F + A\right) \tag{1.8}$$

Where:
A is the active force produced
x is the muscle length
b is the dampening effect
F is the total force produced by the muscle
K is the spring stiffness $\left(\frac{\Delta F}{\Delta x}\right)$

From the model of a muscle, it can be seen that even a muscle which is not actuated by neurons, passively influences on the motions of the joint it is connected to. This is because that the unactuated muscle behaves almost as a spring, (b tends to be much smaller than the two spring constants).

In Figure 1.13 it is illustrated which of the major bi-joint connected muscles that perform work in relation to certain phases of walking, the Figure is redrawn from [16]. The figure is presented as some of these muscles are not receiving actuation signals, it should also be noted that these particular muscles have the side effect that they dynamically couple the two joints they are connected to. Hence the spring effect of a passive muscle becomes important for the body movements.

In general a muscle reacts on signals from neurons, a single "firing" (or in electrical engineering terms a "pulse") results in a muscle twitch, where the peak force of the twitching muscle occurs after approx. 50ms, the rise time of the force depends amongst others on the size of the muscle. After it reaches peak force, it declines in a fashion similar to the rising.

A series of pulses with a pause time shorter than the force rise time of the muscle, will increase the force in the muscle before it has descended to zero. Off course if the frequency is very high, this results in an almost constant force, which can be controlled in power by the amplitude of the pulses. The force of an actuation thus depends on both the frequency and the amplitude of the signal, and can be compared to the concept of driving an electrical motor using pulse width modulation.

When considering the Electromyography (EMG) measurements of the body in motion, pattern recognition theory reveals that there exists a pattern in the signals sent to the major muscles from the brain during walking[41]. The pattern of the EMG signals can be roughly sequenced in five basic phases. Where each phase describes the particular subset of muscles which receives

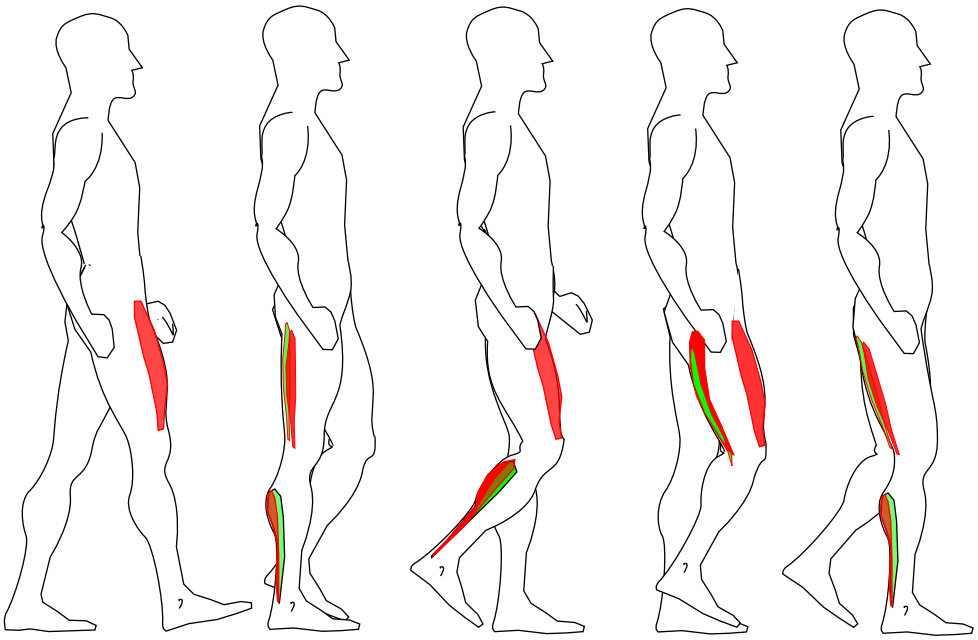


Figure 1.13: Two-joint connected muscles, which exerts force during walking, figure is adapted from [16]. Green muscles are from the inside of the leg. Red is front, back or outside of leg.

actuation signals. The remaining muscles in the body receives close to or no actuation signals in the time spanned by a phase.

The five phases correlates almost perfectly in time with the different walking phases described previously, but not with all the muscles that are exerting forces on the joints. This leads to the conclusion that some of the muscles that perform work on the joints does this passively, and not as a result of active control, while others are actuated by the nervous system to maintain the gait.

In Figure 1.14 the muscle groups which according to [42] are actively controlled are colored in a fashion similar to the one presented in Figure 1.13.

The difference in the two figures (1.13 and 1.14) illustrates that the performance of the muscle groups in the human body during walking is both passive and active. Both figures does however also reveal that only a few of the numerous muscles are actually actuated simultaneously during walking, and that a walking robot which replicates the human controller behavior is going to actuate very differently in the separate phases.

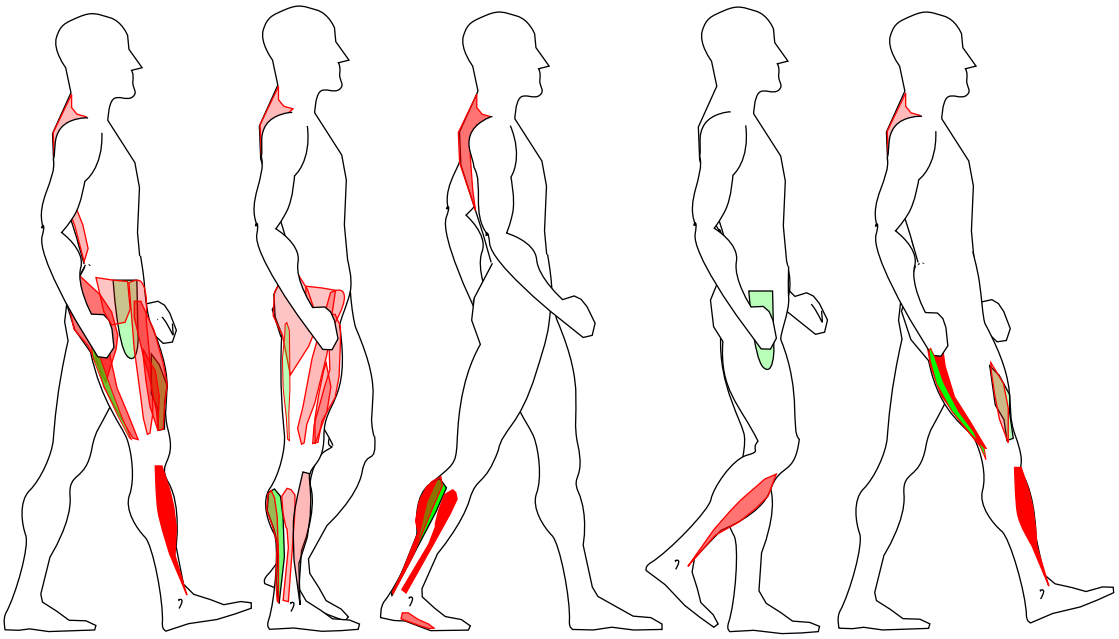


Figure 1.14: Larger muscles which have a significantly high EMG signal in comparison to the noise level. Green muscles are from the inside of the leg. Red is front, back or outside of leg, the intensity of the color indicates roughly the intensity of the EMG signals.

1.6 Bipedal Walking Robots

Other bipedal robots than AAU-BOT1 exists, and to this date many of them have been successfully controlled to achieve walking. Some of the developed robots achieves only static walk, but is often used for publicity, or to demonstrate hardware improvements such as small actuators for fingers, and or arms.

Other teams of robot developers have achieved dynamic walk in the recent years, but have done so either by utilizing large feet, reduced step length, or with a robot which is developed by research teams over a long period of time, for a corporation. In the latter case this unfortunately means that sparse documentation is available, and what may be available can be very robot specific solutions, which requires reverse engineering to study it.

In the following two examples are shown, to illustrate the two different approaches for biped walking investigated for this thesis.

1.6.1 ASIMO

Probably the most famous robot presented to this day, ASIMO features some of the best robotic control implementations seen to this date. See Figure 1.15 for an image of ASIMO.

Unfortunately ASIMO is a prestige and commercial project for Honda, and thus documentation is not released on their implementations and the sparse technical data available is released in Japanese, which have made it nearly impossible for this group to study the material.

The developers of ASIMO claims that they have achieved a form of dynamic walking, with a flat foot approach to ground contact, which allows them to run, and jump. While this may in fact be true, it is from the videos posted by Honda clear that the gait is not replicating a

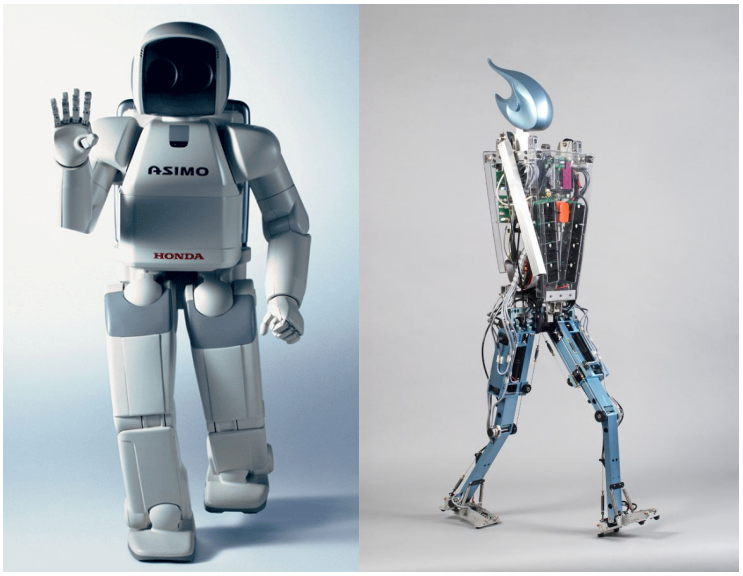


Figure 1.15: (left) Picture of ASIMO, (right) Flame

human-like approach to dynamic walking, and that limits the transferability of the gait to this project. It is however stable, and in many ways equipped with similar actuators to what is found on AAU-BOT1.

As a result of the gait applied ASIMO consumes much more energy than necessary to sustain a gait [40], and does not in the current editions convincingly demonstrate how the optimal implementation of an energy conserving gait for robots can be done. However the numerous tasks which ASIMO has been demonstrated to be able to solve provides inspiration for the next generation of robotic designs and control systems, not only in relation to gait synthesis but also in relation to tasks of work which the robot can be expected to solve.

ASIMO started out with the ability to walk statically, probably by using the ZMP as reference for control. Many robot scientists have used this approach to achieve static walk, and it seems effective for almost any implementation of a biped robot. However the gait which is achieved is far from efficient or aesthetical.

The ZMP specifies the point on the ground with respect to which, the dynamic reaction force at the contact of the foot with the ground, does not produce any moment, i.e. the point where total inertia force equals 0 (zero). The concept assumes the contact area is planar and has sufficiently high friction to keep the feet from sliding. The ZMP as control reference is no longer meaningful if the robot makes multiple non-planar contacts, and can thus only be used effectively on surfaces which comes very close to a planar surface, which is why robots as ASIMO often has difficulties walking up and down stairs.

1.6.2 Flame

In opposition to the flexible ASIMO, multiple robots have been designed for the purpose of walking only. It is amongst these robots that the gaits which closest resembles humans have been achieved so far. One of the successful dynamic walking robots is the Flame robot developed at Delft University. See Figure 1.15. It has a flat foot design, but is able to roll on the corners of the foot, as rubber hemispheres is mounted on the bottom of the foot[18].

Flame features linear actuators with serially connected springs, that to some extend mimics the mechanical behavior of human muscles. The gait is achieved by actuating the joints in a cyclic repetition of the movements in the sagittal plane, and the regulation of balance in the frontal plane is done by sideways placement of the moving feet. Yaw torques exerted on the feet is reduced below the foot to ground stiction by counter-swinging the arms on the robot.

The Flame robot is lightweight, and far from human in density with it's 15kg, however the gait achieved is clearly human-like in it's appearance. The strategy applied for walking with the Flame robot has not had focus in the articles found by this group on the robot, but it can indirectly be deduced by looking at it's ancestry. It's predecessor had no upper body and no frontal plane balance control, but still achieved dynamic gait. This was possible due to the foot design which guided the CoP during a step, and the counter swinging arms mounted at hip level. The only control applied was the cyclic repetition of an on-off state machine which regulated similar spring series actuators to maintain the sagittal plane motions.

The predecessor to the Flame was however unable to handle ground variations of a few mm. Flame can handle quite large ground variations through it's simple control strategy of step width regulation. The approach relies on the hardware, as it is necessary to have almost frictionless joints, and near mass-less legs, to achieve the passive dynamics gait, which the Flame robot relies upon.

1.6.3 Walking Robots from AAU

Since the AAU-BOT1 was handed over to the Department of Control, the goal set has been to achieve static walk with the robot, basic research for static walking was done on bipedal robots using the much smaller and cheaper AAU robot Roberta. The robot was 58 cm tall, had 21 DoF, and was actuated using RC servos. The performance of the robot was not convincing, due to a slow control loop, maxing out at 50 Hz. This made it difficult to test the models used to describe the robot, and during an upgrade attempt the year after it's construction the robot was partly destroyed, and was never reassembled properly.

The modelling method used on the control department on AAU for bipeds was established on Roberta, and this model was later applied on a AAU-BOT1 simulator. The model suggests a Lagrange approach to modelling, which results in a model that seems to describe the system quite well under the assumptions that only 1 foot has ground contact, and that friction forces are so great that it can be assumed that the foot does not move horizontally.

This type of model has the inherited problem that closed chains (which arises when two feet has ground contact) is near impossible to calculate.

For the situation where 2 feet touches the ground, it was suggested for Roberta to use a novel approach to biped modelling, which "ignores" ground contact on the first foot, and calculates the model for the other, then "ignores" the second foots ground contact and calculates it from the first foot, and finally does a linear combination of all the forces and torques which is found in the two models.

Roberta was never able to walk, and it was assumed that the primary reason for this was the slow communication to the actuators. As a result the model was never verified but was never the less inherited to the AAU-BOT1 models. Here two variants was produced, one for simulation, which adds ground forces to the model, and one for the controller which still assumes infinite friction and rigid connection with the ground.

However in practical experiments AAU-BOT1 was unable to handle balancing, and when standing still the posture regulation based on the control model became unstable.

This group have investigated this behavior, and noticed that the robot is able to keep it's balance for a significant time if initiated at a position where it effectively stands on 1 foot.

However when both feet have ground contact the controller is very aggressive in ankle roll, which makes the robot unable to sustain ground contact, and thus unable to sustain balance.

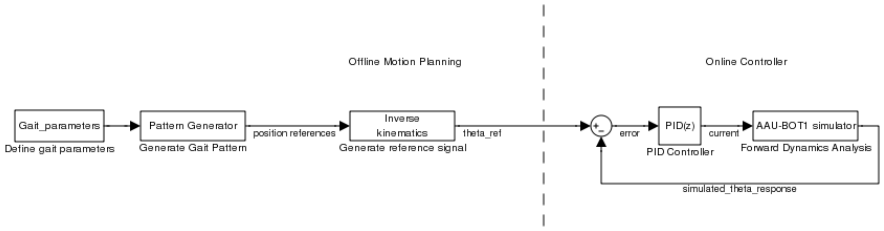


Figure 1.16: Strategy used by [27] for testing the design in simulations.

1.7 Analysis of AAU-BOT1 Walking Attempts

Previous AAU-BOT1 groups have attempted walking with the AAU-BOT1 robot. This chapter provides an overview of the results, and an evaluation of them.

1.7.1 First Attempt, Virtual Dynamic Walk, No Balance Compensation

The mechanic group [27], attempted dynamic and human like walking. The robot was not actually build at the time, hence their control strategy was only tested in simulation. The simulator used, was based on rigid body dynamics, and solved through numerical integration using ode45 in MATLAB. Their overall control strategy was quite simple, and was not compensating for balancing, as an offline trajectory was tracked without balance compensation.ee Figure 1.16. However they walked multiple steps in simulation, and proved that the robot with the designed weight ratios should be able to walk.

The approach to trajectory generation that was used was to predict pelvis motion as a 3D LIPM, revolving about stepwise located ZMP. I.e. if the ZMP is within a foots PoS, that foots center is used for attachment of the LIPM, if none of the feet has the ZMP within the respective PoS it is placed in the exact middle of them.

The resulting motions are not entirely human-like, hence it was also tried tracking the recorded human trajectory. Which all was successive in simulation. The approaches suffers from a lack of feedback, hence if the heel impact happens much before, or later than assumed in trajectory generation, the robot does not compensate and can fall.

1.7.2 Second Attempt, Virtual Static Walking, with Balance Compensation.

The Second Group [31], attempted static walking, as their job was mainly to verify the instrumentation. This trajectory was actually replayed on the robot hardware, but only while the robot was hanging in the air. As this can cause dynamic problems, it was decided to us a very slow trajectory, and thus it was necessary to use a static walk. The playback was done using the build in PID regulators in the EPOS.

[31] chose a posture control scheme known as a LQG Controller, see Figure 1.17, this allows for specific weights on the performance, and on the influence from other joints on the controller gain. This structure allows for optimizing the gains such that a high performance can be achieved while allowing cross couplings to be suppressed, which should give better performance and a more robust controller than a simple PID system.

The LQG control scheme is a powerful tool, but can be difficult to optimize, and especially when the plant is highly non-linear, as is the case with AAU-BOT1. This was clear as the

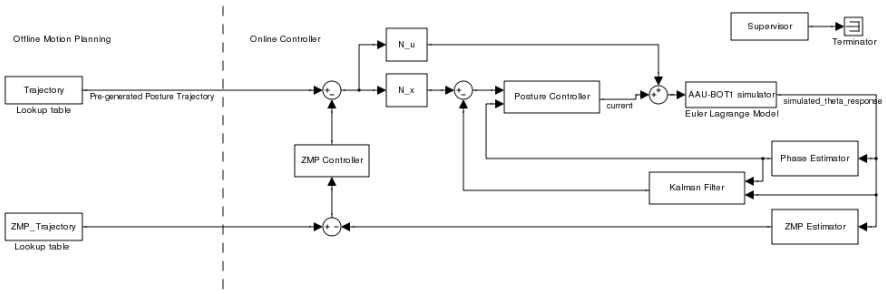


Figure 1.17: Balance Compensation Strategy for Static Walking. Notice the structure with the Kalman filter and the two N matrices. Together they form a LQG controller.

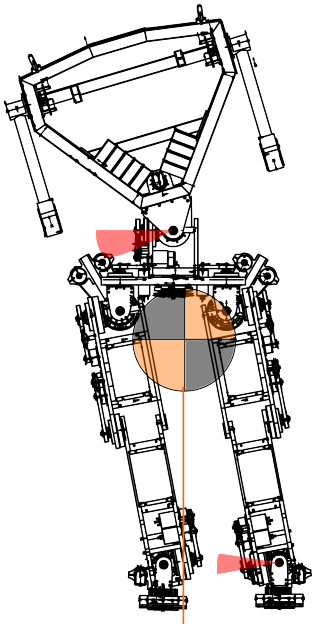


Figure 1.18: Here the robots CoM is drawn while the robot is tilted towards one of the feet. The left ankle (farthest away from the CoM) has met its rotational limit in the roll angle, and so has the waist roll angle. Hence the CoM is shifted as much as possible towards the right foot. However the GCoM barely is within the foots PoS.

group was unable to stabilize the non-linear model without using very large gains, which on the practical model, and in an WebBOTs simulator made the robot unstable.

As a result they could not verify the ZMP balance regulator strategy either.

The group noticed in their report that static walking with the robot would be difficult if not impossible due to joint limitations in the ankles, and the lack of weight in the upper body. However the problem is not investigated further in the original documentation.

A short analysis of the problem reveals some of the issues with attempting static walking on AAU-BOT1.

In Figure 1.18 the frontal plane of the robot is illustrated with the robots CoM moved as far right as is kinematically possible. It is clear that the GCoM is just within the foot PoS but not

by much, giving little room for balancing issues.

Another problem is that even though the GCoM is just within the right foots PoS, and that it should be theoretically possible to stand on that foot alone, the torque required to keep the foot in the same angle, while lifting the other foot, is larger than what can be produced by the relative small actuator.

I.e. the torque required to sustain the stance angle amounts to approximately:

$$\begin{aligned} \tau &= m * g * r \\ \tau &= 37.5\,[Nm] \end{aligned}$$

where:
$r = 0.057\,[m]$ is the horizontal distance to the joint.
$m = 67\,[kg]$ is the mass of the robot.

The torque available (in the case of no friction) should be:

$$\begin{aligned} \tau_a &= k_t * G * I_l \\ \tau_a &= 18.5[Nm] \end{aligned}$$

where:
$k_t = 0.0538$ is the motor constant
$G = 200$ is the combined gear ratio
$I_l = 1.72$ is the nominal current limits.

Furthermore it does not help to adjust the pitch angles symmetrically, as that simply moves the CoM towards the center of the feet; moving the initial angles in such a way that the feet is 7-10cm closer it might be possible, but not really feasible since only small dynamic motions from stiction would make it unstable.

It has been speculated that the nominal current limits may be disrespected for a shorter period of time and Maxon, the DC motor supplier, actually documents how long a time this can be allowed. Hence it may be possible to provide the torques needed, but only for a short period of time, and taking a step within this time frame has not been done successfully. It should also be noted that the ankle roll joint has a stiction that must be overcome, which is estimated as more than 5 Nm in the unloaded case.

1.7.3 Third Attempt, Static Walking with Non Linear Control.

Focus in the third project [30] was laid on posture and balance control, the actual walking trajectory was chosen as a static ZMP based trajectory. The trajectory was verified in simulation, but the posture controller never worked in practice, and the robot never walked.

The overall controller structure that was suggested does in many ways build upon the work done by [31], as can be seen in Figure 1.19. It is still an offline generated static trajectory, and the balance controller is still applied as a modification of the reference trajectory.

The trajectory created for walking statically is in this thesis deemed impossible to realize due to limitations on the robot, with regards to joint angles and torques. The static trajectory produced by [30] was found using a search algorithm and a simulator, which they also used for verifying their trajectory. This simulators behavior in double support can have contributed to the assumption that the trajectory was possible. Also they allowed a violation of the nominal current limits with a factor of 4 for a short duration of time which may have enabled them to shift the CoM closer to the ankle, and then as a result afterward reduce the torques.

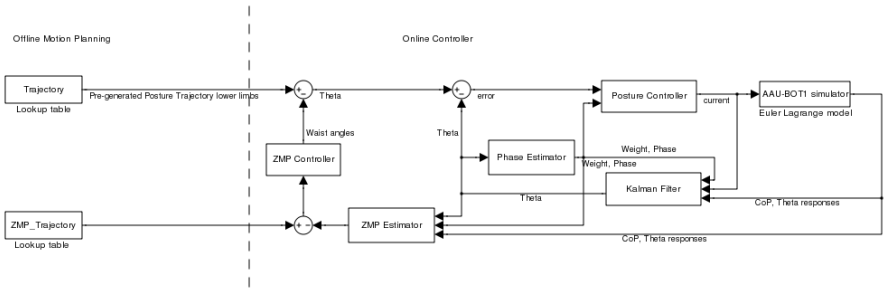


Figure 1.19: Balance compensated strategy. The posture controller is considered a subsystem with well defined interfaces, and the balance is adjusted on moving torso only. The kalman filter is an unscented kalman filter allowing a very accurate estimate of the states of the system.

However this has not been tested outside simulations, and it has not been possible for this group to replicate the claim in practice, within reasonable time.

1.8 Summary of the Prologue

The development of a controller for AAU-BOT1 was at the beginning of this thesis not at a point where walking could be realized. It was estimated before the initiation of the project that static walking could be achieved in the summer of 2009, and that the controllers developed for static walking could be transferred to a dynamic trajectory and thus achieve dynamic walk.

Multiple explanations can be set up to understand why this has not happened. It is not the purpose of this thesis to analyze in detail why this goal was not met. However some indications have been found that it is close to impossible to walk with AAU-BOT1 in this manner, and as a result of this analysis this thesis abandons the strategy of reaching a static gait before a dynamic gait. Even if it is somehow possible to walk statically, i.e. by initializing with very small step width, it is deemed improbable that it is achievable, and there is limited time available to focus on solving this task.

If a controller should only handle angular positions of the joints, which a generic controller is often expected to do, it would still take time to develop it for a biped robot, simply because of the high number of DoF's. If unlimited funding is not present, it requires a fast and accurate simulator, which reproduces the behavior seen on the robot. Otherwise any controller testing, and parameter tuning will quite quickly become very expensive.

A simulator which does not behave like the robot even when the robot stands still and actuates a joint, is not optimal for controller testing, and thus time has to be spent on creating a new test environment for the current and for future developments.

A robot controller is however also expected to keep track of balance and should be able to correct deviations by some actuation strategy. This strategy can be very different for static and dynamic gaits. For static gaits it is often chosen to have a heavy upper body, and then control the angle of the waist, to force the ZMP and CoP inside the PoS at all times. This is clever as it almost decouples the balance control from gait trajectory of the legs.

For dynamic gaits this approach might work, but as the Flame robot shows by example (see 1.6), other approaches can be implemented which instead of continuous balance control, plans a position of the feet to accommodate balance issues.

AAU-BOT1 has a weight distribution which makes the upper body lighter in comparison with other robots, and thus the approach for static walking and balancing requires large movements of the upper body to create the necessary momentum. Hence a dynamic gait that utilizes this approach might work, but it would never be human-like. So far it has however been the approach taken by previous groups to achieve static walk, which is in op.

Based on the research done prior to the development, it is chosen to attempt to replicate the system used for balance controlling PDBR's, and to use the basic dynamic behavior shown in passive dynamics robots, to on-line predict a trajectory which can be tracked.

Where big differences appear between PDBR's and AAU-BOT1 a posture controller needs to compensate, or the individual part of the trajectory needs to be modified. However the basic approach will remain inspired by PDBR's.

2

Conceptualization

"We need to decide as a nation what we want to do [in human spaceflight]. We shouldn't start by designing the next vehicle. That is a trap that we've fallen into several times."
— **Admiral Harold W. Gehman Jr.** (Columbia Accident Investigation Board chairman)

2.1 Project Outline

The concepts for this thesis is presented here to provide an overview of the content. The overall control system concept is based on the analysis presented in the Prologue part, and so is the choice of tools developed for this project.

The work done for this thesis involves practical work, development of tools for controller testing, and a controller design. The practical work and the development of tools such as the simulator, is time consuming, and thus any idea for a tool, or practical improvements has to be considered carefully, before it is initiated.

At the initialization of the project regarding this thesis, a chart was created to get an overview of the tasks needed to be solved before dynamic walking could be achieved. The chart is shown in Figure 2.1. This narrowed the project tasks down to designing a control system, however as the tools previously developed, such as hardware drivers, and simulation models, was investigated it was obvious that more tool development was needed before a controller could be tested, tuned, and implemented.

The considerations and some of the research done specifically for conceptualizing the behavior for each element in the thesis is presented in the following Sections, this section provides an overview of the elements, and how they are interconnected.

At first the controller structure is presented, as the main focus of the thesis is the development of a dynamic walk controller. See Figure 2.2 for an overview of the control system concept. The actual design of the control elements, and the tools is found in the next part of the thesis.

Controller Structure

The control scheme chosen relies on the knowledge that the robot is able to stand when shut off. Hence no balance regulation is necessary when standing still, as long as the robot is not

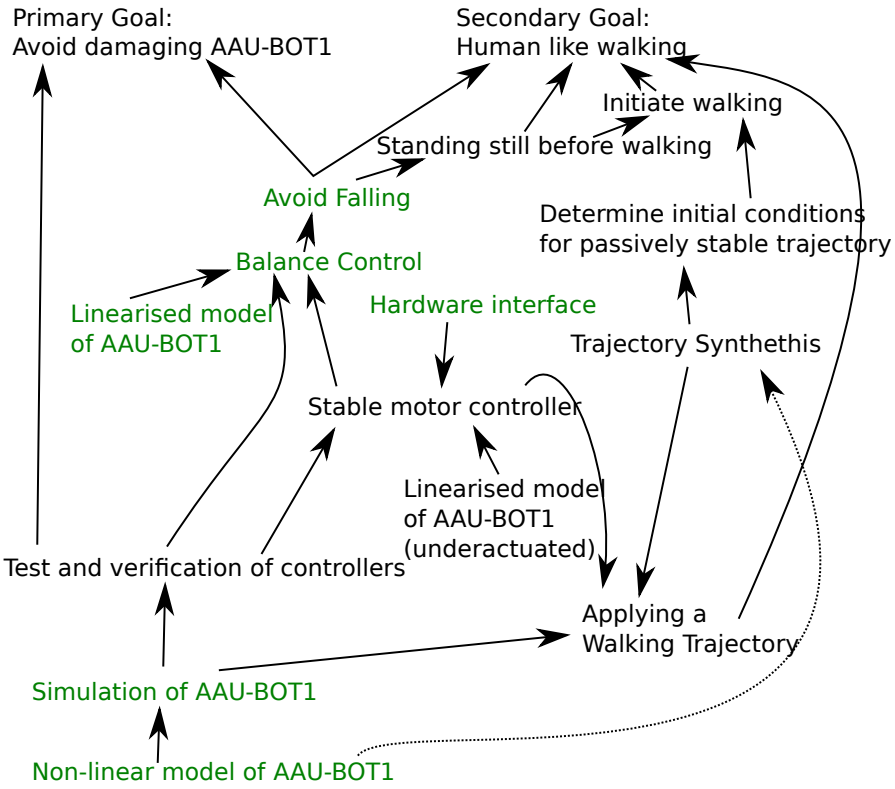


Figure 2.1: Initial chart for planning project tasks. Green text marks tasks that was reported by previous projects as solved.

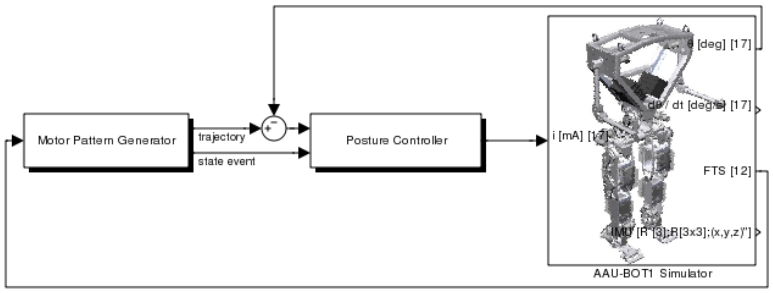


Figure 2.2: The Control system is made up of two individual parts. The motor pattern generator handles trajectory generation, and discrete events. The continous posture controller, acts as a servo controller. The gains in the posture controller may be modified when state changes in the motor generator.

interacting with personnel.

The trajectory is generated on-line, using sensor data to detect discrete events, and thus becomes a part of the feedback loop. The trajectory is then applied to the posture controller, which transforms the angular references into currents for the actuators.

If the robot never interacts with anyone, or anything, the balance regulation can be considered a matter of sustaining walking, by sideways placement of the feet. This choice places balance controlling within the on-line trajectory generation.

The control scheme is then a two part cascade coupled control scheme, with a posture controller as the inner loop, and a Motor Pattern Generator (MPG) as the outer loop.

Posture Controller

The inner loop in the controller is chosen as a classic PID controller, though it is from here on dubbed posture controller. Especially when concerning this part of a robot controller future research on the AAU-BOT project can and should be carried out, as there is room for optimizations with regards to handling dynamic cross couplings.

The primary concern to this group has been to setup a dynamic balance regulation scheme, and an on-line trajectory generation system, the posture controller implemented is created as the outer loops require some posture regulator to be available, this choice has been made to reduce development time.

Motor Pattern Generator

The outer control loop consists of a state machine which is inspired by the human pattern generator system. State transitions relate to the basic support phases, and the robot angles, and are thus affected directly by events occurring on the system. As a result of this setup the outer loop can change the models used for trajectory generation, and thus the behavior depending on the state of the robot.

The trajectory reference must be formed to take into account the current speed and reference while walking. To form the pattern (and thus the trajectory) a reduced dynamic model is used to predict robot behavior, and control it. The model used is inspired by passive dynamics robots, as they give tremendous insight into the basic dynamical principles governing dynamic walking.

In single support the robot is considered a spherical inverted pendulum with a bob in the CoM and a rod to the supporting foot, it is allowed to rotate with no friction about an angle. As a result the task in double support is to throw the robots CoM ballistically into the next single support, with proper initial conditions.

The remaining part of the controller is balance keeping and to move the swinging leg, the two tasks relate to each other, but is considered two separate tasks, hence sagittal plane foot movements are simplified, and step width is chosen as the only balance correcting system.

Visualization tool

It was decided to developer a robot visualizer for interpreting simulation and experimental data. This choice was made as the current implementations of visualizations was not well suited for 3D visualization, and that a 3D tool is immensely powerful for understanding the 3D dynamics.

Simulator

It was decided that a new simulator was needed, that could represent the robot non-linearly, and provide interfaces for controller testing. This was deemed necessary as the existing simulator

was unverified, and that the model techniques used was unfamiliar to this group.

Hardware Improvements

It was deemed necessary to improve the assembly of the AAU-BOT1 hardware, and update the build procedures, as it required much contextual debugging to run a controller, and thus became tiresome to optimize or test a control scheme. Especially the placement of the OBC on a table, required many cables connecting it to the EPOS on the robot, this created dynamic properties which was not included in the models, and it was decided to attach it to the robot. Also multiple bugs was found concerning driver implementations, many of which has not yet been solved, but as a result an observer was implemented that watches for known bugs, and alerts the operators if any occurs. The result of these choices is that much of the hardware setup has been modified, it is however not directly documented in this thesis.

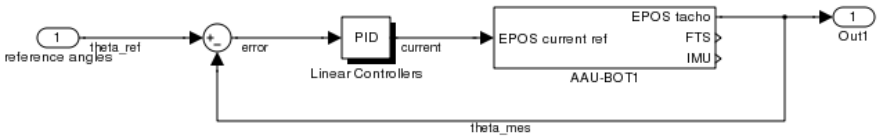


Figure 2.3: A classic PID controller. The simple structure dues not encompass data from other joints, and is thus 17 individual SISO controllers. This structure is prone to cross coupling problems, in this type of systems, and has no explicit compensation for non-linear dynamics.

2.2 Posture Controller Concepts

To move the robot the individual joints must be regulated in some manner. Previous projects has been considering at least three types of controllers, of which some has been more successfully implemented than others.

The three types of controllers attempted has been PID, LQR/LQG and Computed Torque controllers, and as they have individual problems and advantages, and was all considered for this thesis they are briefly explained here.

2.2.1 PID Controller

By considering each joint as a linear system, without any dynamic couplings, the classical Single Input and Single Output (SISO) controller dubbed PID has been applied to transform the individual DC motors to servos. The controller has several drawbacks in comparison to other more complex Multiple Inputs and Multiple Outputs (MIMO) controllers which can be build to take into account joint cross couplings. See Figure 2.3 for an overview of the PID structure.

Apart from the drawbacks concerning cross couplings, there are big differences during walking concerning the joints linearizations. Especially concerning the ankles, as they experience a prismatic load difference between 1kg from the hanging foot in air and 66 kg from the robot when standing on the foot, which states that it cannot be optimized to both cases.

Assuming that a compromise can be found, there now exists two possibilities on the robot. Each EPOS has a build in rate limited PID controller, and a PID controller can also be implemented in software as the EPOS can be configured to act as a current controller.

If choosing the rate limited PID controller, the proportional gain can be chosen very high, as the rate limit reduces the risk of overshooting. This has been done for the control software that is used for acquiring zero positions, and other joint angles, while hanging in the air.

The build in PID controller has some limitations concerning double actuated joints, where any small measurement disagreement concerning angles would create oppositely directed torques, if the position was actually acquired. Hence only one of the two actuators are used, which makes this controller unable to walk, as there is not enough torque available in critical joints such as the knees.

This controller type has when implemented in software been successfully shown to be able to control the individual joints, and walk without balancing in a dynamic simulation.

It was noticed in the modelling phase that the individual joints has a high amount of friction, which can be utilized in a regulator to ensure stability. This is because the Viscous friction has the same properties as the Derivative part of a controller, but has the advantage of being continuous. Discrete derivatives are by nature prone to noise, and can cause a controller to become unstable if the sensor data is noisy.

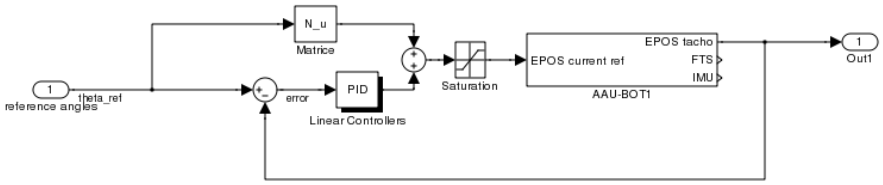


Figure 2.4: Block diagram of a simplified LQR regulator, The PID gains is actually a combined matrix, containing state feedback gains, but as the states are angles, and their derivatives, the resulting controller is the same as depicted here. This type of controller improves the strategy known from PID regulators.

2.2.2 LQR/LQG Controller

An Linear Quadratic Regulator (LQR) controller has been suggested by previous groups. The LQR algorithm is, at its core, just an automated way of finding an appropriate state-feedback controller. A control problem is described as a cost function, where references are absent, and then by applying the LQR algorithm the gain matrices can be defined which stabilizes the model. The model used to calculate the LQR gains is the linearized state space representation of the AAU-BOT1.

Due to the choice of states, the resulting controller, can be thought of as multiple parallel PID regulators, with a feedforward compensation. Due to the difficult choice of linearization points, two suggestions are made for the cost function model.

The first one, is a linearized robot i zero position, which has been shown in [31].

The other is a stepwise linearization, where the cost functions needed to create the feedback gain matrix is determined by a pendulum approach to modelling each joint. Suggesting that a linearization point could be a model where all other joints on the robot is welded.

The fundamental difference between the two models is a matter of viewing the robot as a system that is naturally in double support or single support.

The strategy have not work out for any of the groups that suggested it, so they abandoned it, assuming that the system was to complex to stabilize with such a simple controller. They stated that the cross couplings have shown to make it difficult to design individual controllers for each joint. This is in direct opposition to the findings done by the original mechanics group that could apply the even simpler PID controller in simulations. Also it does not comply with the fact that the physical robot was at the time controllable by a rate limited PID controller. It is suspected that these conclusions can relate to the choices of simulators, and the optimization to an erroneous choice of linearized model and thus the conclusion could be erroneous.

An LQR controller could increase the performance and cannot be rejected as an alternative.

For an overview of an LQR structure see Figure 2.4.

The LQR controller can be improved further by encompassing a kalman filter. This ensures that the derivative parts of the controller does not cause instability, as the filter will reduce noise. Also the model used in the kalman filter can be shaped to determine states that was otherwise unknown, i.e. The passive toe joint angles, or the robots CoM location.

These estimated states can then be fed into the gain matrices. Which allows the controller to be optimized against a cost function that takes into account states which are not directly measurable, and thus it can improve the controller performance further.

The LQG is optimized by the LQR optimization algorithm, and thus relies on a good model of the behavior of the unknown states. However with some imagination, the extra states can be chosen to extend the range in which it can stabilize the system, and this feature is very powerful.

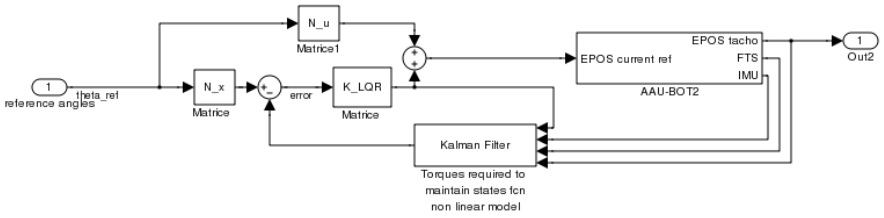


Figure 2.5: Block diagram of a LQG regulator, The LQR gains contains state feedback gains. Due to the kalman filter, states can be chosen which are not directly obtainable, and there is less danger involved in using the derivatives.

The structure of an LQG controller is shown in Figure

2.2.3 Computed Torque Controller

Using a simplified dynamic model, some of the dynamic effects can be inverted in the feedback loop, and used for an on-line linearization of the system. This allows for on-line compensation for cross couplings and thus greatly improves the performance of a linear controller, as the actual actuator output resembles that of a nonlinear controller.

As a motivating example consider a pendulum, and assume for simplicity that it has some friction in the joint and thus it can be controlled stably by means of a proportional controller. Then even without considering the error due to frictions, the basic physics will introduce larger tracking errors as the pendulum rod approaches waterline. This occurs as the torque required just to maintain the angular position increases. A better controller, being the classic PI controller would slowly decrease the angular error due to the integral part, and after some time it would be able to reach the reference angle.

A perfect nonlinear controller would compensate for the extra torque required to reach and maintain the angle, and just increase the torques non-linearly as a function of the pendulum angle and velocity and thus track the reference, which is much faster, and more stable than a PI controller can do it.

A feedback linearized controller for this example would use a pendulum model to predict the extra required torque, and then add it directly to the control systems inner loop, see Equation 2.1, hence the angle/torque relation is linearized from the view point of the cascaded controller, and the proportional controller mentioned earlier will be as good at maintaining waterline angles as it is vertical, that is even without an integral part.

The actuated non-linear pendulums equation of motion is:

$$T = I\ddot{\theta} + mgl \cdot sin(\theta)$$

The computed torque equation, for the feedback linearization controller is

$$\begin{aligned} T_c &= mgl \cdot sin(\theta) + u \\ u &= I \cdot \theta \end{aligned} \tag{2.1}$$

Notice that the input signal model u became linear in the example, but that it is only possible if m, g and l are well known. In general the assumptions made for such a feedback linearized system to work properly is that the model used is adequately well describing the system. If the

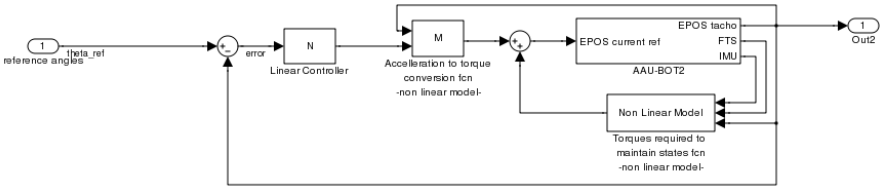


Figure 2.6: Feedback linearized controller, for computed torque control strategy.

system is parametrized reasonably well, and the feedback part has a model structure of a suitable order, it does in fact always linearize the plant behavior.

Should the model be too simple, too complex, or have poorly estimated parameters, it will instead of linearizing the plant just distort the recorded sensor data, or the control signals using a nonlinear filter. The result can be that the real plant has become even harder to stabilize than it was the case without non-linear part of the controller.

A feedback linearized structure can be seen in Figure 2.6, here it is clear that the nonlinear models are located in the inner loop of the controller, which distorts the linear controllers signals if the robot is not in the point in which it was linearized.

The previously implemented computed torque controllers based on feedback linearization was not able to control the robot. The controller could not even maintain the zero position in which it was linearized if it had ground contact. Which is odd, as the robot can do that passively due to friction in the joints. The controller was however verified in the existing simulator.

The advantages of the computed torque method goes well beyond the ability to handle a pendulum, as it has been shown numerous times with other robots that it is possible with a good model, to develop a single posture controller that can handle a robot in both the single support phases and in the double support phase. This eliminates the need of switching the inner controllers, which could otherwise have been necessary to accommodate changes in joint loads.

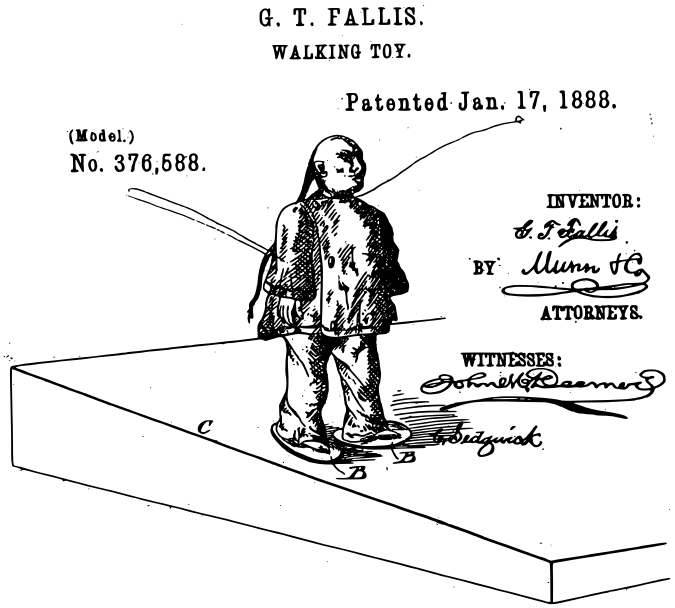


Figure 2.7: 1888 Fallis Ramp walking toy sketch.

2.3 Introduction to Passive Walkers

Two legged ramp walking toys has been known the last century, as early as 1888 George T. Fallis filed a patent for a ramp walking biped toy[13], see Figure 2.7.

The ramp walking toys are usually straight legged, and must rock from side to side in order to get ground clearance for the swinging leg. This produces a wadling walk, quite comparable to the walk of a ducks .

Passive walkers in general has no actuators, and walks down a ramp by means of the machines dynamics and gravity only, hence they provide a reference for a highly efficient bipedal dynamic walk.

Original walking toys as Fallis's, had little to do with human walking, but the efficiency of these simple walker toys inspired Tad McGeer to research their dynamics. To simplify the models, the design was refined towards a 2d walker (Sagittal plane) by setting up the legs in pairs, almost like walking with crutches[24]. Thus he could describe the dynamics of straight legged walkers in 2d.

The dynamics of such a walker is somewhat simple and reduces to a double inverted pendulum model with the two governing motions of equation, see Appendix B.3 for derivation, when assuming that legs are allowed to swing through the floor:

$$(m_1 + m_2)l^2\ddot{\theta} + m_2l^2\ddot{\phi}cos(\theta - \phi) + m_2l^2\dot{\phi}^2sin(\theta - \phi) + gl(m_1 + m_2)sin(\theta) = 0$$

$$m_2l^2\ddot{\phi} + m_2l^2\ddot{\theta}cos(\theta - \phi) - m_2l^2\dot{\theta}^2sin(\theta - \phi) + glm_2sin(\phi) = 0$$

with a speed loss at impact at:

$$\omega^+ = cos(2\theta^-)\omega^-$$

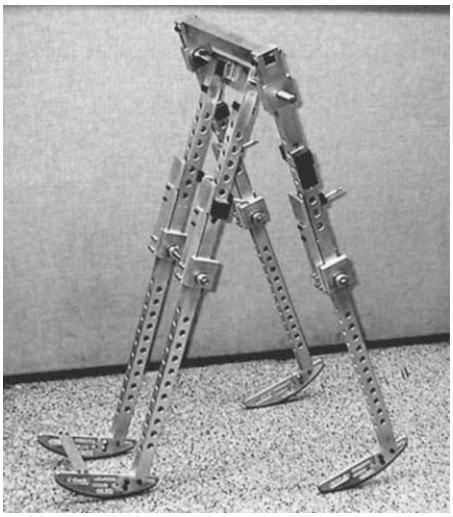


Figure 2.8: An imitation of McGeer's (1990) design, made by Yan Yevmenenko. The motion is constrained to two dimensions by the four legs. The image is borrowed from [6] as the quality is better than the photo found in the original McGeer article.

Where:
θ is the stance angle
ϕ is angle of swinging leg.

It is noteworthy that if the hip mass is much larger than the foot mass, the equations of motion reduces to a simple inverted pendulum, which indicates that an inverted pendulum model may describe a biped system in single support reasonably well.

$$\ddot{\hat{\theta}} = \frac{-g}{l} sin(\theta)$$

In 1990 McGeer published an article about the dynamics of the "crutched" 2d walker where knees was added to the model[25], see Figure 2.8, and as a result he had solved issues about ground clearance.

The kneed passive dynamic model presented by McGeer is very interesting, not only can it recover from small disturbances without the use of a controller, but in 2d it also creates a trajectory which is quite similar to the human walk, see the cartoon in Figure 2.9 for an illustration of the gait performed by the kneed walker.

The gait cycle time exhibited by the 2d walker is generally a bit slower than human walk, but the difference is according to McGeer due to the elasticity of the human body, which increases the legs pendulum frequencies in comparison to the more rigid machine used by McGeer[25].

While the kneed 2d model in itself is interesting to investigate further (see appendix B section B.4 for a derivation and analysis of the model), it is like the straight legged walker, not diverting much from an inverted pendulum model, and it is not confronting the problems of maintaining balance in the frontal plane.

Maintaining frontal plane balance was however already handled by the original 1888 walking toy by Fallis. In the original toy design the shape of the feet, guides the walker to obtain a rocking motion, see Figure 2.10.

Returning the kneed walking models to 3d was done by Andy Ruina, Steven H. Collins and Martijn Wisse in 2001 [6]. Basically the knees from McGeer's 2d walker was combined with

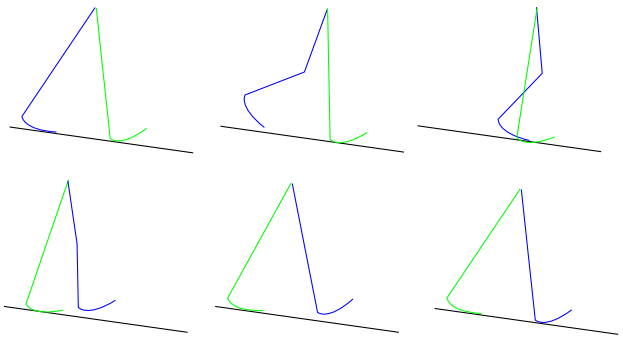


Figure 2.9: Replication of a cartoon found in [25]. It illustrates the cycle of a 2d walker with knees, the movements are quite close to the leg movements of humans in walk.

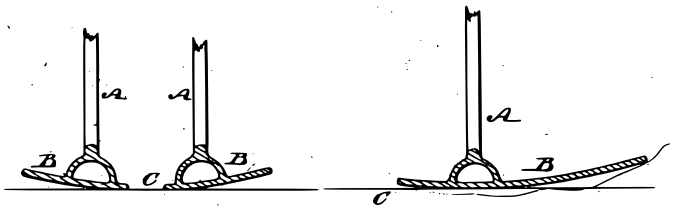


Figure 2.10: Figures representing the foot, from the Fallis patent

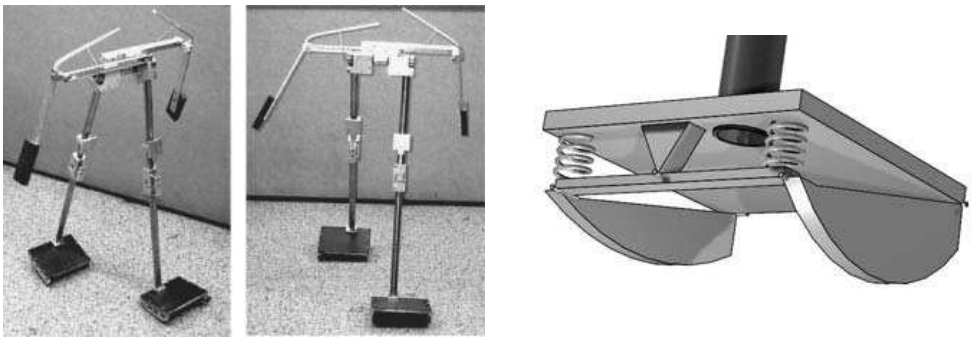


Figure 2.11: Image of the kneed 3d passive dynamics walker, and the foot used to achieve walking. Notice that the curvature is different on the in and outside of the foot. The largest radius is used at the inside, to create a shape somewhat similar to G.T.Fallis..

the foot shape from the original straight legged 3d walker toys. The resulting 3d biped passive walker, was able to walk, but had a tendency to exert to much yaw at the hip. Counter swinging arms was added to the hip design, and that successfully reduced the yaw motions, see Figure 2.11.

In 2005 Andy Ruina, and Steven Collins added power to the design, and thus by means of simple on/off state machine control of the actuators, reproduced the passive walking behavior on flat ground[7]. This has the interesting side effect that it has an energetic cost of transport which is almost equal to humans walking on flat ground.

The latest addition of the actuated passive dynamics walkers is the "flame" robots at Delft University, which uses a sideways foot placement strategy to keep its balance using information

from the inertial sensor, and thus regain balance in even quite large vertical ground disturbances.

2.3.1 Basic Passive Walker Principle

Supposing a functioning passive walker with knees is presented, as shown in Figure 2.9.

While the walker is in the swing phase, the stance leg has a locked knee. Hence the movements of the hip closely resembles an inverted pendulum on the stance leg. As the swing phase is entered with an initial angular velocity, the "inverted pendulum" will swing by it's vertical position, and afterward accelerate towards the ground in front of the walker.

During this time, the free leg swings forward, with an unlocked knee, just as a double pendulum would. The top pendulum relates to the thigh, and the lower pendulum relates to the shin. The shin pendulum swings faster than the thigh, which again swings faster than the stance leg. When the shin and thigh forms a straight line, the knee is locked, and due to the weight distribution of the walker, this happens just in time to have the knee locked swinging leg "catch" the falling hip.

Heel strike occurs, just after the knee is locked, and due to the impact with the ground, some of the angular velocity of the walker is lost.

The double stance phase is entered with some remaining angular velocity, which forces the walker into a new swing phase, now with the other leg as stance leg.

The ramp, or the actuators on the walker, provides extra angular velocity in the swing phase which matches the angular velocity lost at heel strike. Hence the system is stable and walking, and very human like in appearance.

2.3.2 Adding Torso

The basic walker described does not have a torso. It is not necessary to have a torso for the dynamics to work properly, as long as the hip is much heavier than the feet. In theory, a vertically mounted torso, does not affect the walker dynamics differently than an increased hip mass would. However keeping the torso vertically is not a trivial mechanical task, and this is one of the reasons why it has only been done recently. The passive dynamics based robots that has been equipped with a torso so far, has been fitted with a control system, to keep the torso vertical.

2.4 Trajectory Generators

In general the trajectory for the dynamic walker, can be either offline or on-line generated.

If faced with an offline generated trajectory, it is necessary to generate the trajectory of at least a single step in the gait, and then replay this trajectory in a loop. If nothing else is done to the trajectory then the result of this is, that with a proper posture controller, the robot is able to maintain a preknown speed, and motion. When tracking the pre-calculated trajectory is impossible due to some noise, like uneven surface or slippery floor, a set of mathematical rules can often be set up to try to recover the robot from the situation, such that it can track the trajectory again. This task is often handled by a separate balance controller, which as an example can monitor the ZMP movements and apply torques somewhere to the robot to recover the ZMP trajectory. The primary disadvantage of such a set up, is that movement speeds or directions which has not been pre-calculated can only be achieved if it can be somehow be build from pre-calculated trajectories, and this is not always trivial.

Also it is almost impossible to walk on slopes, if this has not been specifically pre-calculated, simply because it is far away from the trajectory used on flat floor.

An on-line trajectory generator, is often avoided by robot scientists, as it usually requires more computing power on the robot, and that an on-line trajectory generator can be quite complex to set up, even without concerning the calculation times. However the advantage of on-line generation is that if the trajectory is generated on-line, the trajectory generation system can be programmed to handle new directions and new speeds, and larger disturbances if rules can be set up that are generalized enough.

Also if the trajectory is not tracked properly then the trajectory can simply be re-planned from the current situation. For a walking robot this can be interpreted as the ability to react to sudden disturbances, and change both speed, feet placement, and upper body angles. A robot which follows an on-line generated trajectory, can also be conceptualized as a robot which does not track a known trajectory at all but simply reacts to it's surroundings.

An on-line generation system can result in non-cyclic gaits, and this makes it hard to guarantee stability, using existing analysis tools. Also on-line trajectory generation is a topic of ongoing research, and no general solution has been found yet. Hence it can become a time demanding task to set up, and it becomes necessary to compare the features of such a system against an offline generated trajectory, and evaluate which solution is more likely to be feasible within the scope of the project.

Articles and books on bipedal walk, has different ways of expressing trajectories. Not only does different scientists disagree on the conventions for modelling a biped robot i.e. orientation and location of angles etc. They also disagree on the definition of a trajectory. Some trajectories represents the relative angular movements of the joints, and some represents the position of the joints in Cartesian coordinates, see Figure 2.12 for an example.

When given a kinematic model one trajectory is simply a mapping of the other through the kinematic model. However it makes comparisons of the different approaches time consuming. Hence the focus in this thesis has been laid on methods that presents results using stick figures, as they are easily comparable.

In this section the different methods considered for trajectory generation are presented.

2.4.1 On-line Trajectory from LIPM

It has been shown in recent years that the LIPM can be used for model predictive trajectory generation. By predicting the motion of a LIPM placed between the ZMP and the CoM, robots have successfully been manipulated to walk. The LIPM is used to predict how the robot falls, and

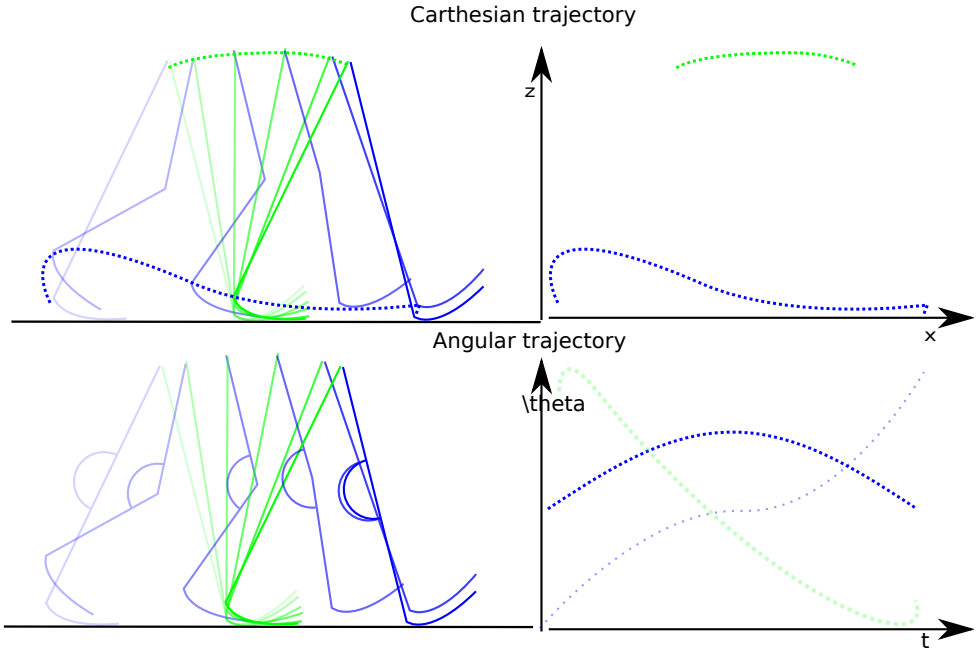


Figure 2.12: The same gait, provided as relative angles, or Cartesian coordinates. Not exactly comparable if the kinematic model parameters are not provided. Other known trajectory forms are in absolute angles on joints, i.e. relative to world axes, and not to parent link.

where to place the feet to sustain the forward flowing motion, and keep the robot dynamically stabilized.

This method have been applied on the virtual AAU-BOT1 successfully, but it has not, as far as this group is aware, been applied to anthromorphic gaits yet, even so it has been applied to create other dynamic and quasi static gaits, with small stride lengths. The method has especially been powerful for robots that have been designed to follow lines with curvatures, or avoid obstacles, and thus are forced to change direction.

2.4.2 Offline Trajectory Using Passive Dynamics

An approach which is sometimes used is to take the models from passive dynamics, and synthesize a trajectory for the active robot to track. The resulting motion can be very anthromorphic, as has been shown by multiple passive dynamics walkers.

One of the greatest advantages of using passive dynamics models for trajectory synthesis, is that the trajectory is by definition a trajectory which guarantees a low energy consumption for the robot. This is because the passive dynamics walker robot moves down a slope without the addition of electrical power. Hence the power needed to walk at a level surface can be made very small.

Also a positive side effect of using this type of synthesis is that the system dynamics are already used in the synthesis, and thus it can be used for determining a nominal trajectory for the robot that can be utilized as a feed-forward linearization of the walking controller. This guarantees that if the model used for generation is sufficiently good, and that errors during walking is sufficiently small, a linear controller can maintain stability infinitely.

One of the biggest problems in using this type of offline generated trajectory is that it is difficult to guarantee stability for larger disturbances, and difficult to determine when and where to add torque for actually achieving the level or uphill walking. However it allows for cyclic repetition, and thus it can be analyzed with regard to stability by means of Poincare first return maps and similar techniques.

AAU-BOT1 is not build for plastic impacts, and hence one of the vital assumptions about heel-strike used in passive dynamics analysis must be softened and compensated if such a trajectory is used.

2.4.3 Trajectory from Phase Locked Loop Analogy

The PDBR's walks in a cyclic anthromorphic way, however when looking at the joint angles only, bipedal gaits can be approximated with a Phase Locked Loop circuit. The benefits of this analogy is that it is possible to determine the initial conditions needed for the cycle to stabilize. Also it is a model which is familiar and has been investigated with regards to stability for a while.

The Phase Locked Loop (PLL) circuits even has the bifurcation properties found in PDBR's. This can be used to identify if a walker is within a bifurcated gait, or within an unstable gait. If within range of a bifurcated gait, then decreasing viscous friction will cause a gait with longer steps. In fact it can be shown through Pointcaré analysis of a Phase Locked Loop circuit that it is possible to force a bifurcated gait into a nominal gait without dropping stability. Hence this simplified model can very well be implemented to create an on-line compensated trajectory. PLL is however not shown in experiments with a real robot yet.

2.4.4 On-line Central Pattern Generator

In the human nervous system the Central Pattern Generator can be thought of as a system that converts a simple motion reference to a more detailed trajectory, using feedback from the muscle groups.

It is responsible for converting relatively simple trajectories like "move hand forward this fast" to "bend elbow shoulder and wrist using these muscles and compensate for the forces in this manner". In humans walking it actually seems that the Central Pattern Generators can maintain walking simply pulsed by the brain in regular intervals.

Hence it should be possible to correlate some of these pulses with sensory feedback, and then mimic the behavior of the Central Pattern Generators by using some simplified models, to maintain walking.

A simple model could be some linear model, that is optimized to produce outputs which are similar to those recorded by a human that walks, and is excited with different disturbances. i.e a small deviation of ground height.

2.4.5 On-line Trajectory State Machine

Inspired from the different ways the human gait can be cut up in phases (see Section 1.5), it is suggested by this group that a state machine can be set up that mimics these phases, and predicts the torques necessary to mimic the trajectory using a reduced model. This approach is somewhat similar to the concept of the Central Pattern Generators.

The concept presented is to calculate the necessary motions for the next phase using a non-linear model, to get the exit conditions for the current state, and then predict a trajectory that enables the robot to match those exit conditions.

When each state does this they create a short term trajectory, or pattern, which recovers the robot from current deviations from a desired movement. The benefits are that the global reference can be changed each time a state is shifted, and that by dividing the task into states, the calculations can be more task specific, and thus less complex.

2.4.6 Replaying Motion Captured Human Trajectory

A simple approach which have already been shown in simulation to be promising, is to take the initial moCap data, that was used for designing the robot, and then apply it as reference. This has been done in simulation by the designing group, and it is reported that without balance control the robot was able to walk in simulation. It was done using PID regulators for the joints, and the angles of the joints as reference. It was thus not concerning the non-linear effects as a control problem, but assumed that they could be neglected.

Even without compensating for the non-linear effects, the robot walked anthromorphically in simulation, however this method has some unfortunate properties. Using this method implies that some other controller handles disturbances, and regains balance if it is gradually lost, due to long term problems in maintaining the gait.

The method was never initialized in simulation, and thus only the stability of the walking sequence has been tested. Hence some initial reference is needed, to start walking from a stand still.

The trajectory is optimized to the human body, and thus not to a rigid robot, which not only is made of metal, but also has another weight distribution. Hence the torques needed to track the trajectory is naturally higher than a trajectory optimized for the robot would create.

The robot would only be able to walk at the recorded speed, and at an even floor. Which is a problem for most offline generated trajectories, but in this case it becomes especially a problem as the trajectory is prerecorded, and thus not easily modifiable. A synthesized trajectory can be recalculated, or modified by utilizing the model, allowing the on-line controller to replace parts of the trajectory with alternates that has been pre-calculated to stabilize some known balance problems.

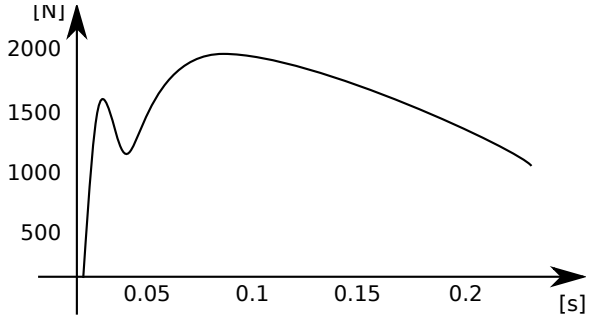


Figure 2.13: Force curve at heel impact, and stance, obtained by force plates on the ground with walking humans. Redrawn from [5].

2.5 Trajectory Challenges

When using simplified models to describe the robots trajectory, it is tempting to use the behavior of the passive walker models as reference. The passive dynamic walkers presents a functional method for achieving dynamic walking which is quite human-like, but still differences between the passive walker gait and the human gait does exist.

Some of the differences in the gaits are irrelevant for this project, as they arises due to non-human mass distribution found on the robot. However a few differences are not irrelevant, as the assumptions made when dealing with a passive dynamics model may not be true for AAU-BOT1.

2.5.1 Impact forces at Heel Strike

A big difference between the human gait, and passive dynamics gait, is the way Heel-strike is carried out. The passive walkers has a curved foot to guide the motion of the stance leg simply because it has no ankles, the combination of this design, and the relative low weight, ensures that the robot can endure the effect of approaching the plastic impact.

Another big difference is that the knee, on the impacting leg of a passive dynamics walker, is locked at heel-strike. This also approaches a completely inelastic (or plastic) impact, where the momentum of the walker is preserved, and the striking foot stays fixed on the ground.

Following this method for walking, and applying it on AAU-BOT1 can result in a great amount of force being transferred to the heel, ankle and knee joints on the robot, which it was never designed for. Humans face the same problem, where the knee especially is not build for the repeated translatory forces. Humans handles this problem with both passive and active shock absorption techniques[5].

The active shock absorbers used by humans includes joint positioning and muscle activity. The passive shock absorbers include synovial fluid, heel pad, muscle fibers, and similar body parts near the joints. The passive shock absorbers cannot be implemented on AAU-BOT1 without redesigning the joints. Hence all the impact force must be absorbed by active control.

Humans relax the muscles in the impacting leg and foot just before impact [36], and after impact the muscles are actuated to control the motion. This results in a force development at the foot as shown in Figure 2.13, where the first peak is due to the collision, and the second peak is actively generated by the muscles, to ensure that the stance is progressing as supposed to.

Notice here that the impact force is relatively small compared to the forces which later arises during the stance phase.

To create a similar concept for the robot, and thus improving the trajectory generated by

a passive walker model, such that it can be used for walking with AAU-BOT1, is a matter of elongating the impact pulse in time, and thus reduce the maximum force at the peak.

2.5.1.1 Absorbing the Impact Forces

Basically some of the force can be absorbed by pre-flexing the knee and ankle joints just before impact, such that the links does not form a straight line. This ensures that some of the force of the impact is absorbed as a deformation of the leg (rotation of the joints). Attaching rotational springs to the joint (or simply P controllers) to sustain the angle of the leg, will allow it to passively absorb some of the force, and when necessary the spring (P controller) can correct the posture of the leg after collision.

The pre-flexion can either be done by setting some constant joint angles as reference, or by applying a more advanced Model Predictive Control (MPC) scheme, the advantages of the more advanced solution is that the pre-flexion angles can be adjusted depending on the impact speed, to reduce the relative impact of the leg with the ground.

2.5.1.2 Calculating the Impact Forces

The problem about impact force calculation is that classical physics define impact as a sudden change in velocity due to a collision. This results in an infinitely short, infinitely high force peak.

In [30], a model of the ground was implemented based on springs. Such a ground model is interesting for impact modelling, as the impact with an ideal spring does not occur instantaneous, but instead a duration of time passes while the spring is compressed, and during this time the force provided by the spring grows from zero to maximum. Hence it produces a finite value, and can be calculated.

In this thesis a similar type of model is implemented in Open Dynamic Engine (openDE) and in Simulink to simulate the results of the impact. The spring constant was to be found through experiments, the results of which is presented in Appendix E.2. For a derivation of the impact model see Appendix D. It is noteworthy that in experiments, with controlled impacts the sensors only registered a spike with a duration of 1 or 2 samples. This means that the duration of the pulse is less than 4 milliseconds, which makes it very hard to absorb the energy by means of reacting to it in a controller. Also for such a short duration the time constants of the FTS amplifiers, and latency of the RS486 come in to play, and it is hard to guarantee that a reaction performed by a controller actually elongates the pulse, instead of just moving the robot unnecessarily after the pulse has already been absorbed by the robot mechanics.

After the experiments where carried out, it was decided not to handle impact by reactions. While it should be possible to do model prediction on the foots path to decrease impact speeds, this particular task should already be handled by a trajectory tracking, if the foots path is defined as positions, instead of joint angles.

2.5.2 Maintaining Balance of the Robot During Walk.

While the robot is walking it must avoid falling, or more strictly speaking, it must only fall in the desired direction. As shown by PDBR's the art of walking anthromorphically is the act of sustaining a forward falling motion, by lifting it up using the stance leg.

If a robot falls too fast, it cannot maintain the fall, and instead within a few steps lies on the ground. If the robot falls too slowly, it might not tilt over and land flat on the ground, but it will neither be able to move forward, and thus stands still. A general approach to balance keeping on biped robots is to track the trajectories of the balance points. The balance points where presented in Section 1.4. The tracking of the points can be done in multiple ways.

It is interesting that maintaining the balance using balance points can also be used while standing still. However, as AAU-BOT1 is able to stand without regulating the balance, it is unnecessary to implement this feature. The problem of maintaining balance if somehow disturbed is though interesting, and more can be found on this subject in Appendix F.

Another and much different approach to balance keeping, is to ignore balance as a continuous problem, and instead consider the CoM as an object which can be thrown ballisticly into an inverted pendulum mode around a foot. This property is what the PDBR's use to maintain balance in 3D. The regulation scheme used on PDBR's to dampen the effect of disturbances, is simply to guide the pendulum motion by curving the feet so that it tends to naturally follow the pre-calculated path.

The latter approach can be recreated in software, by using model prediction for generating the curvatures, the benefit is that the curvatures can then be manipulated to suit more than one walking speed.

AAU-BOT1 does not have the ability to change the foots shape, but a similar effect can be achieved by rotating the ankle joints, to track the desired inverted pendulum motions.

Depending on the choice of balancing strategy the stability analysis of the robot may change rapidly. The reason being that the walking biped system is not easily defined as stable or unstable.

2.5.3 Stability Analysis of a Walking Biped Robot

As mentioned in the prologue, controllers are expected to force a plant to track a referenced output value. Control theory is generally about analyzing if the plant will naturally converge towards the reference, and when it doesn't, what that can be done to it by means of applying control signals, to make sure that it will converge towards it.

It is the engineers task to determine a control law that ensures convergence, and then off course, determine how fast it can be forced to converge using that law, without turning convergence into divergence.

2.5.3.1 Linearizing Approach to Stability Analysis

Linear Time Invariant (LTI) systems, are the control engineers favorite plants, especially if all states can be observed and controlled. They are linear, so the plant can be modelled using a linear model. This is favorable, as many control tools exists for analyzing and controlling, and even optimizing a controller for a linear system.

LTI systems can in general be put on the state space form

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned}$$

and when that is the case, it is sufficient to determine if

$$\dot{x} = (A + BK)x$$

is Hurwitz.

It has the effect that stability can be achieved by choosing K such that the eigenvalues for $(A + BK)$ are all in the open left-half plane. This last statement is also applicable if the system has uncontrollable if those states are already stabilizing.

Nonlinear systems, such as pendulums or boilers are harder to control than linear systems, this is due to the fact that they are allowed to change the relationship between the states depending

on the values of the states themselves. Or more generally, a linear system has some relationship between the states that is defined by gains, but non-linear systems have the state relationships defined by some functions.

For illustration consider a pendulum, which is a classic example of non-linearity, the functions describing the relationships are all trigonometric.

Handling non-linear systems is often done by linearizing the, and then ensure local stability, by determining suitable eigen values. For the pendulum case, this can be done simply by stating that

$$\begin{aligned} sin(\theta) &\approx \theta \\ cos(\theta) &\approx 1 \end{aligned}$$

and then apply this in the equations of motion instead of the trigonometric functions.

Knowing how fast the non-linear equations deviates from the linear equations allows for determination of the Region of Attraction that defines how big an area in the state space that the controller can stabilize.

This method is however best suited for systems that are designed to converge towards a given reference, and then stabilize it there, with an unspecified time frame. The entire dynamic gait of a biped robot is characterized by state evolutions that are never stable, and thus are hard to describe as a desired convergence point in state space.

The x,$\dot{x}$ and even $\ddot{x}$ all changes rapidly during a step, and sometimes discretely or just non-linearly during the gait cycle, this is even true without concerning if the states are chosen as relative angles, or as positions of the joints. Hence the linearization and the following analysis of eigenvalues of a gait tracking controller provides little information on the gaits stability overall, but may only provide some information on parts of the gait.

2.5.3.2 Lyapunov Stability Analysis

Lyapunov stability analysis is widely accepted as a proper way to determine if a controller stabilizes a non-linear system.

A nonlinear system description is expected to be on the form

$$\begin{aligned} \dot{x} &= f(x,t) \\ x(t_0) &= x_0 \end{aligned}$$

Lyapunov stability states that a point $x* \in U \subset \mathcal{R}^{\mathrm{n}}$ is an equilibrium point, if

$$f(x*,t) = 0 \, for \, all \, t \geq 0$$

The general form of bipedal walking, consists of both continuous and discrete elements. This can generally be written as an Euler Lagrange equation for each continuously described motion, and a transition law. This takes the form

$$\begin{aligned} M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = B(q) \cdot u \quad & for \quad q \notin \Gamma_{-}^{(i)} \\ F^{(i)} : \Gamma_{-}^{(i)} \rightarrow \Gamma_{+}^{(i)} \quad & for \quad q \in \Gamma_{-}^{(i)}, \, i = 1, \ldots, Nd \end{aligned} \tag{2.2}$$

Where:
Nd is the number of possible discrete jumps due to instantaneous state updates.

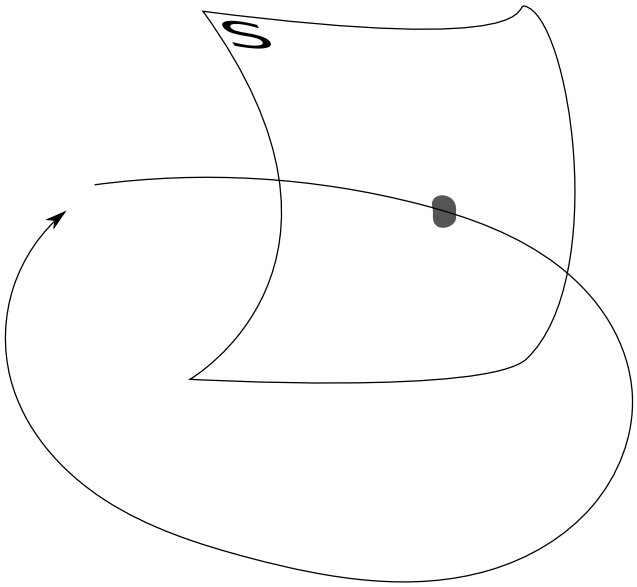


Figure 2.14: The flow of a state is shown as a curve, the transversal surface $\mathcal{S}$ is passed one time for each periodic cycle of the state, the position of the passing reveals if the cycle is attractive, for the given initial conditions.

$\Gamma_{-}^{(i)}$ is the switching surface, or manifold, of the continuous time solutions, this defines the area which triggers a discrete jump.
$F^{(i)}$ is the update law that governs the discrete jump, it determines what happens to the states at the time of switching.

The Equation 2.2 takes the form of a hybrid system. The result of this definition is that the continuous equations of motion are only described on finite time intervals.

If the system should be stable in the sense of Lyapunov, the system should be investigated for all $t \geq t_0$, and it should be determined if the distance to some equilibrium point grows beyond some defined distance. However, This would require the entire gait cycle, including all the jumps is within the boundary, and this would also include alot of options which are in fact unstable.

Further more, for systems as the walking Biped the asymptotic stability is undefined, and the system has only Lipshcits locally between jumps. Hence local stability can only be investigated in the continuous areas. This could make sense if the target for the controller was to stand still.

2.5.3.3 Pointcaré First Return map

It is assumed that a periodical flow of the states occurs. This allows for a surface $\mathcal{S}$ to be created which is transversal to the flow at some given point in the space $[q; \dot{q}] \in \mathcal{R}^2$ formed by the system in Equation 2.2. The surface is constructed as an $(2n-1)$dimensional surface and is called a Pointcaré section.

This means that a surface is created which the state will flow through at each periodical repetition, see Figure 2.14.

For a system like Equation 2.2, a popular and possible choice of $\mathcal{S}$ is one of the switching surfaces $\{\Gamma_{+}^{(i)}, \Gamma_{-}^{(i)}\}$, as they are well defined in the state space, See Figure 2.15.

When only studying the system at a small region of $\mathcal{S}_0 \subset \mathcal{S}$, and at the plane intersection

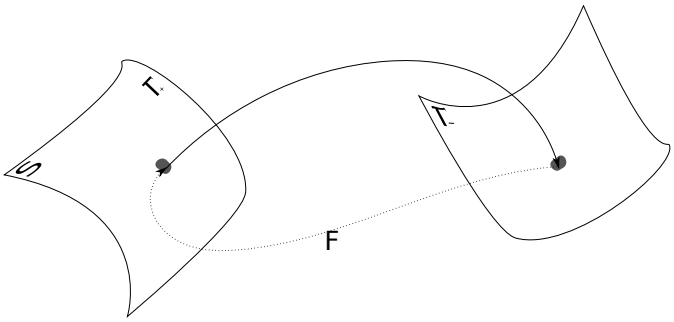


Figure 2.15: Popular choice of the Pointcaré surface when regarding hybrid systems is one of the already defined switching surfaces, as it is usually known that a stable orbit will pass it.

time, the result will be a $(2n-1)$dimensional discrete-time system with the form

$$x \perp [k+1] = P(x \perp [k]), x \perp \in R^{2n-1}$$

Where:
P denotes the Pointcaré mapping $P : \mathcal{S}_0 \rightarrow \mathcal{S}$.

The Pointcaré map reveals if the flow of the state deviates diverges or converges towards a stable orbit. Each isolated periodic orbit, is called a limit cycle if nearby orbits are either attracted to it or repelled from it. A limit cycle in it self can thus be either a stable limit cycle, or an unstable limit cycle.

The linearized version of the Pointcaré map is denoted dP and can be used to verify exponentially orbital stability. It is with a linearized Pointcaré map much like a with a linear system that analysis of the eigenvalues reveals stability properties of the gait. In this case however, local stability is present if the eigenvalues resides within the unit circle.

The linearized Pointcaré map is generally expressed as

$$\begin{aligned} \delta x \perp [k+1] &= \left.\frac{dP}{dx \perp}\right|_{x\perp = x^*_\perp} \cdot \delta x \perp [k] \\ d^- x_\perp &= x_\perp - x^*_\perp \end{aligned}$$

Unfortunately an analytical approach to calculating dP is in general considered impossible, and if possible the complexity increases with the number of states. Hence stability analysis with the use of Pointcaré maps is usually carried out with numerical simulations. i.e. testing the system behavior against a large set of initial values.

As a result the choice of trajectories for a robot is often determined offline, by search algorithms, that then verify the stability of a proposed gait pattern with the eigen values of a linearized Pointcaré map. The innermost eigenvalue determines the rate of convergence towards the stable limit cycle.

The Region of attraction as analyzed in a Pointcaré map gives only information on how stable the limit cycle is at that particular surface in the $[q; \dot{q}] \in \mathcal{R}^2$ space. To gain knowledge that spans the entire gait, the continuous parts between the switching surfaces are also often investigated, this is done by applying the moving Pointcaré section.

The surface $\mathcal{S}$ is considered a function of time, and is allowed to move with the flow of the state.

$$\{\mathcal{S}(t)\}_{t\in o,T}$$

The moving Pointcaré map represents a continuous family of the $(2n-1)$ dimensional surfaces, which is the abstraction of moving the Pointcaré section, along a limit cycle to identify which areas of the orbit that may be less attractive than others.

Actually calculating the moving Pointcaré map requires linearization along the stable limit cycle and is non-trivial. However it is possible, and with this tools the update laws that governs the discrete jumps can, also be included if they can be linearized. The result of this analysis tool is that it can be applied over the entire gait, if a stable limit cycle is known, to verify local stability. The notion of a moving Pointcaré map on a hybrid system is illustrated in Figure .....

The Pointcaré mapping tools are very powerful tools for analyzing periodic gaits, however the reverse calculation, which is to determine a gait directly from a desired Pointcaré map behavior is not yet possible. So the trajectory must still be found and tested for periodicity before the stability can be tested.

Further more the tool is not providing much information about what other limit cycles that exists. So bifurcations, and stable increase or decrease of speed is not determinable using the Pointcaré map technique. In fact the only situation that can be verified for robotic systems, is their ability to sustain a walk in simulations with infinite floors, or on a treadmill running with a constant speed.

2.6 Simulator Model

A simulator was created by previous groups that was considered sufficient for the task of achieving static walk. The simulator which was inherited from previous groups did not come with documentation concerning verification. An analysis of the model used for simulation showed that it contained various parameter errors, such as the location of the CoM of some of the limbs.

While these minor parameter errors could be easily corrected, the basic model structure was not verified either, and it was based on a method that originally was presented as novel. For this thesis authors it would have been preferred to have a simulator model that was verified against the hardware, or similar biped robots, should hardware not have been completed.

The errors found in the simulator implementation where:

- ▷ Erroneous Motor constant distributions.
  (This results in a linear gain error for the joints in the model, resulting in linear gain errors in the controllers)
- ▷ Location of CoM which sometimes is outside the geometry of the limbs it refers to.
  (gives all sorts of errors, such as the length of the linearized pendulums used for linearization and controller tuning.)
- ▷ The toe was not provided with a mass and inertia.
  (negligible error)
- ▷ Inertia matrices was reduced to principal axes without rotating the local coordinate systems.
  (primarily nonlinear errors, it should not affect the linear parts of the controller much)
- ▷ States was extracted directly from the simulator, and used for a controller, which again was used as verification of both. None of them was compared successfully with the real system. The extracted states cannot be obtained from the real system without the use of a sensor fusion algorithm. Hence this introduce yet a source for errors in debugging.

The untested, or undocumented assumptions where:

- ▷ Equal frictions seen from both sides of geared joints.
- ▷ Assumed similar values for joint frictions for all joints, even where different belts, gear ratios, number of motors, and motor constants are installed.
- ▷ The ground friction forces are assumed infinite, but is implemented by a horizontal spring, allowing the foot placement to be corrected. This allows the simulator to absorb torque impulses in the ground model, and thus reduce instability in the simulator.
- ▷ The two nonlinear dynamic models where linearly combined to determine the torques and forces, however the reliability of this model structure has not been tested in experiments with AAU-BOT1, and have only been tested very closely to the linearization point on Roberta. The tests conducted on Roberta does not fit experimental and model data very well, but where assumed to be good enough for the purpose of that robot.
- ▷ Masses where extracted from Solidworks, and the sum of the masses where compared to the AAU-BOT1 weight. A significant difference was found and the difference was distributed on very few links by guessing. The results was never tested in experiments and the friction parameters was afterward estimated using the dynamic model with the assumed masses.

Instead of fixing the errors, and test the simulator to verify that the untested assumptions could be made, this group decided do build a new AAU-BOT1 simulator in Simulink, which is not based on the same assumptions. If this simulator introduces new errors, it is not in any way better, and thus the simulator behavior must be verified through testing.

Requirements was set up for the new simulator.

1. The new simulator should not inherit any parameters from the previous simulation model, without careful considerations, to avoid repeating errors.

2. The dynamics of the system should be handled in a way which allows for closed link loops to be solved, such that it can handle "no foot contact" ,"single stance" and "double stance", without switching the model.

3. The parameters for the simulator must be estimated through experiments to minimize behavior differences between the simulator and the actual robot.

The simulator must also recreate the dynamic behavior with as many DoFs as found on AAU-BOT1, this means that where a joint exists on AAU-BOT1 the simulator should also have a joint. To get the kinematics and model structure as accurate as possible, without disassembling the robot, the existing Solidworks assembly can be used to extract data about the robots skeleton and geometry.

Often in control theory, when modelling the dynamics, a kinematic model is extended to include the dynamics. Such models are often based on the Denavit Hartenberg notation which basically relates different Cartesian coordinate frames to each other by applying 4x4 transformation matrices. The basic property of this type of model is that the kinetics are set up as a hierarchy, where the children transfers the torques and forces to the parents. A model like this can be created when assuming that the final parent in the chain is always anchored in the global frame. This model type is appropriate for many robotic systems, as many robotics are anchored at the ground. For walking robots however the assumption that the feet are anchored to the ground when they touch it, is not always true, modelling this effect can be troublesome.

Another way of modelling physical systems can be to consider a mechanical system as a set of particles. The particles experience torques and forces, when they are applied to them, and they effect other particles when geometric constraints requires it. Particles form the basics of rigid body dynamics.

The mechanical engineers that designed the robot had designed and implemented a simulation model in a rigid body topology which was able to handle the requirement of solving closed linked loops, they dubbed this forward dynamics modelling. This modelling technique is nonlinear, and can contain model components which cannot be linearized. The advantages is though that it is a fairly simple topology, that is easy to relate to the robot structure.

ODE Solvers for Rigid Body Dynamics

For this project two different rigid body dynamics engines where considered as they basically use the same structure as the simulator created by [27].

The faster simulator is the openDE environment, and includes the use of the build-in collision detection, for ground impact detection.

The slower but also MATLAB compatible engine can be set up in the Simulink SimMechanics toolbox. Here collision detection must be implemented manually, and external forces can only be applied to the rigid bodies by means of modelling the actuators and ground effects manually.

The biggest difference is though that openDE applies a static fixed step value for the sample time, where SimMechanics can use a time varying numerical integrator. which allows the

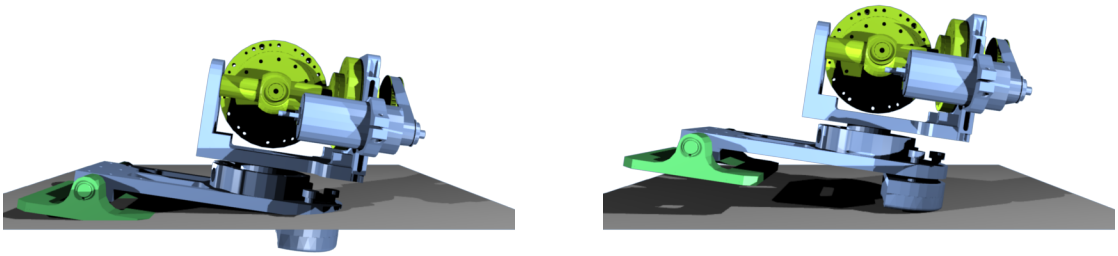


Figure 2.16: The left figure illustrates the penetration depth result of openDE's collision detection, while the right illustrates the result of using zero crossing for collision detection. When ground normal forces are applied as a spring, then using large spring constants can in openDE result in an extremely high force due to the large penetration depth, and propel the robots upwards.

integrator to apply forces at the time of impact, instead of at the time of the simulation step. See Figure 2.16. Furthermore SimMechanics is designed to be used by control engineers, and is shipped with a set of "sensor" and "actuator" analogies.

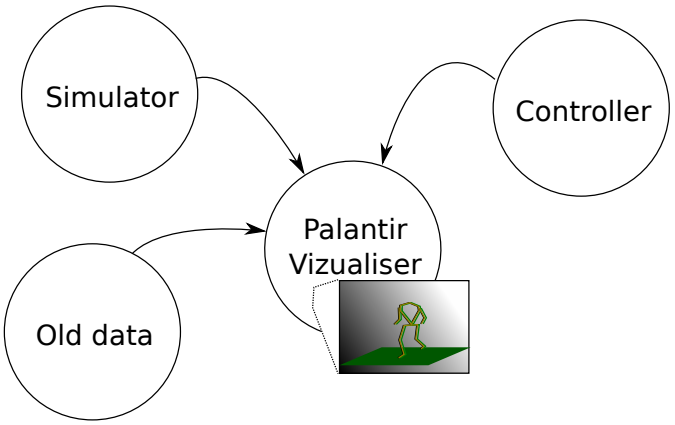


Figure 2.17: Concept for a visualization server.

2.7 Palantir, 3D Visualization Server

To be able to understand the dynamics of a system that operates in three dimensions, it is a great advantage to be able to interpret the data in a three dimensional environment, as the same data can be seen from many angles.

SimMechanics comes with a build in visualization, that can be utilized when using the simulator, it does however slow down the simulation considerably. As the simulator is already expected to be operating much slower than real time, this is inconvenient. Another and more critical problem with the build in visualization tool, is that it cannot be used for playing back recorded data.

Consider the case where the simulator is compared against the physical plant, the ability to playback recorded sensor data in the same environment as the simulation data is shown, is vital to detect small or time dependent deviations.

Even without this comparison it is an advantage to be able to replay sections of the data, and perhaps freeze frame, or watch it in slow-motion, as it allows for reviewing the experiment from many angles, and with different focus.

A visualizer should not load the computer which is running the simulator, and it should not load a controller either when the data is recorded. This is off course not possible if the data should be viewed in real time, but a very low load can be achieved by deploying the visualizer on another computer, and then stream the data using some low priority network protocol such as User Datagram Protocol (UDP). A network topology also has the advantage that any simulator structure can be programmed to stream the data to the visualizer, providing a similar interface for comparing data, if simulators or the data structures are replaced later on. This even provides great advantages over simple file logging, as writing files is slower and uses dynamically memory which might make the controller voilate sample times as you run out of disk space or memory. UDP transmission on the other hand would simply drop packages, and thus giving a predictable small constant delay to the loop.

2.8 Concept Summary

Multiple posture controllers has been considered, and it has been decided to implement a PID controller. This has been chosen to have a functional system to test against, and to have a basis for development of the MPG. It was in the research found that a MIMO controller is needed if the performance should be increased. For the best case, a computed torque controller is suggested, but it is not developed in this thesis.

The biped is in it's nature a hybrid system, and many suggestions has been given to make it walk. Many of them states that a trajectory can be formed, and then tracked. This however leaves balancing as a separate task, that must be somehow put on top of the trajectory. This approach was investigated for this thesis, where it was attempted to generate a nominal trajectory, from the basis of a controlled complex non-linear simulated model of the robot.

However this was initiated late in the project, and it was not feasible to provide a stable trajectory within the time limits of the project using that method. In Section 4.1, the problem of finding a stable trajectory for hybrid system (simple compass walker) is investigated further.

To determine a nominal attractive trajectory some search algorithm is needed, and it was found that the problem of determining a stable gait in itself has a closed loop logic. Either there is a trajectory where the stability can be analyzed, or there is a trajectory that falls instead of walking, so no stability analysis can be done, and the reason why it does not walk is hard to deduce, as there is not much new knowledge gained from a stability analysis of the unstable gait. Often more knowledge was obtainable by visualizing the robot.

Instead of this approach, the function of the human Motor Pattern Generators (MPG's) are analyzed. And it is found that they correlate in behavior with the phases of the walk. This is used to create Virtual MPG's, that generate the next part of the gait pattern, by a combination of reduced model prediction, and event handling.

The basic concept for the MPG is that it is triggered by the discrete events on the system. It is to be implemented as a state machine, that shifts the pattern generator system, and sometimes even the posture controller, according to the event. The events are all triggered by the change in support phase, and states are all updated using update laws, which for instance does not allow discrete changes of angles on the robot.

As a result the MPG changes behavior depending on the support phases, and as such can be compared in principle to the function of the human MPG's.

Models are needed to create such a controller, so before the control design is presented, the models developed for this thesis is presented.

3

Modelling

"There are no physicists in the hottest parts of hell, because the existence of a "hottest part" implies a temperature difference, and any marginally competent physicist would immediately use this to run a heat engine and make some other part of hell comfortably cool. This is obviously impossible."
—Richard Davisson

3.1 Rigid Body Model for the Simulator.

This section presents the basic principles that powers rigid body dynamic models, which is the physics modelling system used in the SimMechanics environment.

3.1.1 Equations of Motion for Particles

The basic Newtonian laws of physics applies to particles. If the position of a particle is described by a vector which varies with time $\vec{x}(t)$, the velocity is given as $\vec{v}(t) = \dot{\vec{x}}(t)$, and the acceleration is $\vec{a}(t) = \dot{\vec{v}}(t)$, and the mass is a scalar. As a result the relationship between an applied force and the resulting motion can be expressed as vectors of the form $\vec{F}(t) = \dot{\vec{a}}(t) \cdot m$. The motions of a particle is then the result of the net sum of all the forces.

$$\vec{F}(t) = \sum_i \vec{F}_i(t)$$

And the equations of linear motion can be expressed as Equation 3.1 and 3.2

$$\dot{\vec{x}}(t) = \vec{v}(t) \tag{3.1}$$

$$\vec{a}(t) = \frac{\vec{F}(t)}{m} \tag{3.2}$$

The principle of a lever is a well known phenomena, in three dimensions the torques which arise on the particle due to the force applied off center, is calculable as in Equation 3.3

$$\vec{\tau}(t) = \vec{r}(t) \times \vec{F}(t) \tag{3.3}$$

The vector $\vec{\tau}$ describes the direction of the axis which the torque is applied about, and the length $|\vec{\tau}|$ is the power of the torque.

Similar to $\vec{v}(t)$, $\vec{\omega}(t)$ is the rotational velocity vector, however the direction describes the axis of rotation, and the length is the angular velocity, in the same analogy $\vec{\alpha}(t)$ is the angular acceleration. It is noticeable that if $\vec{\alpha}(t)$ at some point in time does not have the same direction as $\vec{\omega}(t)$,then $\vec{\omega}(t)$ changes direction, and will continue to do so if not an opposite angular acceleration is applied at some point.

For a particle the geometry is quite simple (it is an infinitesimal small sphere), and as a result the relationship between torque and angular acceleration is $\tau(t) = I \cdot \alpha(t)$. Where $I = m \cdot radius^2$, and as the particle is undeformable, this is just a scalar.

3.1.1.1 Simulation Notes

As is the case for classic physics, a force will change the velocity of a particle, as it creates an acceleration. Due to the change in velocity it also changes the linear momentum $\vec{p}(t)$. A change in momentum in general is dubbed an impulse.

$$\vec{J} = \int \vec{F} dt$$

If a constant force is applied for a known duration of time (Δt) the impulse reduces to $\vec{J} = \vec{F}\Delta t = \Delta p$. Which is useful for approximating $\vec{p}(t)$ when solving numerically. Similarly $\vec{\tau}\Delta t = I\Delta\vec{\omega} = \Delta\vec{L}$, where $\vec{L}$ is the angular momentum.

3.1.2 Rigid Bodies

A rigid body is an undeformable body, which if spherical behaves like a single large particle.

More specifically a rigid body is a system of infinitely small particles distributed evenly within a geometric body, where each particle is constrained in motion by the geometry of the body. Hence for each of the particles, the basic laws of physics apply, and due to the geometric constraint, a particle is not allowed to move without moving all the other particles in the body such that the geometry is unchanged.

The mass of the body is the sum of all the particles masses, however if calculated as infinitely small particles, it is described by the density times the volume. ($m = \int_V \rho(\vec{x}) dV$) where the volume becomes an integral.

A result of the properties of a rigid body, the concept of CoM is very important. If a linear force is applied to the body exactly towards the CoM the result is just a linear movement of the body, but if the same force is given to the body in some direction which is not pointing directly towards the CoM, a linear motion, and a rotational motion is produced.

CoM is simply a vector, which locates where the masses of the rigid body are balanced out in all directions.

$$\vec{x_{CoM}} = \frac{1}{m} \sum_i m_i \vec{x}_i \tag{3.4}$$

or in the continues case

$$\vec{x_{CoM}} = \frac{1}{m} \int_V \rho(\vec{x}) \vec{x} dV \tag{3.5}$$

The motions of a rigid body can be determined separately for the translational, and rotational accelerations. It is though necessary to understand the concept of Inertia in 3d first.

Mass Moment of Inertia

In two dimensions the angular momentum is found as $L(t) = I \cdot \omega(t)$, where I is a scalar, but in three dimensions rotations are allowed in more than one axis. Depending on the geometry of the object, the object will behave differently when rotated about different axes. i.e. A long stick will behave differently depending on which axis is used for rotation, (consider longitudinal or perpendicular axes). Hence I has to depend on the geometry of the body.

The mass moment of inertia can be represented as 3x3 matrix.

$$I = \begin{bmatrix} I_{xx} & -I_{yx} & -I_{zx} \\ -I_{xy} & I_{yy} & -I_{zy} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \tag{3.6}$$

Where the diagonal elements are known as the *moment of inertia coefficients*:

$$\begin{aligned} I_{xx} &= \sum_i m_i(y_i^2 + z_i^2) \\ I_{yy} &= \sum_i m_i(x_i^2 + z_i^2) \\ I_{zz} &= \sum_i m_i(x_i^2 + y_i^2) \end{aligned}$$

and the remaining three elements are known as the *products of inertia*:

$$\begin{aligned} I_{xy} &= \sum_i m_i x_i y_i \\ I_{xz} &= \sum_i m_i x_i z_i \\ I_{yz} &= \sum_i m_i x_i y_i \end{aligned}$$

Off course for continuously defined bodies, such as spheres, the sum becomes an integral. In a body space the matrix I is static, in a world space it is time dependent, due to the rotation of the body over time. The world space matrix can be found by applying the rotational matrix R.

$$I_{world}(t) = R(t) I_{body} R(t)^T$$

The matrix I is always calculated for a rotation around a specific point, and it does not need to be CoM. However often I is calculated as seen from CoM, this is because the coefficients becomes smallest at CoM. Fortunately I can be found for any other point, if I_{CoM} is known, and a vector $\vec{d}$ describes the location of the point of rotation.

$$
\begin{aligned}
I_{xx} &= I_{CoMxx} + m(d_y^2 + d_z^2) \\
I_{yy} &= I_{CoMyy} + m(d_x^2 + d_z^2) \\
I_{zz} &= I_{CoMzz} + m(d_x^2 + d_y^2) \\
I_{xy} &= I_{CoMxy} + m d_x d_y \\
I_{xz} &= I_{CoMxz} + m d_x d_z \\
I_{yz} &= I_{CoMyz} + m d_y d_z
\end{aligned}
$$

The inertia matrix is symmetric, so it is always possible to find a new rotated coordinate system in which the inertia matrix is a diagonal matrix (the products of inertia are all zero). The procedure of finding such a diagonal inertia matrix is known as principal axis transformation, and the axes of the rotated coordinate systems are the principal axes. The principal axes will correspond to the axes of symmetry. Even for arbitrary shaped bodies principal axes can be found so that all products of inertia are zero.

Time Derivative of a Rotation Matrix

While it is quite easy to write that $\dot{R} = \frac{dR}{dt}$, it can be useful to recap how this is actually calculated, to be able to handle rigid bodies which is not oriented exactly as the world space.

$$\dot{R(t)} = skew[\omega(t)] \cdot R(t)$$

where the skew function defines a symmetric skew matrix:

$$skew[\omega(t)] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

3.1.2.1 Equations of Motion of a Rigid Body

The linear motion analogue is given by Newton's second law $\vec{F}(t) = m \cdot \vec{a}(t)$. For angular motion this is more complicated since the inertia tensor is not constant:

$$\vec{\tau}(t) = \vec{\omega}(t) \times [I(t) \cdot \vec{\omega}(t)] + I(t)\frac{d\vec{\omega}(t)}{dt}$$

Where $\vec{\omega}(t) \times [I(t) \cdot \vec{\omega}(t)]$ is the Coriolis term.

Combining linear and angular motion yields the so called Newton-Euler equations of rigid body motion:

$$
\begin{aligned}
\dot{\vec{x}} &= \vec{v} \\
\dot{\vec{v}} &= \frac{\vec{F}}{m} \\
\dot{R(t)} &= skew[\omega(t)] \cdot R(t) \\
\dot{\vec{\omega}} &= I^{-1}[\vec{\tau} - \vec{\omega} \times I\vec{\omega}]
\end{aligned}
$$

Now the reaction of a body to a force can be computed. If a force is acting on some point P, the CoM will be accelerated as if the force would act directly on the center of mass. In

addition, the force will cause a torque, which will start to rotate the body. The forces are all applied translational to CoM and all the forces are also converted to torques and applied to CoM. Accelerations are found using the equations of motion for a rigid body, and the velocity and position is a matter of integrating. In SimMechanics the numeric integrator used can be chosen in the Simulink dialogs.

3.1.2.2 Constraints

Constraints are rules like "The rigid body must not penetrate this surface", or "these two bodies are connected at this ball joint". The constraints are often describable by applying geometric analysis.

The task of a constraint solver is to "do something", such that the constraints are not violated. The position, the velocity, and the forces can all be changed from a constraint solver, however it is not considered a good idea to manipulate the position of a body if it can be avoided as it greatly reduces accuracy of the results if Euler or Runge-Kutta schemes is used[12].

In general constraints are usually solved by adding forces to bodies which otherwise would violate constraints, and then resolve their equations of motion, this is an iterative process, and can at worst be unsolvable if many bodies are involved. So constraints are often solved as a a least minimum square problem, to minimize errors with regards to constraints using forces.

The constraint solver is part of the SimScape package that contains SimMechanics for Simulink.

Rigid bodies are also used in the controller, though only for integrating a simple inverted 3D pendulum model, the constraint is solved there by updating the position directly. Thus another integrator is needed than the one provided with SimMechanics. See Sub Section 4.3.1.1 for more on this integrator.

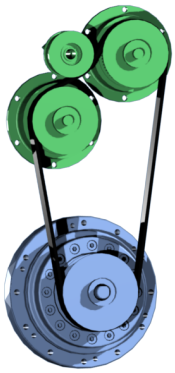


Figure 3.1: Gear setup, this is for one of the knees, two motors are connected to a belt with spur gears, the belt is connected to a harmonic drive gear.

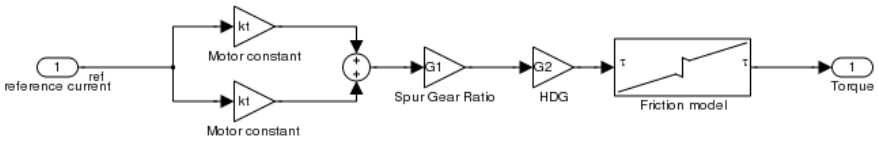


Figure 3.2: Simple linear gear model.

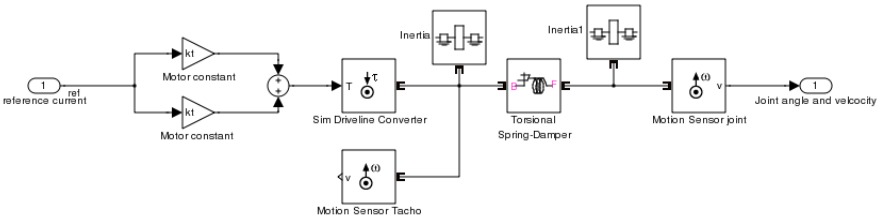


Figure 3.3: Simple linear gear model.

3.2 Gears and Friction Model

The gears used for transferring power from the DC-motor to the axle, is a complex mechanism that consists of a belt, two spur gears, and a HDG. See Figure 3.1.

Previous groups have suggested different models for this system focusing on different aspects of the gear configuration. The simplest model that has been suggested is a linear gear with a gear ratio 'G', and a classic friction model. Such a model is shown in Figure 3.2.

In comparison the most complex model that has been suggested has been created as a model that includes gear slack, by inserting a spring as the torque transfer mechanism, see Figure 3.3.

From the experiments conducted on the arm joints by [31], a classic friction model seemed appropriate, and from the experiments conducted on the left hip roll joint ([30]) the addition of the gear slack reduces the error further, when angles are studied from the tacho.

However the experiments conducted using both the new potentiometers, and the tachos for odiometry, indicates that the slack in fact is small, as the two seldom diverts. The effects previously explained as gear slack is present and synchronized on both sides of the gear axles,

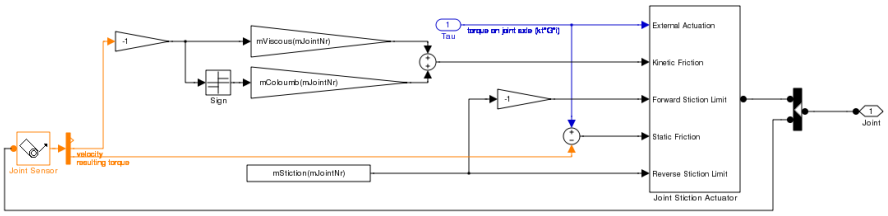


Figure 3.4: The friction model used in the SimMechanics simulator.

and should thus be related directly to frictions.

For most of the joints the gear slack is very small, and it is decided that gear slack can be neglected for all practical purposes. Thus the simple gear model previously suggested is chosen. It is however implemented in a new simulation environment, and the available model structure for the frictions are somewhat different, as joints can be locked and unlocked in the SimMechanics environment.

The chosen model is a state model, which locks the joint when the torques and angular velocities experienced by the joint drops below a threshold, similarly the only way to unlock the joint, is if the joint experiences a torque that is higher than the threshold.

When unlocked, the viscous coefficient and a Coulomb friction is applied, as shown in Figure 3.4. Notice that the threshold and the Coulomb friction is applied as [Nm], this is because it is assumed that translational loads to the joint does not cause the friction in the gears to change.

3.2.1 Belt Drive Gear Improvements.

Steps was later taken to reduce the gear slack further. The reason for this was not actually considerations about the gear slack, as much as it related to friction issues. Especially at the knee joints.

The knee was worse to model than on any other joint. However if the friction was modelled as a two state Coulomb friction, it seemed that the behavior could be replicated in the model. Interestingly enough the state shifts correlated with the amount of teeth on the motor shafts spur gear.

After consultation with a mechanic, it was decided to change the spur gears and the belt type, to a type with more, but smaller teeth, this should result in a less varying friction, which can be better modelled by a classic friction model. As a side effect the belts are available as a Kevlar reinforced belt, which is less elastic and stronger than the previously used steel reinforced belts.

This group recommends that all gears are changed with Kevlar belts when possible. However as this would take precious time at a late and critical part of this project, it was not done during the time span of this project. Instead belts was loosened, as some of them was tightened more than was necessary. Having the belts too tight increases friction between the belts and the gears. The loosening reduced the problems, and did not infer measurable gear slack, but it is still recommended that the belts are upgraded.

3.2.2 Harmonic Drive Gears

According to [38], nonlinear friction effects can be included to improve the gear model of the HDG, in fact several important characteristics of typical harmonic-drive behavior can be observed.

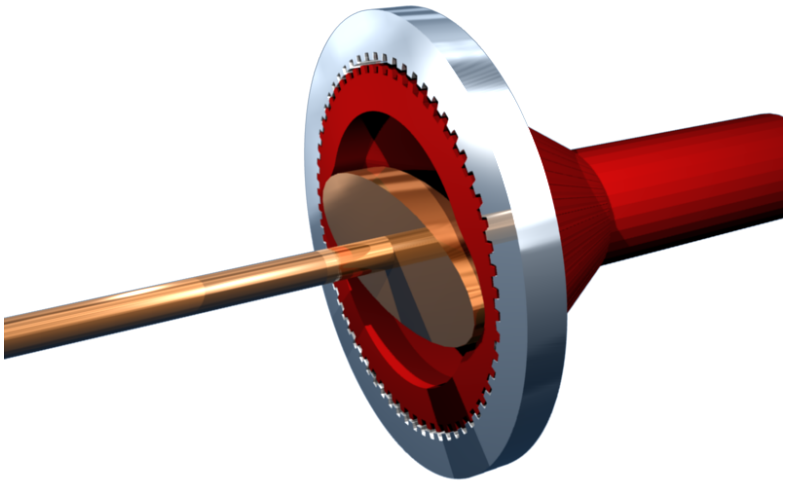


Figure 3.5: A cross section of a HDG, the copper wave generator is driven by the inner shaft, the red flexible spline is rotated as the teeth interacts with the steel ring. The flexible spline is expanded by the copper wave generator; as it rotates half a turn it forces the red shaft to move one teeth.

1. Since kinematic inaccuracies in the transmission cause velocity fluctuations which excite system resonance, a substantial portion of the operating range of each harmonic drive is contaminated by serious vibration.
2. Energy dissipation increases in these regions of resonance and hinders the increase in rotational velocity.
3. An unpredictable jump in velocity can often occur when the transmission harnesses enough energy to push through a system resonance.

This is due to the construction of the HDG.

An inner elliptical disk called a wave generator, is attached to the inner shaft, (the fast moving shaft). The outer shaft, (the slow shaft), is attached to a flexible spline, which have teeth, that are attached to the circular casing, when it rotates. The wave generator is rotated and thus stretches the spline, so that it connects with the circular casing at different points. This creates a wave like behavior on the spline, and due to the teeth, it forces the flexible spline to rotate. The gear ratio is determined by the number of teeth on the flexible spline, and on the circular casing.

A simple illustration can be seen in Figure 3.5.

There is friction between the two shafts as they are in contact with each other that will resemble classic friction, with a viscous and a Coulomb friction, but there is also a resonant friction as the flexible spline grabs the circular frames teeth. Further more the elasticity of the flexible spline gives dynamic torsion on the coupling between the shafts, and as a result a friction that is inverse proportional to speed. Hence friction components vary with both angles, and angular velocities, in a non-linear fashion.

These angle dependent properties are not modelled by a classic friction model.

G_{HDG}	Type		η_R	
100	14		7	
160	20		0	
120	20		7	
100	17		6	
120	17		8	

Table 3.1: Correction factors for the HDG used on the robot [%].

3.2.3 Efficiency of the Harmonic Drive Gears

The efficiency depicts how much of the applied torque that is transferred. The remaining torque is lost in the gear, and is usually lost due to friction.

The efficiency of the HDG gears has been estimated by [27] as

$$\eta = K \cdot (\eta_R + \eta_e)$$

Where:

$K = 0.3 \cdot log(V_\tau) + 1$

η_R is the HDG units rated torque.

η_e is the HDG units size, and ratio.

$V_\tau = \frac{\tau_{avg}}{\tau_N}$

τ_{avg} is the average torque on the output shaft.

τ_N is the rated torque at rated speed.

According to the HDG datasheets $\eta_R$is non-linearly temperature dependent, and nearly linearly speed dependent. This implies that the analogy of a viscous friction is somewhat appropriate if it is assumed that the temperature is kept constant at 20 ^{o}C.

From the datasheets the correction values η_e are provided as:

Which is quite small corrections, but also illustrates that, the linear approximation using a pure viscous friction might not be optimal, as the corrections done is speed dependent. It is expected that the corrections are small, and that the model in general is accurate enough without including this correction, and it is not included in the friction models.

When considering the angle dependent friction, there has not for this thesis been found a model which has been able to improve the prediction reasonably well.

3.2.4 Gear Ratios

The gear configuration on the robot is very joint dependent, all joints consists of a belt/ spur gear, and a HDG gear, and the applied model is a linear gain, using the gear ratios. The gear ratios are distributed on the robot as shown in Table 3.2. The combined gear ratio for a joint is simply the multiplied gear constants for the joint.

HDG Gear Ratio	Belt Drive gear Ratio	Number of motors	Joint name
100	44/22	1	Ankle Roll
160	35/18	2	Ankle Pitch
100	52/39	2	Knee
120	57/28	1	Hip Pitch
120	69/30	2	Hip Roll
100	40/19	1	Hip Yaw
100	46/16	1	Pelvis Yaw
120	57/19	1	Waist Pitch
100	57/19	1	Waist Roll
100	1.25	1	Shoulder

Table 3.2: Location of gears, the gear ratio G is the multiplum of the two provided gear ratios.

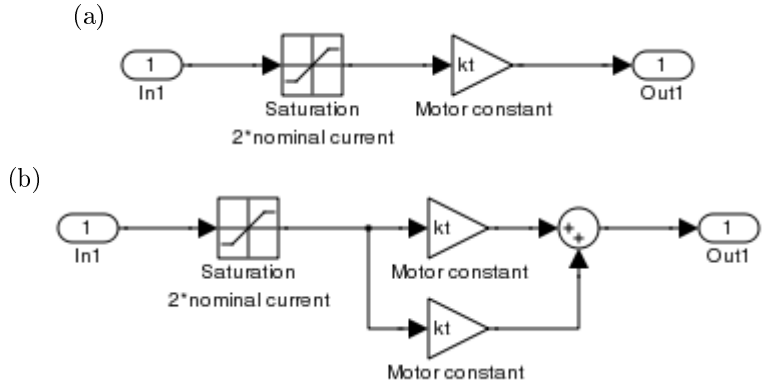


Figure 3.6: Motor models are assumed linear, but saturation is applied in the simulator, and in the controller, to avoid the motors to be overloaded by more than twice the nominal currents.

3.3 DC Motor Model

The motors used are Maxon DC Motors, of 60, 90 and 150 Watts.

The motors are brushless, and are powered with a 60V DC supply. For this project, the motors are assumed to have a time constant much smaller than any other dynamic part on the system, and are as a result linearized to a motor constant. Maxon specifies a nominal current for each motor type, that should not be violated. However they also specify for how long time it is allowed to violate the nominal current without overheating the motor. Some of the joints on the robot provides so much stiction to the system, that the motors have to be fed close to the nominal current to generate the torque needed to actuate, hence it is decided to allow this for a short duration of time.

The motor model is thus a saturated gain, as shown in Figure 3.6(a), joints with two motors are considered parallel torque providers, where the torque is simply summed before the gears, as shown in Figure 3.6(b).

The motor parameters is given in Table 3.3

The distribution of motors is provided in Table

motor type	motor constant (kt) [Nm/A]	Nominal current [A]
60 watt	$53.8 \cdot 10^{-3}$	1.72
90 watt	$62.2 \cdot 10^{-3}$	1.63
150 watt	$60.3 \cdot 10^{-3}$	3.12

Table 3.3: Motor Parameters found in the datasheet.

motor type	Number of motors	Joint name
60 watt	1	Ankle Roll
150 watt	2	Ankle Pitch
150 watt	2	Knee
150 watt	1	Hip Pitch
150 watt	2	Hip Roll
90 watt	1	Hip Yaw
60 watt	1	Pelvis Yaw
150 watt	1	Waist Pitch
90 watt	1	Waist Roll
60 watt	1	Shoulder

Table 3.4: Motors distributed on the robot, notice that left and right side are symmetric, so the distinction is not provided in this table.

3.3.1 EPOS

The EPOS are the means of communicating with the motors from the computer. However as they are run in a current control mode, they are not investigated with regards to dynamics. It is noted that they infer a small delay due to the communication over CAN, and that they infer a small noise to the currents applied, as they are not ideal current regulators. However they are sufficiently close to ideal to neglect this effect.

When this has been mentioned it is noteworthy that there is placed 5 separate CAN networks on AAU-BOT1, and that communication with the EPOS has a low latency as a consequence of this. The name convention on the CAN for communicating with the EPOS is illustrated in Figure 3.7.

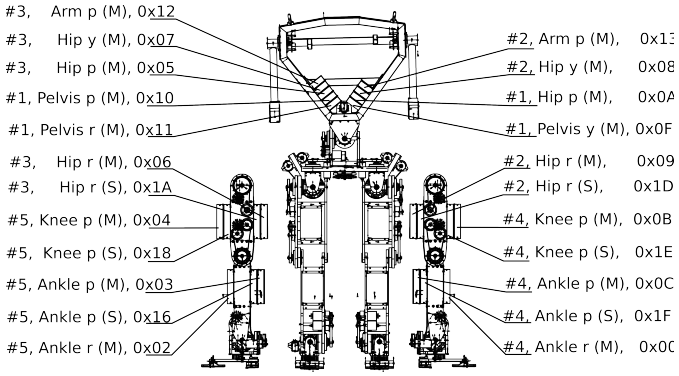


Figure 3.7: The distribution of the EPOS on the robot, and on the five CAN networks. The descriptive texts are formed as CAN network number, Joint (Master/Slave) and Address.

3.4 Force Torque Sensor Model

The FTS sensors is made up of six individual strain gauges. The are mounted in pairs on three beams, such that each beam has a bending, and a shearing oriented strain gauge. There are off course some crosstalk between them as they are bendt when the other is sheared, and vice-versa.

The FTS sensor is shown in Figure 3.8. The beams are rotated 120 degrees, so the measurements done needs to be rotated, and combined to get the data as an axis aligned set of forces and torques.

It is assumed that the strain gauges are nearly linear around the point of operations for AAU-BOT1, and thus the rotation of the gauges, and the combination of the data can be formed as:

$$\begin{bmatrix} F \\ \tau \end{bmatrix} = C_{6x6} \cdot v$$

Where:
C_{6x6} is a matrix containing both the rotation, and the linear combination of the sensor data.
v is a vector containing all the sensory data, sorted as $[v_{b1}, v_{s1}, v_{b2}, v_{s2}, v_{b3}, v_{s3}]^T$.
F is a vector containin the forces $[F_x, F_y, F_z]^T$
τ is a vector containing the torqes $[\tau_x, \tau_y, \tau_z]^T$

The matrix constants used is found by the previous group, and it is assumed that the calibration done was done sufficiently well, and is still representing the dynamics properly.

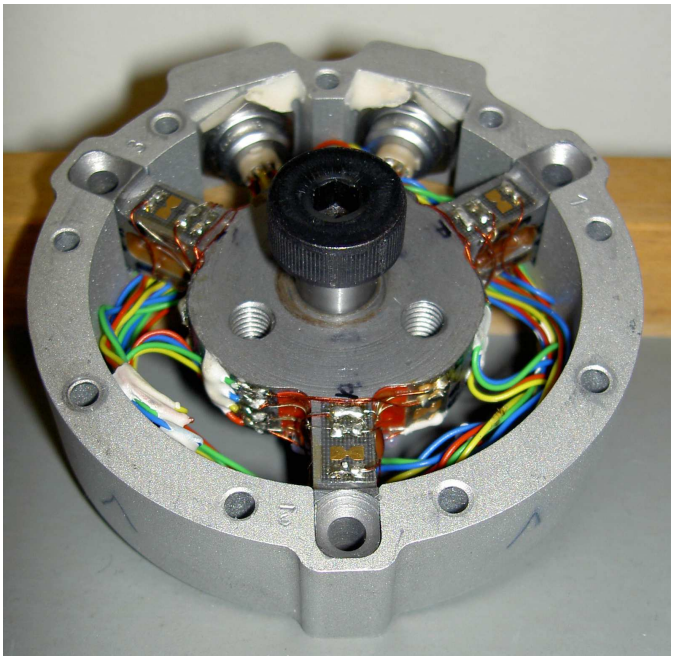


Figure 3.8: An image of the FTS sensor. The center part is connected to the leg, and the outer rim is attached to the foot. The three beams, that connects the center part with the outer rim, is bend or sheared slightly as a result of external forces.

3.5 Forward Kinematics

To be able to determine the CoM location, and other similar information, it is necessary to map the information available to obtain body positions. This is done through forward kinematics, which uses the joint angular sensors, and the Inertia Measurement Unit (IMU) data to calculate positions.

In previous AAU-BOT1 projects, a kinematic model was developed that was fast to compute. The model uses global joint angles, and kinematic link vectors to calculate the current orientation of each link. The only problem with this model, is that it is necessary to know the global joint angles, to form the kinematic chains.

When simulating, the global joint angles can be obtained quite easily, however while the robot is active, the measurements done on joint angles are all provided relative between links, and not as absolute rotations. Hence to determine the global rotation matrices, the relative joint angles, must be put into rotation matrices and then multiplied together to form the needed global rotations.

The result is a matrix for each joint, that is multiplied on each link vector, and then finally assembled to form the positions. It is a specific case of more general solutions such as the Denavit Hartenberg model, which generalizes joint movements as a series of transformation matrices.

In this thesis the existing model is extended to include the location of the IMU allowing the use of the model in a global coordinates system, and not only in a robot specific coordinate system. Also alot of CoM locations of the individual bodies, where found to be placed at unreasonable coordinates, hence they are updated in this thesis.

The model is implemented as a series of child and parent chains, and assembles the vertices

which holds the locations of relevant joints and link CoM's in parent coordinates. The kinematic link vector, describes the position of the child joint relative to the parents joint in zero position.

$$r_i = [x_i, y_i, z_i]^T$$

To assemble the chain in zero position, children are added to the parents, and then placed in a matrix.

$$r_A = [r_n + \cdots + r_1, r_2 + r_1, r_1, r_0]$$

Assembly when some joints are rotated, is a matter of rotating children before they are added.

$$r_A = [R_1^0 R_2^1 \cdots R_n^{n-1} r_n + \cdots + R_1^0 r_1, R_1^0 R_2^1 r_2 + R_1^0 r_1, R_1^0 r_1, r_0]$$

Also notice that an assembled chain may be represented in a chains parent, by applying a rotation, and a translation.

$$r_B = R_A^B r_A + r_B$$

Any joint on the robot is only rotating about one axis, so by assembling the chain from child to parent, the rotation matrices are easily determined as one of the three principal axis rotation matrices.

$$\begin{aligned} R_x(\theta) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \\ R_y(\theta) &= \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \\ R_z(\theta) &= \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Rotations about the IMU, is provided by the sensor as a quaternion, this is converted to a rotation matrix, as it was necessary during the development to cancel out the yaw rotation. This was due to the power cables, that distorts the magnetic field, which the IMU uses to orient it self. A rotation matrix can be formed from a quaternion by:

$$R_{xyz} = R_q = \begin{bmatrix} (1 - 2q_y^2 - 2q_z^2) & (2q_x q_y - 2q_z q_w) & (2q_x q_z + 2q_y q_w) \\ (2q_x q_y + 2q_z q_w) & (1 - 2q_x^2 - 2q_z^2) & (2q_y q_z - 2q_x q_w) \\ (2q_x q_z + 2q_y q_w) & (2q_y q_z + 2q_x q_w) & (1 - 2q_x^2 - 2q_y^2) \end{bmatrix}$$

Where:
$q = [q_x q_y q_z q_w]$ is a quaternion.
The new kinematic link vector and CoM vectors are presented in Table 3.5 and 3.6, and in Figure 3.9.

3.5.1 Estimating Toe Angles

The toe angles are not directly obtainable from sensors, but can be estimated using the IMU and the forward kinematics.

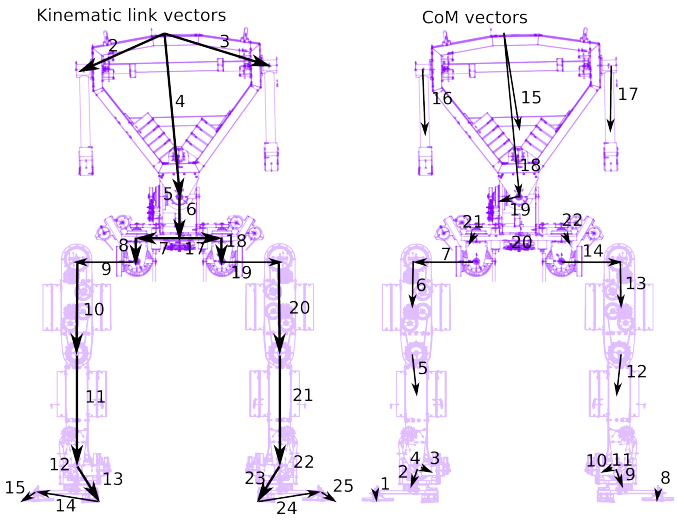


Figure 3.9: Location of the link and CoM vectors.

Index	CoM vectors [m]	Mass [kg]	Link Name	Parent joint
1	$[0.0031, 0.0033, 0.0149]^T$	0.1722	Right Toe	Right Toe
2	$[0.0309, -0.0029, 0.0900]^T$	0.8017	Right Heel	Right Ankle Roll
3	$[-0.0483, -0.0183, -0.0143]^T$	1.5151	Right Ankle Bracket	Right Ankle Roll
4	$[-0.0031, 0.0046, -0.0002]^T$	0.4698	Right Ankle Cross Axle	Right Ankle Pitch
5	$[0.0008, 0.0306, -0.2458]^T$	5.2752	Right Shin	Right Knee
6	$[0.0016, -0.0469, -0.1445]^T$	6.6561	Right Thigh	Right Hip Pitch
7	$[0.0007, -0.0080, -0.0002]^T$	0.6398	Right Hip Cross Axle	Right Hip Roll
8	$[-0.0001, 0.0001, -0.0149]^T$	0.1722	Left Toe	Left Toe
9	$[0.0281, -0.0000, -0.0900]^T$	0.8017	Left Heel	Left Ankle Roll
10	$[-0.0521, -0.0175, -0.0142]^T$	1.5151	Left Ankle Bracket	Left Ankle Roll
11	$[-0.0060, -0.0061, -0.0002]^T$	0.4698	Left Ankle Cross Axle	Left Ankle Pitch
12	$[-0.0020, -0.0320, -0.2456]^T$	5.2752	Left Shin	Left Knee
13	$[0.0002, 0.0455, -0.1445]^T$	6.6561	Left Thigh	Left Hip Pitch
14	$[0.0007, 0.0066, -0.0002]^T$	0.6398	Left Hip Cross Axle	Left Hip Roll
15	$[0.0418, 0.0010, -0.2748]^T$	11.6181	Torso	IMU
16	$[0.1037, -0.0046, -0.2451]^T$	0.9059	Right Arm	Right Shoulder
17	$[0.1037, -0.0052, -0.2310]^T$	0.9059	Left Arm	Left Shoulder
18	$[-0.0046, -0.0040, -0.0001]^T$	0.7543	Waist Cross Axle	Waist Roll
19	$[-0.0036, -0.0390, -0.0410]^T$	2.1675	Waist Bracket	Waist Pitch
20	$[-0.0221, -0.0007, 0.0158]^T$	4.6891	Pelvis	Waist Yaw
21	$[-0.0465, -0.0287, -0.0205]^T$	3.0516	Right Hip Bracket	Right Hip Yaw
22	$[-0.0465, 0.0274, -0.0206]^T$	3.0516	Left Hip Bracket	Left Hip Yaw

Table 3.5: CoM vectors, and their relations to other joints.

Index	Kinematic Link vectors [m]	Joint name	Parent Joint	Joint Rotation
1	$[0,0,0]^T$	IMU	None	R_{xyz}
2	$[0.0484,-0.2931,-0.1360]^T$	Right Shoulder	IMU	R_y
3	$[0.0484,0.3169,-0.1360]^T$	Left Shoulder	IMU	R_y
4	$[0.0434,0.0119,-0.5550]^T$	Waist Pitch	IMU	R_y
5	$[0,0,0]^T$	Waist Roll	Waist Pitch	R_x
6	$[0,0,-0.1500]^T$	Pelvis Yaw	Waist Roll	R_z
7	$[0,-0.1400,0]^T$	Right Hip Yaw	Pelvis Yaw	R_z
8	$[0,0,-0.0640]^T$	Right Hip Roll	Right Hip Yaw	R_x
9	$[0,0,0]^T$	Right Hip Pitch	Right Hip Roll	R_y
10	$[0,-0.0120,-0.3100]^T$	Right Knee	Right Hip Pitch	R_y
11	$[0,0,-0.3700]^T$	Right Ankle Pitch	Right Knee	R_y
12	$[0,0,0]^T$	Right Ankle Roll	Right Ankle Pitch	R_x
13	$[-0.0640,-0.0029,-0.1139]^T$	Right Heel	Right Ankle Roll	-
14	$[-0.1920,-0.0029,-0.0209]^T$	Right Toe Joint	Right Heel	R_y
15	$[0.0531,-0.0029,-0.0209]^T$	Right Toe Tip	Right Toe Joint	-
16	$[-0.0469,-0.0029,-0.209]^T$	Right Meta Phalanx	Right Toe Joint	-
17	$[0,0.1400,0]^T$	Left Hip Yaw	Pelvis Yaw	R_z
18	$[0,0,-0.0640]$	Left Hip Roll	Left Hip Yaw	R_x
19	$[0,0,0]^T$	Left Hip Pitch	Left Hip Roll	R_y
20	$[0,0.0120,0.3100]^T$	Left Knee	Left Hip Pitch	R_y
21	$[0,0,-0.3700]^T$	Left Ankle Pitch	Left Knee	R_y
22	$[0,0,0]^T$	Left Ankle Roll	Left Ankle Pitch	R_x
23	$[-0.0640,0,-0.1139]^T$	Left Heel	Left Ankle Roll	-
24	$[0.1920,0,0.0209]^T$	Left Toe Joint	Left Heel	R_y
25	$[0.0531,0,-0.0209]^T$	Left Toe Tip	Left Toe Joint	-
26	$[-0.0469,0,-0.0209]^T$	Left Meta Phalanx	Left Toe Joint	-

Table 3.6: Kinematic Link Vectors .

For this thesis it was considered if the robot could do push off by standing on the toes. To do this and maintain balance it is necessary to estimate the toe angle, so that the correct position of the CoM can be estimated.

It is assumed for this task, that the foot is always parallel with the ground in the foots roll axis.

$$\theta_{Toe} = -asin\left(\frac{(Heel_z - ToeTip_z}{|ToeTip + ToeJoint)|}\right)$$

3.5.2 Estimating Global Position

To get the robots coordinates in world coordinates, it is possible to integrate the position of the robot. A simple version where the accelerometers in the IMU is not combined with the kinematics, is used in this thesis. It is instead assumed that a foot does not slip on the floor, when ground contact has been made.

The FTS in the ankles reveals if a foot has ground contact, and thus the following rules are used for integration.

$$
\begin{aligned}
for \quad LeftFootContact \quad & \begin{cases} LeftToeTip = LeftToeTip[n-1] \\ RightToeTip = RightToeTip - LeftToeTip[n] + LeftToeTip[n-1] \end{cases} \\
for \quad RightFootContact \quad & \begin{cases} LeftToeTip = LeftToeTip - RightToeTip[n] + RightToeTip[n-1] \\ RightToeTip = RightToeTip[n-1] \end{cases} \\
for \quad BothFeetContact \quad & \begin{cases} LeftToeTip = \frac{(LeftToeTip - RightToeTip[n] + RightToeTip[n-1]) + LeftToeTip[n-1]}{2} \\ RightToeTip = \frac{(RightToeTip - LeftToeTip[n] + LeftToeTip[n-1]) + RightToeTip[n-1]}{2} \end{cases} \\
IMU_{pos} = \quad & -(RightToeTip)
\end{aligned}
$$

Notice the notation of $[n-1]$, this indicates the old world coordinates for this point, where $[n]$ indicates the new position, new positions are calculated in IMU coordinates, and thus when the location of the feet in world coordinates are integrated, the IMU can be located in world coordinates.

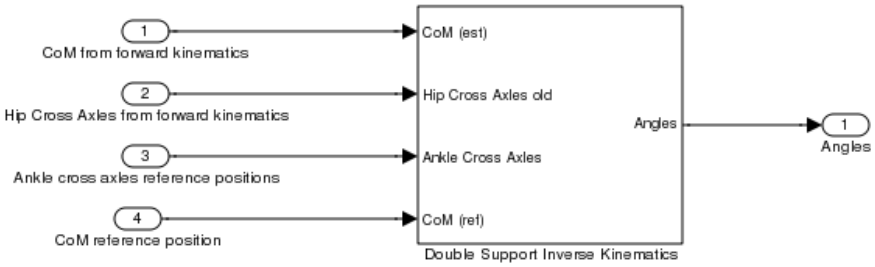


Figure 3.10: Black box representation of the inverse kinematics.

3.6 Inverse Kinematics

The opposite of the forward kinematics is the inverse kinematics. When capturing motion for movies or kinematic analysis, using markers, and visual positioning, the two should map each other directly and uniquely. However, where forward kinematics often have a single solution, as all the joint angles are nearly always provided, inverse kinematics often allows redundant solutions, as all the joint positions are seldomnly provided.

In fact, inverse kinematics are often applied when it is more or less irrelevant how most of the joints are rotated, and the only important thing is where the end effector link is positioned.

Inverse kinematics has already been suggested for the robot, and solution has been developed, for either a left or a right phase, but the states regarding positions are not relating to the solution for walking described in this thesis. Hence the inverse kinematics is reformed to include CoM as one of the input positions. See Figure 3.10 for an input / output overview of the inverse kinematics block.

Instead of reevaluating the original approach to include an extra link, which it was never designed for, this solution splits up the problem in a series of trigonometric solutions. This trigonometric approach allows for detection of joint limitations while calculating, and decisions can be made on what to do in case of an impossible reference, however there has not been implemented any error handling at this point in time.

3.6.1 Including CoM

As CoM is not rigidly connected with the robot, but defined as the weighed average of all masses, a simple inverse kinematics solution that tracks CoM is difficult, as there will nearly always be redundant solutions with regards to moving the CoM.

For this solution it is chosen that for 1 sample in time to consider the CoM as rigidly connected with the kinematics, and introduce a tracking latency of one sample. The system should converge towards the correct position, as long as it is possible to reach it, as the delay introduced will not cause errors or oscillations larger than the system can be overshooted in one sample, with no initial velocity.

The CoM is related to the hips position, and this relation is then used to determine new hip positions.

$$\begin{aligned} CoM_{RH}[n] &= RH[n-1] - CoM_{RH}[n-1] \\ RH[n] &= CoM_{RH}[n] + CoM_{ref}[n] \end{aligned}$$

Where:

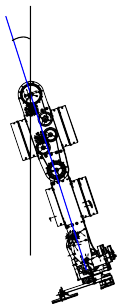


Figure 3.11: Leg pitch angle, is the pitch angle of the vector that spans from the hip to the ankle. Also denoted θ_{lp}

RH is a shorthand notation for "Right Hip Cross Axle"
$[n-1]$ denotes the values in the last time sample

3.6.2 Trigonometric Alignments

Now that the hip cross axles are located, the ankles are taken into the system, and the joint angles can be found. Here the solution is presented for a single leg, the other leg is simply a duplicated system.

It is assumed that the legs are always kept axis aligned with each other, and with the robot planes. This means that if any of the yaw angles are used, the others must be given a similar angle. It is also assumed that the waist is kept planar with the ground.

3.6.2.1 Pitch Angles

The pitch angles all relate to one another, this solution assumes that the waist is always kept vertical in the sagittal plane. When this is the case, the knee, the hip, and the ankle forms a triangle, that determines the leg length.

However as the ankle and the hip may have rotated the leg also, the first thing that is determined is the leg pitch angle θ_{lp} See Figure 3.11.

The leg pitch angle should be independent of the leg roll angle, and to make sure that happens, the leg can be considered in the plane of the rolled leg before a trigonometric function is applied. Or a projected vector could be used. In this case the Pythagorean is used to cancel out the effect of a rotation on the roll angle.

$$\theta_{lp} = atan2(\sqrt{leg_z^2 + leg_y^2}, leg_x) - \frac{1}{2}\pi$$

Where:
leg_{xyz} is the leg vector components, the leg vector stretches from the hip cross axle to the ankle cross axle.
The next thing that can be calculated is the length of the leg, here it would be natural to use the euclidean norm directly, but the kinematic link vectors reveals that the hip and the ankles are not located directly above one another in zero position. There is a small displacement in the y axis, which must be subtracted from the leg vectors total length, to determine the length which the pitch angles can actually adjust.

The length is found as

$$leg_{lp} = \sqrt{(leg_x^2 + leg_y^2 + leg_z^2) - d^2}$$

Where:
d is the displacement in the y axis.
Notice that this equation potentially can give complex results, but this requires that the knee is bend so much that the legs length is less than the displacement. This is not physically possible, so this error will not occur in pratice.

The knee angle can now be found as a simple cosine relation.

$$\theta_{knee} = \pi - acos(\frac{a^2 + b^2 - leg_{lp}^2}{2 \cdot a \cdot b})$$

Where:
a is the shin length, defined as the z component of the kinematic vector, (0.37)
b is the thigh length, defined as the z component of the kinematic vector, (0.31)
Now determining the hip angles, is a matter of combining the knee angle, and the leg angle, but as the knee angle itself produces a leg angle this must be taken into account. Fortunately this can also be done by a cosine relation.

$$\begin{aligned} \theta_{Hippitch} &= \theta_{lp} - \theta_{knee} + acos(\frac{a^2 - b^2 + leg_{lp}^2}{2 \cdot a \cdot leg_{lp}}) \\ \theta_{anklepitch} &= -(\theta_{Hippitch} + \theta_{knee}) \end{aligned}$$

3.6.2.2 Roll Angles

Just as with the pitch angles, the leg length is found to determine the hip and ankle angles, and using the same logic, the leg pitch should not effect the leg roll angles. But this time around, there is one less variable to calculate and this makes things simpler, and the ankle and the hip can be found directly.

$$\begin{aligned} \theta_{Hiproll} &= -\left(acos\left(\frac{Hip \bullet (-leg)}{leg_{lp} \cdot \|Hip\|}\right) + \frac{1}{2}\pi\right) + atan2(d, leg_{lp}) \\ \theta_{Ankleroll} &= atan2\left(-leg_y, \sqrt{leg_z^2 + leg_x^2}\right) + atan2(d, leg_{lp}) \end{aligned}$$

Where:
Hip is the vector spanning from the legs hip cross axle, to the other hip cross axle.

4

Controller Design

"Make no little plans; they have no magic to stir men's blood and probably themselves will not be realized. Make big plans; aim high in hope and work, remembering that a noble, logical diagram once recorded will not die, but long after we are gone be a living thing, asserting itself with ever-growing insistence. Remember that our sons and our grandsons are going to do things that would stagger us. Let your watchword be order and your beacon beauty."
—Daniel Hudson Burnham

4.1 Limit Cycle of Hybrid System

The biped walker is a hybrid system. It has discrete switching events with update laws, and it has continuous parts, that can be described as Euler Lagrange equations of motion. It is natural to determine a stable limit cycle from a Hybrid system analysis. For AAU-BOT1 there exists multiple DoF, and multiple states. This makes the equations describing the dynamic system almost impossible to gain insights from, in their full extends.

Reducing the dynamics, by mapping some states as functions of others reduces the complexity. i.e. the knee, hip, and ankle angles, can be reduced to a function of leg angle and leg length. Further more the dynamics can be greatly reduced by eliminating some of DoF for the robot. A popular choice is to analyze the robot in the sagittal plane only.

An extreme of the reduced dynamics approach is to analyze the system in the sagittal plane, and assume that the legs has mass-less prismatic tips, so that changing the length of a leg does not move the mass of the leg. The resulting system is referred to as a compass gait walker, and is recognized as McGeer's 2D straight legged walker, as seen in Appendix B, where this example is analyzed using a classical Newtonian approach.

The equations of motion has been presented earlier in 2.3, and is here simply rewritten to be on a hybrid system formulation of the states. See Equation 4.1 and 4.2 , and Figure 4.1 for a physical interpretation of the state variables.

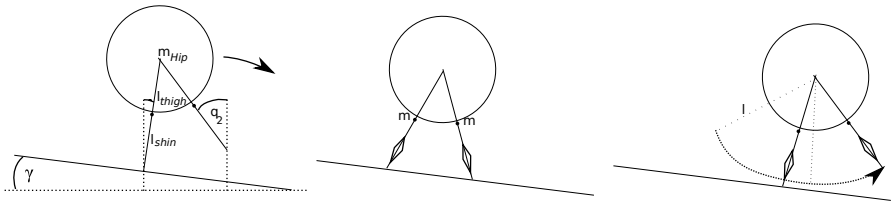


Figure 4.1: Compass gait walker, notice the prismatic joints, that retracts to avoid ground when swinging, it is not modeled, but simply assumed.

$$\left.\begin{array}{l} p_1\ddot{q_1} - p_2 cos(q_1 - q_2)\ddot{q_2} - p_2 sin(q_1 - q_2)\dot{q_2}^2 - p_4 sin(q_1) = 0 \\ p_3\ddot{q_2} - p_2 cos(q_1 - q_2)\ddot{q_1} - p_2 sin(q_1 - q_2)\dot{q_1}^2 + p_5 sin(q_2) = 0 \end{array}\right\} \quad for \quad q \notin \Gamma_- \tag{4.1}$$

$$F^{(i)} : \Gamma_-^{(i)} \rightarrow \Gamma_+^{(i)} : \{q_1^+ = q_2^-, q_2^+ = q_1^-, \dot{q}^+ = P_q(q^-)\dot{q}^-\} \quad for \quad q \in \Gamma_- \tag{4.2}$$

Where $P_q(q^-)$ is a function of q^- , which is the value of q at surface impact time, and p refers to physical parameters, such as length and mass:

$$\begin{aligned} p_1 &= (m_{Hip} \cdot l + m \cdot l_{shin} + ml)g \\ p_2 &= ml \cdot l_{shin} \\ p_3 &= m \cdot l_{shin}^2 \\ p_4 &= (m_{Hip} \cdot l + m \cdot l_{shin} + ml)g \\ p_5 &= m \cdot l_{shin} g \end{aligned}$$

and q is a state notation, refering to leg angles:

$$\begin{aligned} q_1 &= \theta \\ q_2 &= \phi \end{aligned}$$

The switching surface Γ denotes the geometric surface which stops the continuous flow, and as a result it must be the slope which the compass gait walker walks down, so a function can be written where both feet has ground contact.

$$\Gamma = \{q \in \mathcal{R}^2 : H(q) = cos(q_1 + \gamma) - cos(q_2 + \gamma) = 0\}$$

As the compass gait walker retracts the leg during a swing, this only occurs when the legs are extended fully and thus have similar lengths. The impulse effects of the impact can be described by a reset map:

$$P_q = \begin{bmatrix} p_1 - p_2 cos(q_1^- - q_2^-) & p_3 - p_2 cos(q_1^- - q_2^-) \\ -p_2 cos(q_1^- - q_2^-) & p_3 \end{bmatrix}^{-1} \begin{bmatrix} p_7 cos(q_1^- - q_2^-) - p_6 & -p_6 \\ -p_6 & 0 \end{bmatrix} \tag{4.3}$$

Where:
$p_6 = m \cdot l_{thigh} l_{shin}$
$p_7 = m_{Hip} l^2 + 2m \cdot l_{thigh} l$

4.1.1 Limit Cycles of a Hybrid System

When the slope γ is shallow, the gait is symmetric and periodic, and can be uniquely defined by a vector of 9 parameters, when considering the half period $T = T_p/2 > 0$.

$$p_* = [a, b, c, d, e, f, g, h, T] \in \mathcal{R}^9$$

And the the following 8 constants will denote the initial and final states:

$$\begin{aligned} q_*(0+) &= [q_{1*}(0+), q_{2*}(0+)]^T = [a, e]^T \\ \dot{q}_*(0+) &= [\dot{q}_{1*}(0+), \dot{q}_{2*}(0+)]^T = [b, f]^T \\ q_*(T-) &= [q_{1*}(T-), q_{2*}(T-)]^T = [c, g]^T \\ \dot{q}_*(T-) &= [\dot{q}_{1*}(T-), \dot{q}_{2*}(T-)]^T = [d, h]^T \end{aligned}$$

Allowing five algebraic relations to be defined. Combining Equation 4.1, 4.2, and 4.3, the following relation can be set up:

$$\underbrace{\begin{bmatrix} a \\ e \end{bmatrix}}_{q_*(0+)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} c \\ g \end{bmatrix}}_{q_*(T^-)} = \begin{bmatrix} g \\ c \end{bmatrix}$$

and

$$\underbrace{\begin{bmatrix} b \\ f \end{bmatrix}}_{\dot{q}_*(0+)} = P_q \left(\underbrace{\begin{bmatrix} c \\ g \end{bmatrix}}_{\dot{q}_*(T^-)} \right) \begin{bmatrix} d \\ h \end{bmatrix}$$

and then some of the constants can be found as

$$\begin{aligned} g &= a \\ c &= e = -a - 2\gamma \\ f &= \frac{c \cdot cos(2a + 2\gamma)p_2 - p_6 d}{p_3} \\ h &= \frac{d(p_3 p_7 - p_6 p_2) - b(p_3 p_1 - p_2^2 cos(2a + 2\gamma))}{p_3 p_6} cos(2a + 2\gamma) \end{aligned}$$

This leaves only the a,b,d and T to be defined. T can be found through numerical integration of the system equations, as the time for the walker to complete a step. So the problem of determining the last three constants can be defined as a standard optimization routine, and solved.

A popular choice of method is simply to run numerical integrations (simulations) and determine the constants which provides a stable limit cycle. This method should be applicable to the more complex system describing AAU-BOT1. However, it was experienced that in general when the system becomes more complex, the number of variables that must be optimized increases, and due to the simulation speed of the simulator, searching for a stable hybrid gait was not feasible within the time that remained of the project.

Had a limit cycle been found, the torques required to complete the gait could have been found by applying the system equations, and then a nominal trajectory could be defined in terms of

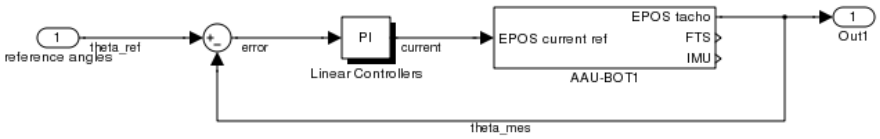


Figure 4.2: A classic PID controller. The simple structure dues not encompass data from other joints, and is thus 17 individual SISO controllers. This structure is prone to cross coupling problems..

state flow, and torque flows for each actuator. The non-linearity's could then be compensated with either nonlinear feedback (calculating the torque due to the gait on the fly), or feedforward compensation (offline trajectory, allows for pre-calculated torques). The control problem could be reduced to a linear problem of reducing the errors in trajectory tracking.

Instead of this solution, it is proposed to predict motions by reducing the model to an inverted pendulum. This governs the motion of the major part of the body during single support, if only the free swinging leg is allowed to move relative to the body. To do so requires a posture controller that converts motions to torques, and tracks them.

4.2 Posture Controller Design

The chosen posture controller has the structure shown in Figure 4.2, and is a classic PI regulator. Each joint is assumed to be independent of the other joints, and linearizable about the robots zero position. This assumption is known to be inadequate in some cases. The controller type is chosen due to a desire of minimizing development time, and to be able to quickly apply experiments for simulator parameter testing.

The PI controller is tuned for the free movement case presented next, and then afterward adapted for walking, and standing.

4.2.1 Free Move PI Tuning

The robot was fixated at the torso in an experimental rig, shown in Figure 4.3. Using the rig for experiments required the free movement version of the PI regulator. Each joint is considered an individual SISO system, consisting of a base frame, and an attached pendulum. The parent base frame is the torso, and, when modelling joints not directly attached to the torso, joints in between are considered welded, and thus a piece of the base plate.

Recollecting the State Space approach to linearized modelling, the basic matrices needed are A, B, C and sometimes D.

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

The modelling key is to choose a proper state vector, x, which has been chosen as the pendulum angle, and the derivative.

As the D matrix has little relevance for the plant behavior, it is as usual set to zero. The A matrix contains the passive dynamics behavior, and the B matrix contains the control signals effect on the actuator. The C is merely a choice of which states that are represented in the output.

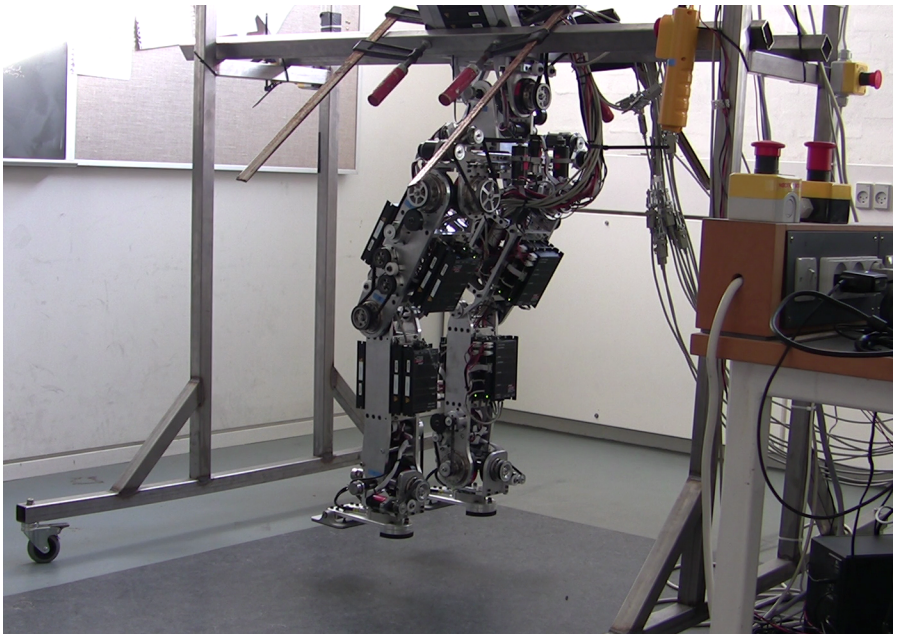


Figure 4.3: The robot was attached to a frame, such that it hang from the torso, in near rigid conditions.

$$
\begin{aligned}
A &= \begin{bmatrix} 0 & 1 \\ \frac{mlg}{I} & -\frac{\mu}{I} \end{bmatrix} \\
B &= \begin{bmatrix} 0 \\ \frac{G \cdot kt}{I} \end{bmatrix} \\
C &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
D &= 0
\end{aligned}
$$

Where
l is the pendulum rod length
m is the pendulum bob mass
G is the gear constant relating to the joint
kt is the motor constant relating to the joint
μ is the viscous friction component.
g is the gravity constant.
The resulting transfer functions can be found using the $tf(StateSpaceSystem)$ in MATLAB. They will have the form:

$$F(s) = \frac{\frac{G \cdot kt}{I}}{S^2 + \frac{\mu}{I}S - \frac{mlg}{I}}$$

The controller is used to move the poles in closed loop to get a better response from the system. Classically a step response would be used to show the efficiency of the controller, and a

zero pole placement, or a root locus plot would be used to show stability. However, the linearized model is not perfect, and for some of the joints other considerations must be taken into account, which is why the gains implemented deviates from the optimal gains for this non-linear model.

In general it was chosen to have a relatively slow response time on the linear model, to ensure a larger stability margin. The final gains where found using a step, a ramp, and a sine response on the nonlinear model, and later on the physical system. The physical system was used for posture control tuning as the stiction on the joints is considerable, tends to be angle and speed dependent, and hard to model accurately.

One of the crucial errors of the linear model occurs at the waist of the robot, where the three roll axes are assumed in the model to be independent. However, as this is where the effect of the cross couplings are maximized when the robot hangs freely in the air, the assumption leads to unstable controllers. Hence the proportional gain is scaled further down, to improve stability. This off course lowers the response time even more, and this is one of the areas where a MIMO controller could be used to improve the responses.

4.2.2 Hybrid Posture Controller Issues

When the robot stands on the feet, the robot is suddenly acting as an inverted pendulum on the joints. Inverting a pendulum can be done by multiplying g with -1. This moves the poles, as the last term of the denominator changes sign. For many of the joints, especially at the middle of the robot this does not destabilize them, as the controllers where tuned aggressively enough to keep the poles in the left half plane in both cases.

Where problems do occur due to the occurrence of support, is where the change of loads does not only involve the change of the sign, but also the mass with which the joint is loaded. This is particular easy to illustrate for the ankles, as they go from the case of having a pendulum of a mass close 1kg, hanging below the joint, to the case where 66kg of the robot is above the joint, acting as an inverted pendulum.

The difference is so extreme, that the controller that is tuned for the low mass, regular pendulum case, cannot maintain any near vertical angle on the robot when in Single support, and the controller tuned for Single support, will overshoot, to the point of instability just because it is lifted from the ground. Somewhere in between the two tuned gains, the double support is found.

The solution to this stability issue, is simply to switch the controllers for those joints depending on the robots state. Controllers that switches mode depending on discrete events are often called multi modal controllers, and are usually conceptualized as a state response to an event. This is in opposition to the continuous controllers that operates with an input/output terminology.

The proposed setup with a continuous controller that is defined for and engaged in each mode is known as a hybrid controller, and the usage of such a control system should always be backed up by an analysis of what happens to the system when the continuous controllers are switched by an event.

This is not done for the posture controller in this thesis. Primarily because it would be time consuming to develop the models required for an analysis and as a result take time from the development of the other parts of the controller. Instead it was simply assumed that it would not render the system unstable, and tested carefully in practice, where it worked as intended.

The determined controller gains are found in Table 4.1.

The multi modal part of the controller, is integrated in the MPG, and setup as a regular state machine.

$K_p \cdot 10^3$	$K_I \cdot 10^3$	$K_p \cdot 10^3$(ground contact)	Joint Name
500	0	250	Right Ankle Roll
250	0	250	Right Ankle Pitch
550	55	550	Right Knee
500	50	500	Right Hip Pitch
250	50	250	Right Hip Roll
500	0	500	Right Hip Yaw
500	0	500	Left Hip Yaw
250	50	250	Left Hip Roll
500	50	500	Left Hip Pitch
550	55	550	Left Knee
250	0	250	Left Ankle Pitch
500	0	250	Right Ankle Roll
300	60	300	Pelvis Yaw
500	50	500	Waist Pitch
500	75	500	Waist Roll
150	15	150	Right Arm
150	15	150	Left Arm

Table 4.1: The motors are fed the current reference in mA, hence the controller gains are multiplied with 10^3, however if the EPOS worked in SI units, that would not have been necessary.

4.3 Motor Pattern Generator

The MPG is responsible for generating the trajectory for the posture controller. This is done on-line, using two basic models for motion generation. Which model is used is dependent on the occurrence of a discrete event. This forms a hybrid control system that changes behavior when the system passes a switching surface, and operates continuously when the switching surface is left, and continues to do so until the next one is passed.

The multi modal part of the controller is implemented as a state machine as illustrated in Figure 4.4. The content of the individual states generate the pattern in continuous time.

It is chosen to have a waiting state, in where the robot has an active posture controller, but applies a constant reference. This has the effect that the FTS sensor data can be offsetted, correctly, and the drift can be compensated before experiment start.

The transition from the wait state is when ground contact is obtained on both feet. This sets starts a timer, so that the operator can move away from the robot before it moves. This is important, as the safety procedures in the laboratory must be respected, See Appendix K for more on laboratory conditions.

Only four of the states are actually generating a trajectory pattern, that will move the robot. Of the four, two of them are exact mirrors of the two others. So in reality, only two MPG states are distinctly different. The two are explored in the following sections.

The last state is a termination of the motion. If the robot looses ground contact it holds a zero current reference to the actuators. This is a safety feature, if the operator should panic and hoist the robot instead of shutting it down.

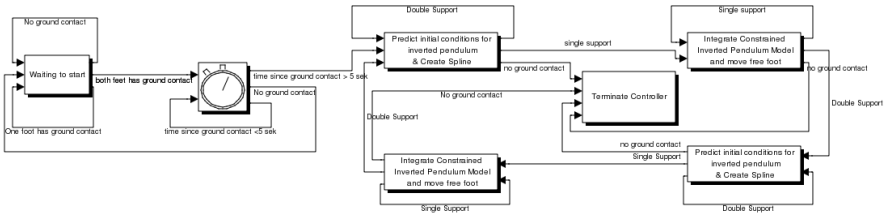


Figure 4.4: An overview of the state machine governing the shifts in behavior of the MPG. There is actually two distinct modes, in which the MPG operates, it is repeated and mirrored two create four main phase related states. The remaining states are implemented for initiation and safety reasons.

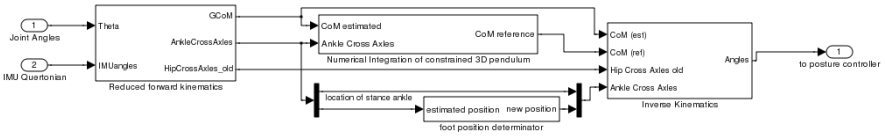


Figure 4.5: Overview of the single support motor pattern generator system.

4.3.1 Single Support MPG

The basic idea in single support is to utilize the natural passive dynamics, and assume that the ankle joint on the support leg is a passive friction less joint, and then let the robot fall as a result of the entry velocities and angles. This is relating to the passive walkers, as the also assumes a passive link between the robot and the ground.

This assumption is far from true on AAU-BOT1, so to create an approximate to the assumptions, the ankle needs to actively track the motions that would have occurred if it was a passive frictionless ball joint. To do this a simple model of the robot (a nonlinear 3D inverted pendulum model) is utilized. The model is unfortunately not simple enough to predict the entire motion in before hand and then store the trajectory, but it can be integrated with a fixed step integration, which can be done fast enough to track the physical model on-line.

See Figure 4.5, for an overview of the MPG system in single support.

As the trajectory is generated on the fly, the model also includes a virtual wall, which ensures that small entry angles does not cause CoM to move beyond vertical position in the frontal plane. This will, if the velocity is close enough to zero when vertical position is reached, ensure frontal plane balance while in single support. As the robot is only allowed to tilt towards the side where the other foots placement is possible.

The placement of the foot is approximated with a B-Spline, as the step length is given, and the target velocity is known as zero, the only unknown is the foots step width. This depends on the diversion of the inverted pendulum from the ideal target chosen in double support.

4.3.1.1 3D Pendulum Model (Standing Pencil Model)

A 3D pendulum model, can be setup by a simple Euler Equation, with a moment due to gravity is included.

$$J\dot{\omega} = J\omega \times \omega + mg\rho \times R^T e_3$$

Where:

J is the constant Inertia Matrix, given in the Body fixed frame
ρ is the rod length
e_3 is the direction vector for the gravity constants in the inertial frame, i.e. $e_3 = [0, 0, 1]^T$
$R^T$is the rotation from the inertia frame to the pendulum fixed frame, so that $R^T e_3$ is the gravity orientation in the pendulum fixed frame.

Remembering that a cross product in 3D is merely a skew symmetric matrix operation, the Euler equation is strikingly familiar to that of a single rigid body, with the addition of a constraint that gives the point of rotation. It can also be noted that the model is plainly uninteresting if the rod length is zero, as it then models a satellite, with no gravity effects. To solve a rigid body system with constraints, a numerical integrator is needed. Simulink and MATLAB comes with a variety of integrators, but none of them are symplectic (energy conserving). Sim-Mechanics comes with a build in symplectic integrator, but it does not run very fast, can't handle singularities, which is a problem for integrating an inverted pendulum at very low speeds, and it is not optimized for fixed step integration.

A better integrator alternative is a symplectic integrator suited for fixed step integration, that is not prone to singularities. In this thesis the Störmer Verlet method for numeric integration is applied.

4.3.1.2 Störmer Verlet Integrator

The Störmer Verlet integrator is a symplectic integrator normally used in molecular dynamics. The integrator it self is quite simple, but well suited for this type of problem. The integrator is based on two third-order Taylor Series expansions on the position of a particle in a three dimensional space. The first Taylor expansion is forward in time, the other is backward.

If the position of the particle is denoted $r(t)$ the Taylor expansions are:

$$r(t + \Delta t) = r(t) + \dot{r}(t)\Delta t + \frac{1}{2}\ddot{r}(t)\Delta t^2 + \frac{1}{6}\dddot{r}(t)\Delta t^3 + \mathcal{O}(\Delta t^4) \quad (4.4)$$

$$r(t - \Delta t) = r(t) - \dot{r}(t)\Delta t + \frac{1}{2}\ddot{r}(t)\Delta t^2 - \frac{1}{6}\dddot{r}(t)\Delta t^3 + \mathcal{O}(\Delta t^4) \quad (4.5)$$

Where:
$\mathcal{O}$ denotes the upper bound on the error.

The two Equations 4.4, can be combined to a single equation, which is the basic Störmer Verlet Integrator.

$$r(t + \Delta t) = 2r(t) - r(t - \Delta t) + \ddot{r}(t)\Delta t^2 + \mathcal{O}(\Delta t^4)$$

The integrator thus avoids using the velocity component to integrate the motions. Only the position and the acceleration is used to predict the next acceleration. This off course infers an error in the integration if there exists velocity specific acceleration terms, such as viscous friction, or aerodynamic effects. For the case of a simple pendulum, no such terms exists, so this is not a problem. Should they have been included, the Störmer Verlet Leapfrog integrator adds the velocity term, without violating the energy conservation.

Constraints

The integrator predicts the behavior of a particle in a vector field defined by $\ddot{r}(t)$, it can be defined as a constant, or as a function. For the system described in this thesis a time invariant homogeneous gravitational field is used ($[0, 0, -g]$). Hence a particle would forever be accelerated downwards.

The integrator does not use velocity components, so constraints like, keep this distance to a position, can be quite easily implemented, by moving the object when updating the position. Still if the constraint is something like a collision detection it can be expected that an object can penetrate the surface as a function of it's current speed, and the sample time.

Constraint Example 3D pendulum

A constraint that will produce a simple 3D pendulum, can be formed as:

$$r(t+\Delta t) = \frac{\widetilde{r}(t+\Delta t)}{\|\widetilde{r}(t+\Delta t)\|} \cdot l$$

Where:
$\widetilde{r}(t+\Delta t)$ is the output from the unconstrained Verlet integrator.

Constraint Example Floor

A constraint that will produce a fully plastic impact in $z = 0$ will be:

$$r(t+\Delta t) = \begin{cases} \widetilde{r}(t+\Delta t) & for\, \widetilde{r}_z(t) \geq 0 \\ 0 & for\, \widetilde{r}_z(t) < 0 \end{cases}$$

A constraint that will produce a simple floor in $z = 0$, with fully elastic impacts will be:

$$r(t+\Delta t) = \begin{cases} \widetilde{r}(t+\Delta t) & for\, \widetilde{r}_z(t) \geq 0 \\ \widetilde{r}(t+\Delta t) + 2(r(t) - \widetilde{r}(t-\Delta t)) & for\, \widetilde{r}_z(t) < 0 \end{cases}$$

Notice though that a fully plastic impact absorbs energy from the system.

4.3.1.3 Free Moving Leg Placement.

The foot placement serves two purposes, the first is to sustain frontal plane balance, the other is to ensure that the forward motion in the saggitale plane is possible.

The step width should never change if the system is waking in a stabile gait, with no deviations regarding sideways balancing. Should this however not be the case, larger step width slows down a fall towards the foot, and smaller stepwidth makes the robot less stable. The robot can become to stable, and thus unable to walk.

It is decided that a nominal stepwidth is chosen which the system returns to if in balance. If frontal plane accellerations on the way out of a single support is larger than predicted the stepwidth is increased proportionally to the error in the inverted pendulm angle.

The forward target is actually known, as the step length function predicted it, it is however suggested that a different approach is used to actually place the foot. This is done to compensate for errors occuring during the step.

An initial guess for step location is found by adding the step length to the old ankle location. The step width for the initial location can be changed during the step according to the balance. The line provided between the old ankle, and the desired ankle location, is used for tracking in the plane.

Instead of integrating the location on the line as a function of time, the location on it is found by forming a line between the stance ankle, through the current GCoM location onto the line. The point of intersection is the current desired position of the ankle.

This gives an accelleration in the beginning of a swing, and a decelleration at the end. The height of the ankle can then be found by:

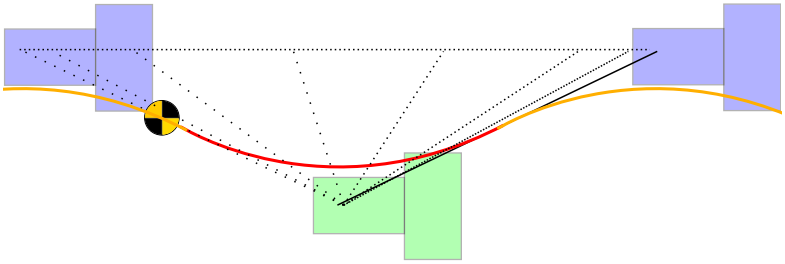


Figure 4.6: The strategy of foot placement. A straight line is formed from the old ankle location, and directly to the initial desired ankle location. If balancing is not necessesary this line is not altered. Motion along the line is given as the intersection between the elongated vector from the stance ankle to the GCoM

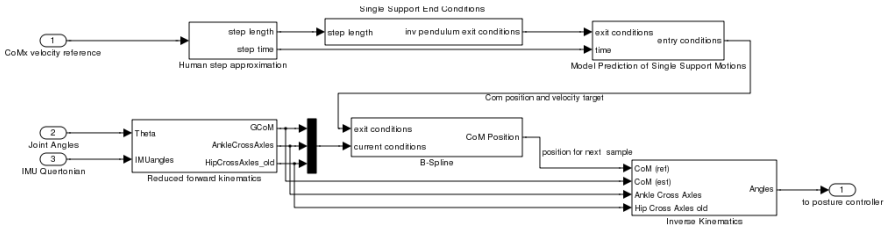


Figure 4.7: Overview of the double support pattern generator system.

$$foot_z = .42^{-(x^2+x)} \cdot \frac{2}{1+e^{\alpha x}}$$

Where:
x is travelled distance on the ankle to ankle line, normed by the total desired distance
α is a coefficient for determining the flattening.

4.3.2 Double Support MPG

When in Double Support, The pattern generator is responsible for guiding the CoM into the next Single Support phase with the proper velocities, and positions at the time of state shift. This must be done without moving the feet.

To do this properly it is necessary to predict which double support exit velocities, and what double support exit position that will provide the desired motion in the Single Support mode.

The desired Single Support motion is only provided as the exit conditions of the Single Support mode, which must place CoM between the feet in the frontal plane, and with a forward travel distance S that is predicted by the step length approximation, at a time t. Converting this to entry conditions for single support is determined by model prediction, and when the entry conditions are found, they are used as targeting for double support path planning.

This is all tangled up with forward and inverse kinematics to get a Cartesian coordinate system for path generation, and still actually provide a reference signal for the posture controller. See Figure 4.7 for an overview of how the double support pattern generator determines the path.

4.3.2.1 Model Prediction of Single Support Behavior

Entering single support with a CoM velocity much larger than 0, will produce dynamic effects, that can cause the robot to fall. Any acceleration in joints, will effect the dynamic behavior, however depending on the masses connected to the joints, the effects can be either large or small.

It was chosen to model the relationship between the support foot, and the entire robots CoM as a 3D inverted pendulum. To avoid falling it is necessary to predict the behavior of it, so that the entry velocities of the pendulum is not causing it to fall the wrong way.

This is quite problematic to predict when using a simple model, as the inverted pendulum system is not analytically integratable.

However the planar pendulum, can be very well approximated with a function that resembles what would intuitively be expected of the analytic integral, this is mostly due to the fact that the first integral of motion can be found in the planar case, so this is done first.

Analytic Determination of the First integral of Motion of a 2D pendulum

A simple planar pendulum model should be well known to most engineers, and is presented in Equation 4.6, without further introductions.

$$\ddot{\theta} = \frac{-g}{l} sin(\theta) \tag{4.6}$$

where:
θ is the angle between the rod and downwards vertical.
l is the length of the rod

The equation of motion for the pendulum, can in it self be numerically integrated and allow simulation, or model prediction for controllers that have a reasonably low horizon, however for long term motion planning, the analytic integral can provide an extremely fast prediction, and even give better accuracy. Often the solution to the first integral of the equation of motion is provided as Equation 4.7, however this result is not exactly true, as it does not include initial angular velocity.

$$\dot{\theta} = \int_{\theta_0}^{\theta} \ddot{\theta} = \sqrt{\frac{2g}{l}(cos(\theta) - cos(\theta_0))} \tag{4.7}$$

Initial angular velocity has relevance for this thesis as this is the double support exit velocity of the CoM, hence the first integrate of the equation of motion is derived below:

$$\begin{aligned} \Delta P &= mgh \\ \Delta K &= \frac{1}{2}mv^2 + \frac{1}{2}mv_0^2 \end{aligned}$$

where:
m is the bob mass
v is the bob velocity
h is the vertical height of the bob.

The energy is conserved, hence all potential energy is translated to kinetic energy, and vice-versa.

$$
\begin{aligned}
\Delta P &= \Delta K = mgh = \frac{1}{2}mv^2 + \frac{1}{2}mv_0^2 \\
\frac{2mgh}{m} &= (v^2 + v_0^2) \\
v^2 &= 2gh - v_0^2 \\
v &= l \cdot \dot{\theta} \\
h &= l \cdot (cos(\theta) - cos(\theta_0)) \\
\dot{\theta} &= \frac{1}{l}\sqrt{2gh + l\dot{\theta}^2} \\
\dot{\theta}^2 &= \frac{1}{l^2}\left(2gl(cos(\theta) - cos(\theta_0)) + l^2\dot{\theta}_0^2\right) \qquad (4.8) \\
\dot{\theta}^2 &= \frac{1}{l}\left(2g(cos(\theta) - cos(\theta_0))\right) + \dot{\theta}_0^2 \qquad (4.9)
\end{aligned}
$$

The first integral of the inverted pendulum model presented in Equation 4.8, can be used to predict the motion of the planar robots CoM as a function of the entry velocity, and the entry angle. The model can be applied on a planar robot when in single support and assuming frictionless passive ankles, and otherwise static joints. Hence a few remarks on balancing are noteworthy.

When $\dot{\theta_0}$ has a non-zero value, it adds or subtracts linearly to the angular velocity of an inverted pendulum, this means that a very large value for $\dot{\theta_0}$ will cause the inverted pendulum to pass both the vertical positions and thus continue to spin around.

Smaller values will cause the pendulum to oscillate, with possibly a larger peakθ than θ_0.

This last observation can be used to plan the single support motion, if the robot was only modelled in the frontal plane. It can simply be calculated how low the entry velocity θ_{y0} must be to ensure that the robot does not pass vertical, and thus falls back towards double support when the step nears completion.

The opposite logic applies if the robot is defined as a Compass walker, that walks in the sagittal plane only. Here the entry velocity θ_{x0} must be larger enough to ensure that vertical is passed.

As a result of the observations on balance the entry angle θ_0 can be defined as $\theta_0 = \pm\theta_e$, depending on the desired path in relation to vertical. It is the same as stating that the inverted pendulum must return to the same height of CoM as it entered Single Support in, but depending on the sign, it happens on different sides of the vertical position.

But as the walking speed is not directly applicable in this equation, it is not yet integrated enough to be used in the prediction of the desired entry angular velocity.

Approximation of the Analytic Integral of the Planar Pendulum.

If the translational average speed of the CoM is provided as reference signal, it can be converted to a step length, using the step length approximation formula in Section 5.2. Given this, the total travel time of the single support can according to the passive walker dynamics, be approximated by assuming that no time is spent in double support.

It is necessary to convert this, time and position information into an entry velocity for the inverted pendulum. For the stable symmetric periodic walk case$\theta_0 = \theta_e$ the angle conversion is straight forward and not shown here.

The entry velocity must only be based on the entry angle and the desired travel time, t. To approximate a function that can do that it is decided to use empirical data from a numerical

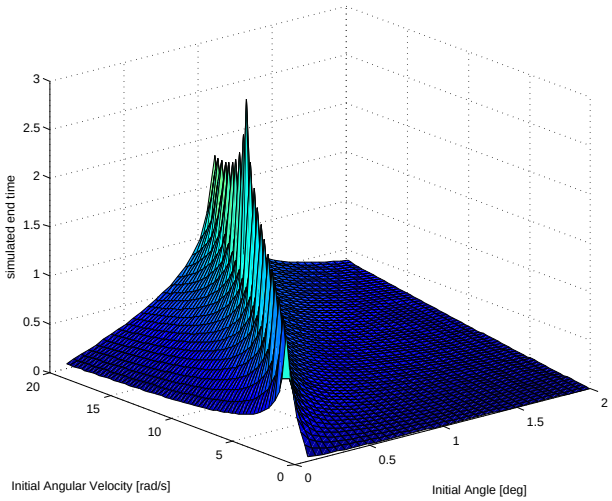


Figure 4.8: The resulting three

simulation, and then fit a function to the behavior of it.

Using the ODE45 integrator in Simulink, a simple planar pendulum is implemented, and solved numerically. The initial angles are then sweeped linearly, and so is the initial velocity, with the stop conditions $\theta_0 = \pm\theta_e$.

As there is three variables, the resulting relation can be plotted as a three dimensional surface, see Figure 4.8 for the result of the sweep.

The plot reveals that a ridge is formed, which looks like a symmetric shape formed by mirrored exponential functions, about a point of infinite growth. This ridge goes towards infinite time, so it is suggested that a relationship between time and the two other variables can be established by using an exponential function, that has different signs, depending on which side of the ridge it should drop towards.

A fraction that could give such a performance is

$$\frac{2}{1 \pm e^{-t\alpha}} - 1$$

Also it looks like there is an angle of the ridge, which can be defined by the entry angle, or the velocity alone.

Hence it is suggested that the approximation could take the form:

$$\left(\frac{2}{1 \pm e^{-t\alpha}} - 1\right) \beta cos(\theta_0)$$

However no suitable α or β could be found that fitted the proposed function, however the error seemed to have a deviation which could be linearized by replacing the $\beta cos(\theta_0)$ with the first integral of motion. And then a good fit could be made by selecting α as π.

By changing the length of the pendulum l it was revealed that α should be dependent, and decreasing as a function of l, and that the curve could be fitted very well by inheriting first part of the integral of motion, letting $\alpha = \sqrt{(\frac{g}{l})}$.

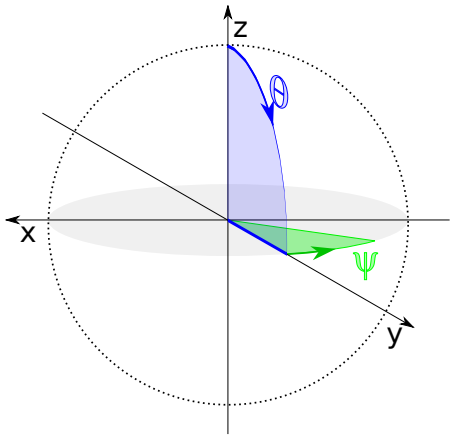


Figure 4.9: The angles used for the approximate solution of a 3D pendulum.

The resulting approximations are when attempting to return to the entry angle $\theta_0 = \theta_e$ by:

$$\dot{\theta}_0 = \left(\frac{2}{1 + e^{-t\sqrt{\frac{g}{l}}}} - 1 \right) \sqrt{2\frac{g}{l}\left(1 - \cos\left(\theta_0\right)\right)} \tag{4.10}$$

and when a compass walker is predicted, the desired $\theta_0 = -\theta_e$ gives:

$$\dot{\theta}_0 = \left(\frac{2}{1 - e^{-t\sqrt{\frac{g}{l}}}} - 1 \right) \sqrt{2\frac{g}{l}\left(1 - \cos\left(\theta_0\right)\right)} \tag{4.11}$$

Transfer to 3D Inverted Pendulum Integral Approximation.

The two Equations 4.10 and 4.11, provides the desired results of the Inverted 3D pendulum (Pencil Model), in the specific cases where the pendulum is expected to move in a vertical plane.

As the rotation about the rod of the 3D pendulum is irrelevant (the ankle of the robot only have a pitch and a roll axis), the model can be reduce to a 2D spherical system. With the choice of coordinates θ denoting the angle from vertical and down, and the angle ψ, for the horizontal angle. See Figure 4.9 for an illustration of the location and orientation of the angles.

When $\psi = \dot{\psi} = 0$ and $\psi = \pi, \dot{\psi} = 0$ the solution to the approximation is given as Equations 4.10 and 4.11, as the pendulum reduces to the planar case.

As a result any multiplications must reduce to one in the planar case, and any summations must become zero in the planar case. When the entry angle ψ_0 is known, it can be rotated to always be zero. This can be done without loss of generality, as it merely rotates the output angle with the same rotation. The added function must be dependent on at least the desiredψ_{end}

A 2D plot can now be created, where θ_0 is kept constant, and the end time, t_{end}, is seen as a function of two free variables, $\dot{\theta}_0$ and $\dot{\psi}_0$, the resulting plot is seen in Figure 4.10.

It can be deducted from the plot that a circular shape can be drawn that fits all the situations where the system actually returns to the same height as it started, for any given time.

The center of the circle is the location of angular velocities, that makes the pendulum stabilize in a vertical stance. The two intersections with $\dot{\psi} = 0$ axis is also known since this reduces the problem to planar case.

Using these informations functions to approximate the initial velocities are found as:

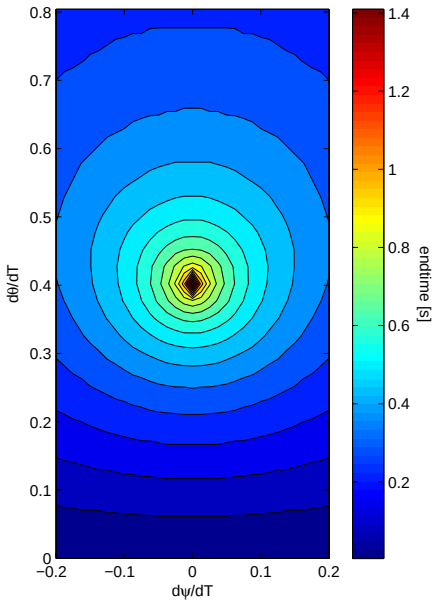


Figure 4.10: This plot illustrated the end time as a function of initial $\dot{\theta}$ and $\dot{\psi}$. It is seen that for a fixed end time the possible solutions forms a circle, and since both intersections with the $\dot{\psi} = 0$ axis known from equations 4.10 and 4.11, the solutions for different ψ_{end} can be found on the circle.

$$
\begin{aligned}
\Upsilon &= \frac{\theta_0}{|\theta_0|} \cdot \sqrt{2\frac{g}{l}\left(1 - \cos\left(\theta_0\right)\right)} \\
\dot{\theta}_{max} &= \left(\frac{2}{1 - e^{-t\sqrt{\frac{g}{l}}}} - 1\right)\Upsilon \\
\dot{\theta}_{min} &= \left(\frac{2}{1 + e^{-t\sqrt{\frac{g}{l}}}} - 1\right)\Upsilon \\
\dot{\theta}_0 &= -cos(\psi_{end})\left(\frac{\dot{\theta}_{max} - \dot{\theta}_{min}}{2}\right) + \left(\frac{\dot{\theta}_{max} + \dot{\theta}_{min}}{2}\right) \\
\dot{\psi}_0 &= -sin(\psi_{end})\left(\frac{\dot{\theta}_{max} - \dot{\theta}_{min}}{2}\right)
\end{aligned}
$$

4.3.2.2 Path Planning

The CoM current position and velocity of the CoM, must be changed to accommodate the exit conditions of the double support state. This is done by utilizing a Uniform Rational B-Spline, that is a specific case of the more general class of splines known as NURBS.

The advantage of using a B-Spline is that the position and velocity can be defined in the start and endpoints, and that a polynomia is used to fit the requirements. Hence a path is always found which transforms the current states to the desired values smoothly.

The B-Spline changes shape dependent on the constraints on the acceleration and jerk. If the jerk is non-zero constant the acceleration changes linearly as a function of time. Thus the jerk becomes the lowest order that can be used as a constant to change the velocity, of a particle in a homogeneous field.

$$\dddot{P} = J$$

As a result the acceleration during the travel of the spline is:

$$\ddot{P} = Jt + a_0$$

And the velocity:

$$\dot{P} = \frac{1}{2}Jt^2 + a_0 t + v_0 \tag{4.12}$$

So the position at time t is:

$$P = \frac{1}{6}Jt^3 + \frac{1}{2}a_0 t^2 + v_0 t + p_0$$

Where:
v_0 and p_0 known, as they are the entry conditions for the double support.

At the time of state change to single support, $t = t_{end}$ we know the desired exit position $P = p_{end}$ and exit velocity $\dot{P} = v_{end}$, as they where found by the model predicted inverted pendulum. The only free variables are now a_0, t and J.

J can however be isolated:

$$J = \frac{6}{t_{end}^3}\left(p_{end} - \frac{1}{2}a_0 t_{end}^2 - v_0 t_{end} - p_0\right)$$

and substituted into Equation 4.12:

$$v_{end} = \frac{1}{2}\frac{6}{t_{end}^3}\left(p_{end} - \frac{1}{2}a_0 t_{end}^2 - v_0 t_{end} - p_0\right)t_{end}^2 + a_0 t_{end} + v_0$$

This in term allows for determining the initial acceleration a_0. Which is unknown as it is immediately after the impact.

$$a_0 = 6\frac{p_{end} - p_0}{t_{end}^2} - \frac{4v_0 + 2v_{end}}{t_{end}}$$

and J can be determined as:

$$J = 2\frac{v_{end} - v_0}{t_{end}^2} - 2\frac{a_0}{t_{end}}$$

The only unknown for forming the spline is now the exit time, t_{end}, and unfortunately this one cannot be derived analytically [32]; It is decided that a linear approximation of the travel time is sufficient, since the velocity is not expected to change much. It is approximated by the distance and the average velocity.

$$t_{end} \simeq \frac{|p_{end} - p_0|}{\frac{1}{2}|v_{end} + v_0|}$$

So as all the constants are defined, the spline equation can be solved for any time t. When t takes the values between t_0 and $t_{end}$the spline patches current conditions with the single support conditions.

5

Implementation

"If debugging is the process of removing bugs, then programming must be the process of putting them in."
—Edsger W. Dijkstra

5.1 Estimation of CoP

The CoP is the weighed sum of all the ground normal forces experienced within a PoS, if all vertices on the PoS is known, it is rather easy to determine it, as the definition alone, provides the equation:

$$CoP_x = \frac{\sum_i F_{zi}\vec{x}_i}{\sum_i F_{zi}}$$

Unfortunately all vertices that has ground contact are not measurable on the physical robot. Instead each ankle is equipped with a single FTS, that registers the effects of ground contact on the ankle. To transform these measurements to estimates of the CoP location, it is necessary to know where the ground is in relation to the FTS, as the forces and torques measured needs to be transformed into the ground plane. The FTS readings are converted to axis aligned forces and torqes by the transformation matrix shown in Section 3.4.

Even more interestingly it is necessary to know if there even is ground contact, as the FTS will register the weight and accelerations of the foot as a very small inverted GRF when lifted from the ground.

This contribution will locate a CoP somewhere, even though it should actually have been undefined.

The global CoP location can be found as the estimated CoP for each foot weighed with the GRF. Fortunately if just a single foot has ground contact, the error due in CoP estimation is by nature very small, as the GRF experienced by a foot with ground contact, greatly outweighs the ones without, in a weighed average.

To calculate the CoP location it is easier to determine the iZMP as it is also defined outside the PoS, and has a simple translation from the FTS measurements. From this the CoP can be easily calculated as the place within the PoS which is closest to the iZMP.

The FTS are assumed (in ZMP calculation) to work as follows:

The sensor is rigidly mounted on the foot, an the forces of the robot translates down upon the foot as if it is planar on the ground. I.e. when standing still FTS_z will be **negative** and close to $\left|g \cdot \frac{m}{2}\right|$in magnitude.

$$\begin{aligned} \tau_{CoM} &= (P_{fts} - P_{CoM}) \times F_{fts} - \tau_{fts} \\ F_{grf} &= F_0 - F_{fts} \end{aligned}$$

Where:
τ_{CoM} is the torque applied at center of mass
$(P_{fts} - P_{CoM})$ is the distance from center of mass to the fts
F_0 is the force measured when the robot is hanging still in the air.

From this the iZMP can be calculated in a plane described by a normal vector, $P_{plane} = [\ 0 \quad 0 \quad P_{plane,z}\ ]$, as follows

$$\begin{aligned} d &= \frac{F_{grf} \times \tau_{CoM}}{\|F_{grf}\|^2} \\ t &= \frac{P_{plane,z} - d}{F_{grf,z}} \\ iZMP &= d + t \cdot F_{grf} \end{aligned}$$

5.2 Determination of Step Length

From the desired walking velocity, it is necessary to determine the desired step length to evaluate how large the inter-leg angle will be at impact, and hence how long time the robot will have to be in single support mode.

A change in walking speed can be achieved in at least two ways, the first being increasing the step frequency. A higher velocity will produce a higher frequency, if the step length is not changed. The other option is changing the step length as this changes the velocity when assuming that the step frequency can be maintained.

According to [20]. Humans have a tendency to change both the frequency and the step length, to reduce energy consumption.

There is not yet developed a simple model which can predict the optimum step length to a frequency (or velocity), for a generic biped system, and instead of re-inventing the wheel, it is decided to fit the choice of step length to the empirically known human behavior. While this is not necessarily optimum for AAU-BOT1, it is considered adequate for the purpose of walking anthromorphically.

Human step length and velocity data was retrieved from [20].

The data is undefined for speeds lower than 0.5 m/s, and undefined above 2 m/s. The upper limit is way beyond the proposed 1 m/s that the robot is destined to walk, but the lower speeds needs to be defined.

Two data fits was done, the first proposed is a standard polynomial fit, which has the drawback of not crossing a zero step length when having zero velocity. The other is a simple power function, which ensures crossing in zero, but has larger tracking error.

The resulting fits are shown in Figure 5.1.

The resulting motions, in the previously undefined areas are for the polynomial, that the robot will have to either stand completely still, or move slowly by taking some relatively long steps (approximately a foot length), allowing it to move quasi statically, as it would be required to stay in single support for a long duration of time. In comparison the power function predicts

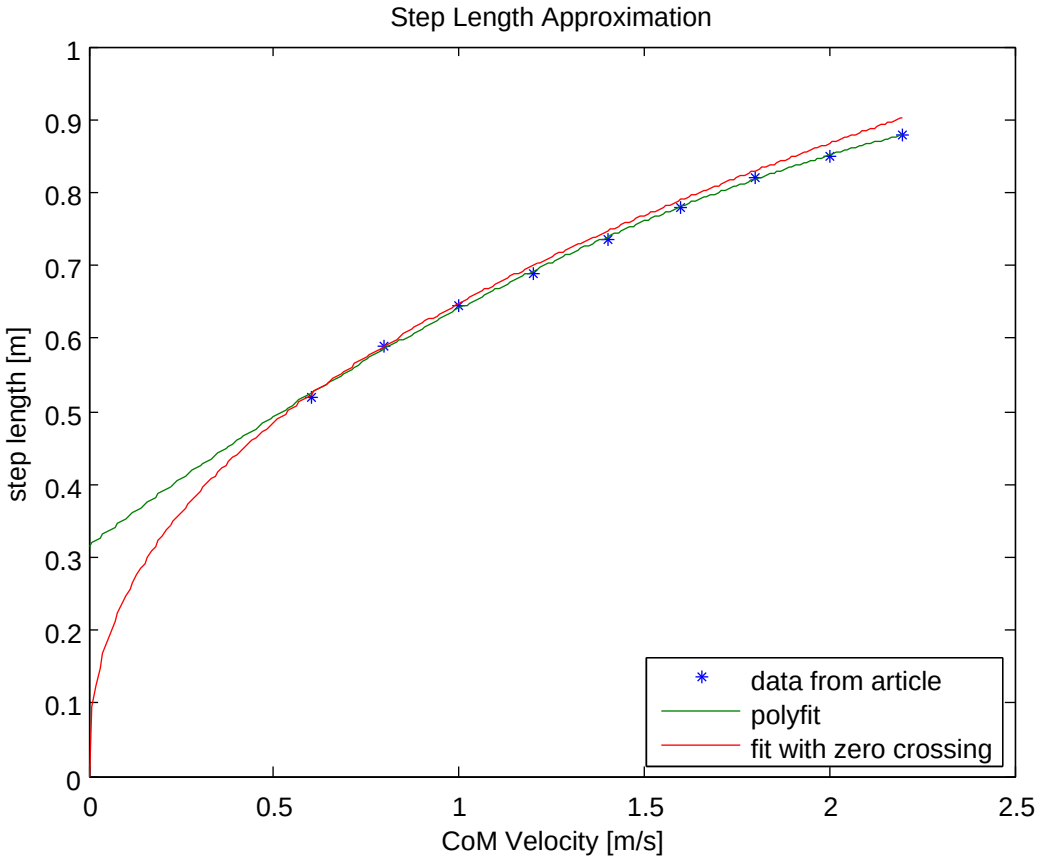


Figure 5.1: Approximated Step Length as a function of velocity.

that very small step lengths are needed to walk very slowly, and thus the robot has very little time to actually accomplish the step.

It is however decided as the rise time is very fast for the power function, that it is acceptable, as the advantage of not moving the feet forward when the CoM velocity is zero, is a desirable feature.

The Power function determined is shown in Equation 5.1.

$$S = \sqrt{.42} \cdot V_{CoM_x}^{.42} \tag{5.1}$$

5.3 State Machine

To make a state machine in simulink a simple enable system block was created which had the limitation that only one state transition can occur during each timestep. Since it is expected to be in any timestep for at least one sample time, this is not considered to be a problem, but rather a feature as it created an upper bound on the execution time.

The state block shown on figure 5.2 is the block that contains the state logic. The block

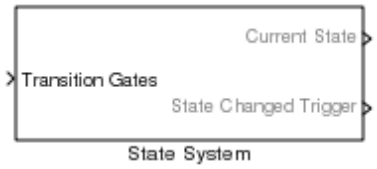


Figure 5.2: State determination block, The content of the block is an embedded matlab code, which implements the Equations 5.2 and 5.3.

takes the transition truth vector T in and outputs the next state, S_+. The previous state is kept internally, whose intial value is a block parameter.

The transition matrix, Γ is likewise a block parameter.

$$\Gamma = \begin{bmatrix} I_1 & S_{-,1} & S_{+,1} \\ I_2 & S_{-,2} & S_{+,2} \\ I_i & S_{-,i} & S_{+,i} \end{bmatrix} \tag{5.2}$$

$$S_+ = \begin{cases} S_{+,1} & for\, S_- = S_{-,1} \vee T_{I_1} \\ S_{+,2} & for\, S_- = S_{-,2} \vee T_{I_2} \vee \neg T_{I_1} \\ \vdots & \vdots \\ S_{+,i} & for\, S_- = S_{-,i} \vee T_{I_i} \vee \neg T_{I_{i-1}} \vee \cdots \vee \neg T_{I_1} \end{cases} \tag{5.3}$$

In this way the ordering of Γ acts as a prioitization in case multiple transitions is possible. I.e. the statemachine shown on figure 4.4 on page 88 is implemented here as:

1. Terminate Controller.
2. Waiting to start or the timer (Collapsed into one for simplicity).
3. Right Foot state.
4. Right to Left Foot state.
5. Left Foot state.
6. Left to Right Foot state.

The conditions for state changes consists of the following 6 statements.

1. No feet has contact with the ground.
2. Both feet has had contact with the ground the last 5 seconds.
3. Spline tracking is done or left foot has lost contact with the ground.
4. Has been in this state for a duration of more than half the pendulum sving time and left foot has gained contact.
5. Spline tracking is done or right foot has lost contact with the ground.
6. Has been in this state for a duration of more than half the pendulum sving time and right foot has gained contact.

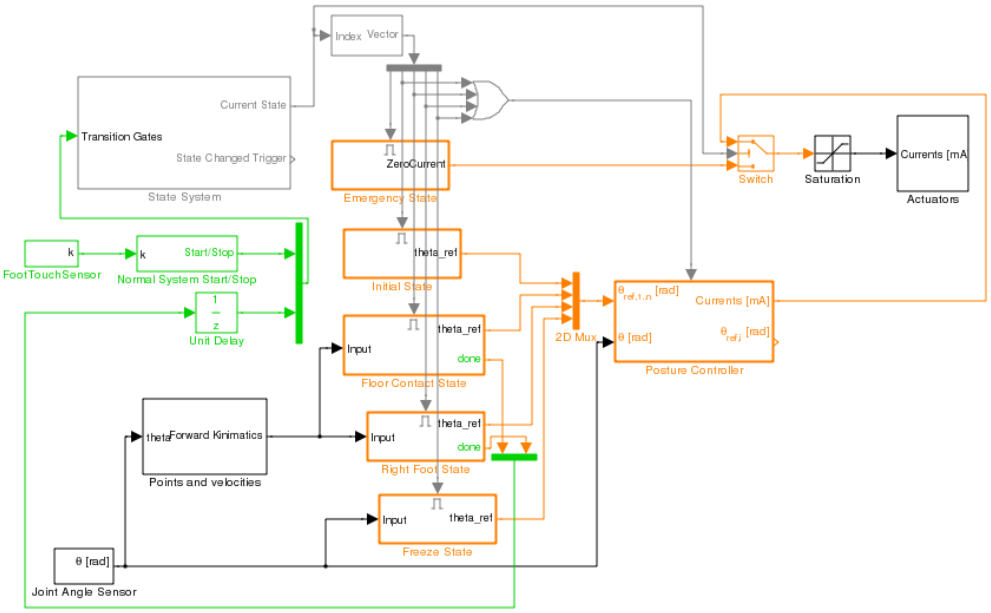


Figure 5.3: Figure showing the final implementation of a half cycle. Green is transitions, Black is input/output, Gray is state and Orange is references or onbounded output.

These conditions is then used to make the following state transitions:

$$\Gamma = \begin{bmatrix} 1 & 3 & 1 \\ 1 & 4 & 1 \\ 1 & 5 & 1 \\ 1 & 6 & 1 \\ 2 & 2 & 3 \\ 3 & 3 & 4 \\ 4 & 4 & 5 \\ 5 & 5 & 6 \\ 6 & 6 & 3 \end{bmatrix}$$

Since all the state transitions that go to the termination state is above any other transition they are given higher priority.

The state is then used to enable simulink subsystems and switch output. Disabling and reenabling a system also causes it to lose internal state varibles[1] which is useful to prevent windup in the posture controller. A final working example is shown on figure 5.3, which can be executed on the OBC.

5.4 Simulator Implementation

The simulator model is available as a library in Matlab/aaubot_simulink/aaubot1_library.mdl, in general objects in the Simulink model have been masked with 3D renderings of the objects, to ease understanding of what the individual object models.

[1] This is actually only true if the enable block is setup correctly

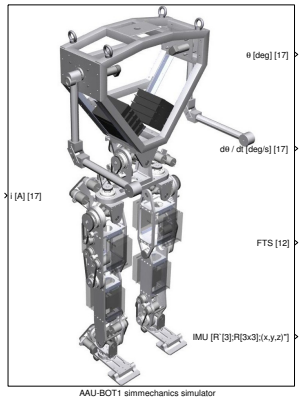


Figure 5.4: The Dynamics Simulator block available in the library.

Robot Object Name	Number of Bodies	Joint orientation with parent (globally)
Toe / Phalanges	2	pitch
Foot / Meta tarsals	2	pitch
Ankle Cross Axle / Tarsals	2	roll
Shin / Tibia	2	pitch
Thigh / Femur	2	pitch
Hip Cross Axle	2	roll
Hip Bracket / Coxa Link	2	yaw
Coxa Girdle / Pelvis	1	yaw
Waist Bracket / low abdomen	1	pitch
Waist Cross Axle / vertebral flex	1	roll
Torso / vertebral	1	6 DoF to world
Arms	2	pitch

Table 5.1: The relation between SimMechanics bodies, and the anatomy of the robot

The model structure, and thus indirectly the kinematic properties has been extracted from Solidworks using the SimMechanics Link. The link tool also extracts mass and inertia matrices and joint axes orientations. In this manner all the forward kinematics is indirectly build into the positions of the individual bodies coordinate systems and the joints that relate them.

The robot structure is setup as rigid bodies, that are located such that they align certain coordinates. The location of those coordinates, is the location of the joints in zero position, when described from the individual rigid body. Constraints are applied as joint definitions, they determines which of the 6 possible DoF in a joint that may be manipulated, and which that may not, and thus forces a relation of the algined coordinates, if one of the bodies are rotated or moved.

The dynamic model which is the core of the simulator, is implemented as a single Simulink subsystem block, it takes a vector containing the current reference for the actuators, and outputs the data which is expected by ideal sensors, see Figure 5.4.

The Dynamic model consists of 20 individual rigid bodies. One for each kinematic link on the robot, see Figure 1.2 in Section 1.3, and one for each DoF above 1 for a joint, see Table 5.1

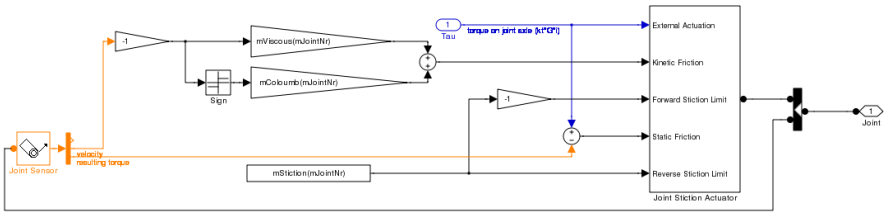


Figure 5.5: The friction model used in the SimMechanics simulator.

All the joints that connects the rigid bodies, are chosen as a revolutionary joint to the kinematic parent, except the torso body, which is connected to the ground through an unconstrained six DoF joint. An unconstrained joint dynamically removes the ground from the simulator model dynamics, and only uses it for transformations to and from world coordinates. Each joint is attached to a sensor block, and a joint friction actuator block, allowing extraction of data concerning torques, angles, angular accellerations, etc. to the conventional Simulink environment.

The joint friction actuator block, is a state machine that locks or unlocks a joint if the joint test torque, and the joint velocity passes a stiction threshold. When unlocked a viscous and Coulomb friction is applied, the aaubot library features an improved version of this block, and it is this model which is implemented in the dynamics model. The content of the library block is shown in Figure 5.5.

The relation between the blocks in the dynamics simulator model can be seen in Figure 5.6.

The ground normal force is modelled as springs and dampers connected to multiple vertices on the feet. It is implemented using zero crossing detection, and thus implements variable time steps.

Ground friction is implemented as a continously defined function, that approaches the classic friction model. This model has been chosen, as the simulation can otherwise run into algebraic problems, due to the non-convex properties of a classic friction model when the feet are standing still. The ground model implemented is illustrated in Figure 5.7, and the friction model is found in Equation . Notice that the friction formulation is one dimensional, it is however duplicated and applied in both axes, to provide a reaction in the ground plane. The error that arises when the foot slips in a non axis aligned direction is expected to be tolerable.

$$F = \begin{cases} -\frac{v}{|v|}\left(F_C + (F_{brk} - F_C)\, e^{-c_v|v|}\right) + f \cdot v & for\, v \geq v_{threshold} \\ v\frac{\left(f \cdot v_{threshold} + \left(F_C + (F_{brk} - F_C)e^{(-c_v v_{threshold})}\right)\right)}{v_{threshold}} & for\, v < v_{threshold} \end{cases}$$

Where:
F is the friction force
F_C is the Coulomb friction
F_{brk} is the breakaway force, i.e. the force that is required to overcome both stiction and coloumb.
f is the viscous friction coefficient
v is the velocity of the foot relative to ground
c_v is the coeffecient for the logaritmic drop from stiction to pure coulomb.

When the velocity is below a threshold value, a straight line is formed that crosses zero, but is not vertical. This approximation to the friction model, allows for very small motions proportional to the forces experienced, when the foot is moving slowly, but is much faster than the classic model.

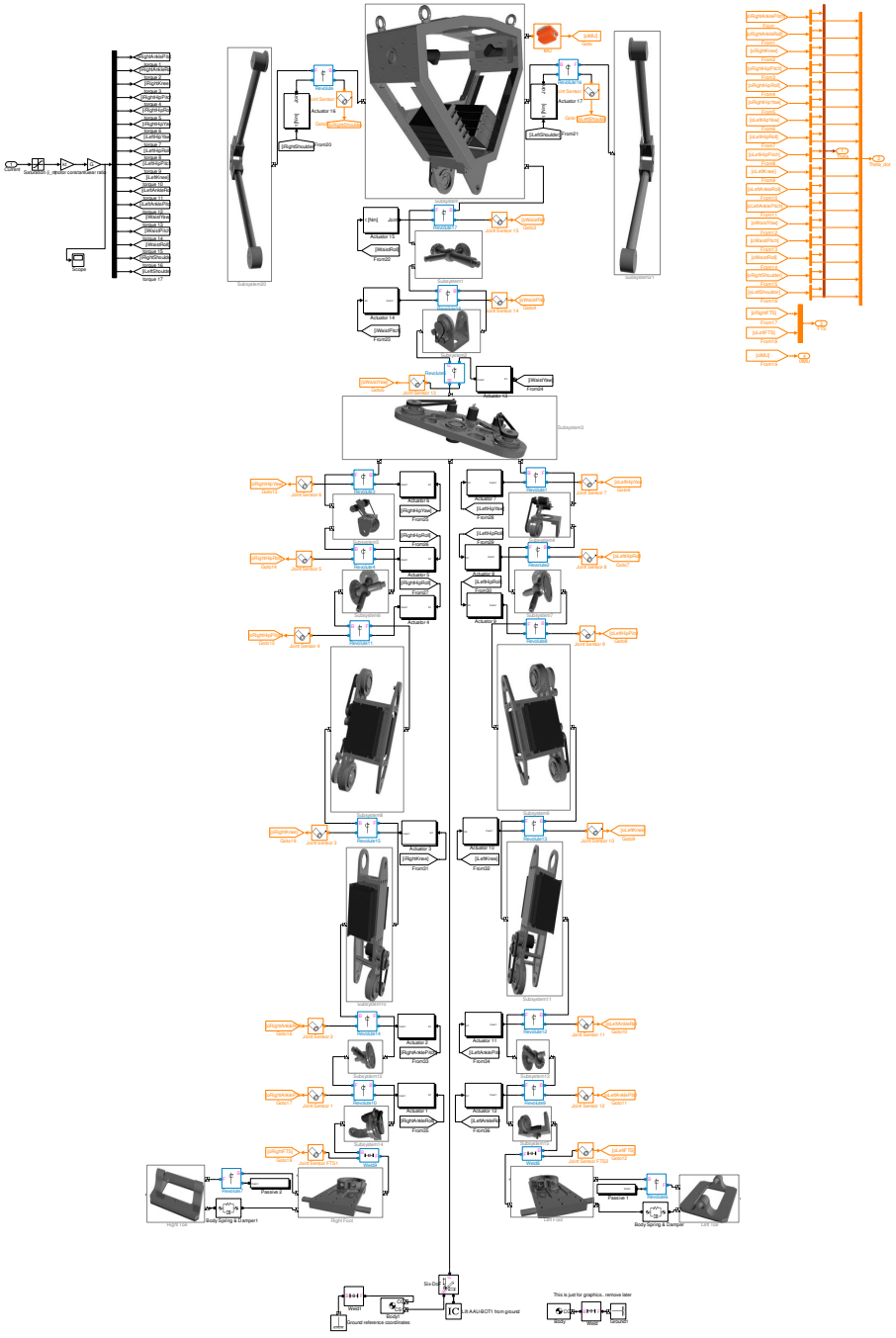


Figure 5.6: The Objects in the dynamics simulator block available in the library.

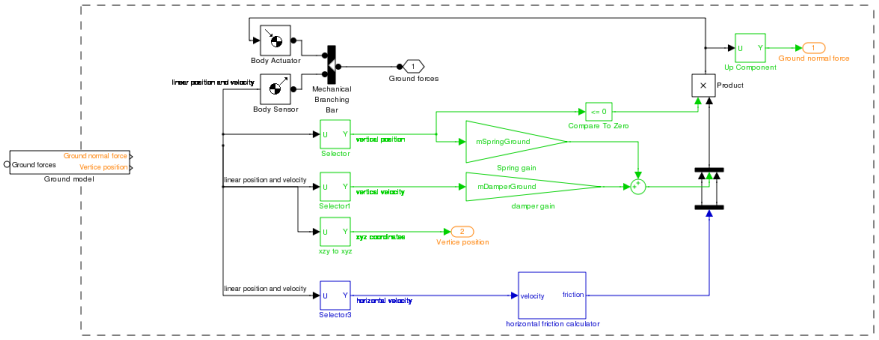


Figure 5.7: The ground model used in the SimMechanics simulator. The green colored blocks calculates and applies ground normal forces, as a spring/damper. The friction is continuously calculated, but only applied if the ground normal force is non-zero

5.5 Visualization implementation

While the onboard computer uses UDP from transmission. The implemented version accepts both UDP and TCP, funneling the data into the same path in the system.

When a client connect to the palantir server a Client object is created to hold everything associated with this client making it easy to remove one or more clients later.

Each Client has one or more lists as described in the palantir protocol, with one being the most common. For each list a Timestamp instance is created, and for each value in the list a Datalog instance is created.

When the list is updated each value, which may consist of more than one float, is added to the appropiate DataLog instance, and the Timestamp class is updated with the timestamp for that update.

The Timestamp class responsibility is to match a given time to an index, which can then be used by the Datalog class to return the zero-order-hold value at any time. The ability to skip both forward and backward in time is crucial for gaining both acceptable performance and replay features.

An additional efftect of creating the update list is that the value is bound to function which is executed during the main loop, which looks like this:

```
while true:
sleepIfRunningTooFast() #A max framerate limits the amount of system reasourced used

    for client in clients:
        for list in client.updatelists:
            list.updateTime(getPseudoTime()) #Skip values
            for (object,value) in list.bindings
              updateVisual(object,value)
```

Since the getPseudoTime() is a real time clock with an offset and a scale, the impact of a too slow computer is that it will just lower the framerate, by skipping many more values.

The scale and offset of the pseudo clock can be controlled using GUI elements, as seen on figure 5.8, thus enabling the user to skip in the data, play slowmotion or even make the time flow backwards.

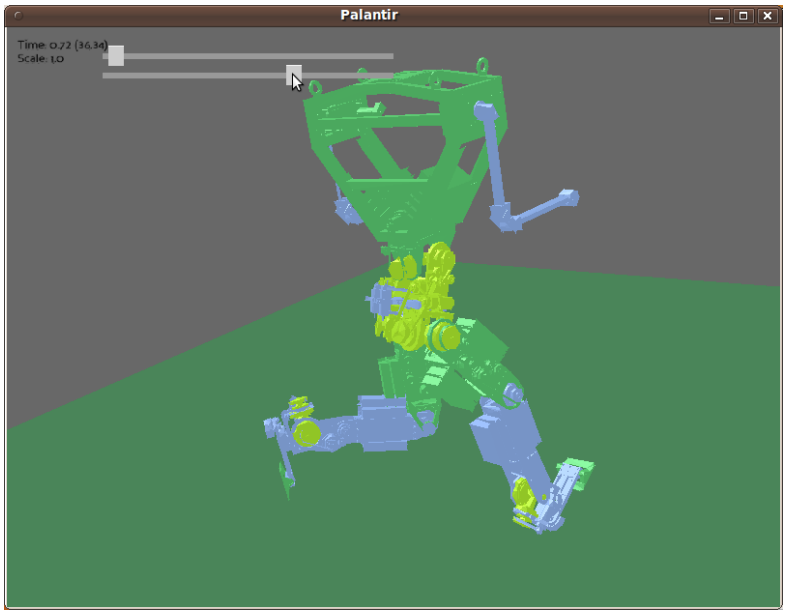


Figure 5.8: Screenshot of Palantir.

6

Results

"The joy of engineering is to find a straight line on a double logarithmic diagram. "
—Thomas Koenig

6.1 Verification of the Free Move PID Regulator

The PID regulator implemented, was first tuned for moving joints when the robot did not stand on the ground. It has been tested by applying a reference signal to it.

The signal has been chosen as a ramp signal, with symmetric return to zero, where each joint is tested separated in time. The reason for this choice of test signal is that the controller is tuned using step responses, which is a very different type of signal than the ramp. The actual trajectory was from research on human trajectories expected to be similar to either ramps, sines, or some combination of them.

Figure 6.1 shows the test signal and the responses as a function of time, the suppression of dynamic cross couplings are only indirectly shown, as the only joint which moves is the in general the one being tested. Should cross couplings be added from multiple joints in motion the effect is not tested, by this experiment.

In general the joints was tested individually according to their location in the joint index vector. The joint index vector contains the 17 actuated joints, and is ordered according to the old kinematic vectors. This information is not otherwise used, as the joints are often just indexed by name, however the index vector is shown in Table 6.1, for easy translation of the order in the figures.

It should be noted that the roll actuator on the waist does not have enough power to rotate the entire lower body the tested five degrees. Instead the rotation on that joint is achieved by counter-moving the hip roll axes. which in the figure produces a double spike.

The effect of counter-moving the hip joints is actually a positive consequence of the dynamic cross coupling effects, but when utilizing them, there always exists a risk of destabilizing the system if the controllers are not tuned properly. So this part of the test was first added late in the process, when all other parts of the test was verified, as this particular part of the test could gain full attention from the operators.

The errors in tracking is shown in Figure 6.1. It is clear that the joints in general are trailing the reference, but that the absolute error does not grow beyond one degree in worst case.

Joint ID	Joint Name	Joint ID cont.	Joint Name Cont.
1	Right Ankle Roll	10	Left Knee
2	Right Ankle Pitch	11	Left Ankle Pitch
3	Right Knee	12	Left Ankle Roll
4	Right Hip Pitch	13	Pelvis Yaw
5	Right Hip Roll	14	Waist Pitch
6	Right Hip Yaw	15	Waist Roll
7	Left Hip Yaw	16	Right Shoulder
8	Left Hip Roll	17	Left Shoulder
9	Left Hip Pitch		

Table 6.1: The old joint index vector.

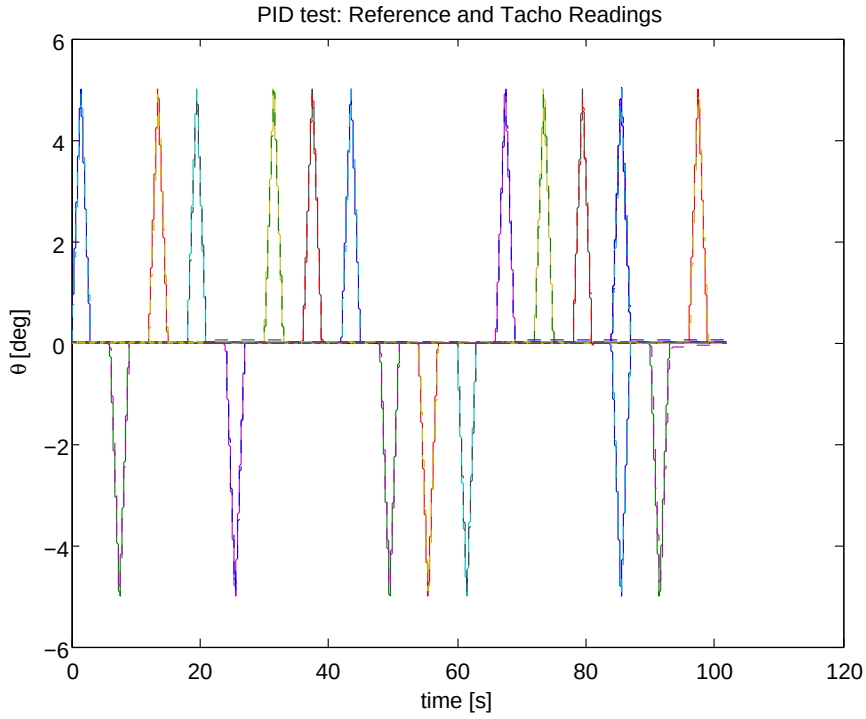


Figure 6.1: The reference signal, and the measured joint responses, for the entire experiment. Signals are applied to the joints in the order of their appearance in the joint index vector6.1.

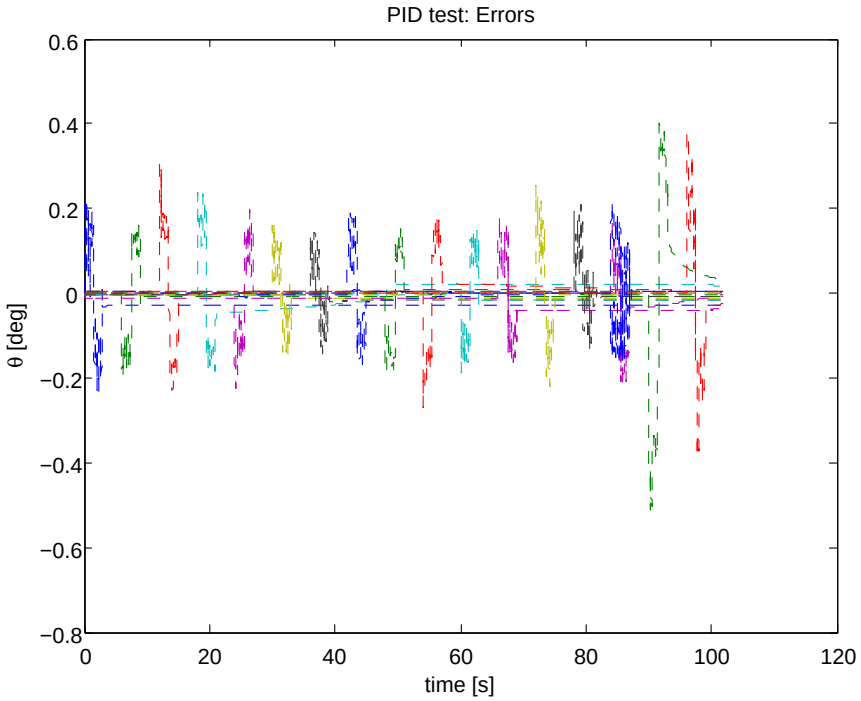


Figure 6.2: The error in tracking. Notice that the error is nearly constant when the ramp is being tracked, this indicates a small delay in comparison with the reference signal. Each spike in the error relates chronologically to the joint index vector6.1

6.2 Pitch Verification Tests of PID Regulator

It has been confirmed that the PID regulator can track a ramp, when there is only limited cross couplings. It was afterward also tested with a sine wave, where some dynamic cross couplings are inferred by moving multiple joints at the same time, this section illustrates the results of that test.

6.2.1 Test While Hanging in Torso

To avoid damaging the robot if anything went wrong, only the pitch angles were tested in this manner. The test rotated the hip, knee, and ankle pitch axes synchronously on both legs. The leg was retracted and then extended, in a vertical position. It can be argued that a complete test should do all combinations of joint rotations possible, and extend and retract the legs with different angles on the hip to fully test for dynamic cross coupling suppression. This was not done, as the waist joint is underpowered, and it was not desired to overload that actuator for a long duration of time.

The signal applied to the knees is a sine wave, offsetted positively by half the peak-peak amplitude. The signal applied to hip and ankles are the same signal multiplied with $\frac{-1}{2}$, as this retracts the leg, and afterward extends it.

The test is run multiple times, and visually it is confirmed that the robot executes the entire motion without becoming destabilized. However the joints do have high frequency low amplitude shakes, when moving. It can appear as if the controller is destabilized by applying a sine wave,

but this is not the case, as the error never grows beyond what was found on the ramp, and is suppressed when the sine is close to a constant value for a short duration of time.

The effect is replicable by rotating the knee only, and it is suspected that the origin of it is the combination of a linear controller on a joint which is far from linear with regards to friction, the knees have been shown to behave poorly with regards to linearization, and to have the largest problems with regards to friction modelling.

The dynamic effects from the knee tracking errors are transferred to the other joints by dynamic cross couplings, and they must also suppress the errors that arise due to the dynamics from the knee. The tracking of the sine is illustrated in Figure 6.3 where this effect is clearly identified.

It can also be seen that the PID controller is able to sustain a relatively low angular error, even when the cross coupled dynamics are inferred on the joint.

Another plausible explanation, is that the linear controller is acting proportionally to the error, and as a result the error needs to be at a certain amplitude before the controller delivers the necessary power to overcome the friction and dynamics in the individual joints.

6.2.2 Test While Standing on Ground

The signal that was applied to the knees was a sine wave, offsetted positively by half the peak-peak amplitude. The signal applied to hip and ankles were the same signal multiplied with $\frac{-1}{2}$.

A result of that particular choice is that the foot is kept in the same planar orientation as the waist, as long as joint limits are respected. As the thigh and shins does not have the exact same length, the foot is though moved relatively to the waist in both the x and z axes. Another way of thinking about this feature is that if the robot was standing on the feet, the upper body of the robot would be lowered towards the ground, and then after wards lifted until the robot is in zero position.

Testing this requires a different set of PID gains, as the dynamic loads on some of the joints are significantly different than what was experienced when the robot was hanging in the torso. Apart from this, when the PID has been re-tuned to stand on the ground, similar results are achieved as was seen when hanging in the torso. The tracking errors are in fact a bit smaller. A video of the experiment can be seen on the CD as Videos/skovs.m4v.

The reduction in errors, is primarily attributed to the dynamics that ground contact infers, as this can dampen errors by linking the two legs via frictions to each other.

6.3 Walk Test

The pictures in figure 6.5 shows the robot taking a single step. The resulting ZMP is shown on figure 6.6. The experiment is started in the air and the robot is then put down on the ground; After 10 seconds the robot starts to accelerate towards the right foot. The is also seen on the ZMP plot where ZMP moves near the right foot. At $t = 20s$ it is supposed to lose ground contact and the robots becomes marginally stable, where it sways from side to side untils it settles. ZMP during this moves unexpectly out of PoS, which either can mean that the posture controller is working against the upwards fall, or that the foot is not parallel with the ground, and thereby voiding one of the assumptions for the ZMP calculation. When the left foot regains contact the ZMP moves to a point near the middle of the two feet. Both in the start and in the end of the experiment ZMP is below the left foot, but this is the where the robot is either raised or lowered, and it is very likely that one foot gains or loses contact before the other.

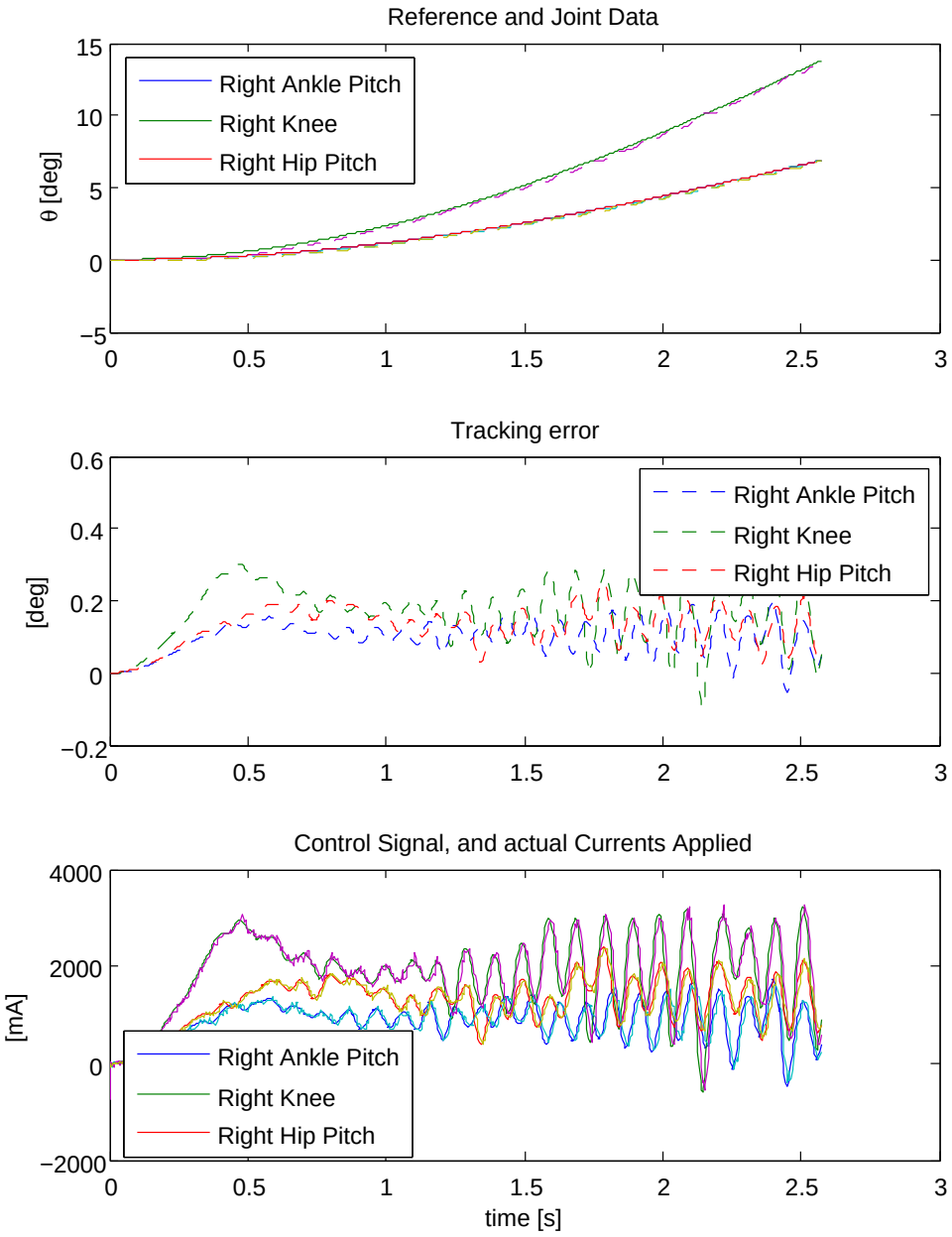


Figure 6.3: The beginning of the experiment, where the legs are retracted in the air. The focus is laid on the beginning of the experiment as a suitable zoom was needed to show the tracking error. The error is similar or less in amplitude for the entire span of the sine. The

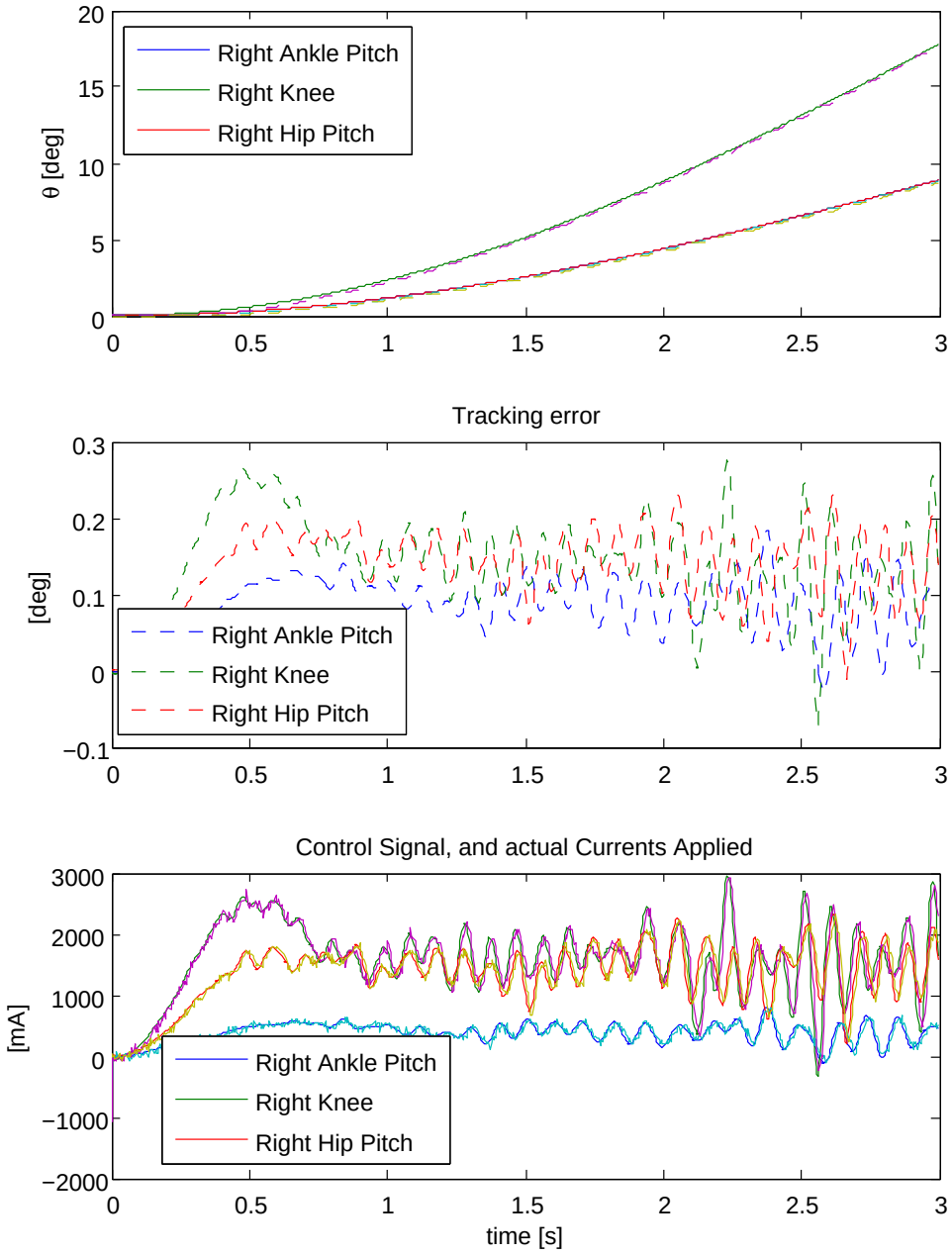


Figure 6.4: The beginning of the experiment, where the legs are retracted in the air. The focus is laid on the beginning of the experiment as a suitable zoom was needed to show the tracking error. The error is similar or less in amplitude for the entire span of the sine. The

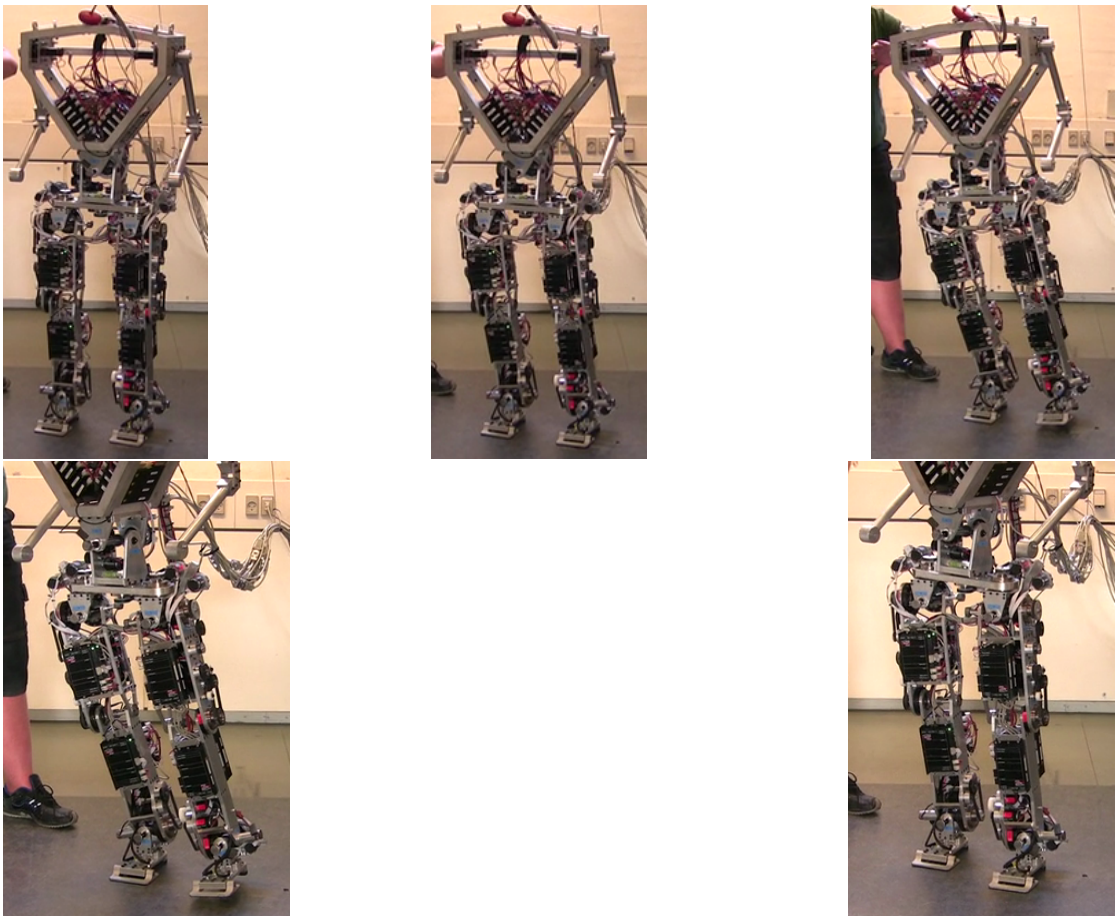


Figure 6.5: Pictures of a step. The full movie can be seen on the CD as Videos/firststep.m4v. From Top-left is corresponds to $t = 5s$,$t = 16s$,$t = 25s$,$t = 41s$,$t = 60s$

While the experiment only had a single step some important conclusions can be deduced from it; The overall strategy of accelerating the robot to gain single stance is possible, and the calculation of ZMP using a FTS can be unstable when switching from double-stance to single-stance.

6.4 Test of Inverse Kinematics.

To test the Inverse Kinematics, two tests are set up.

The first is to determine how large deviations from the forward kinematics that can be expected, and the second is to determine if the Forward/Inverse kinematics can be applied directly in the control loop as intended, this is very important, as they play a crucial part of tracking the trajectories formed by the spline and inverse pendulum generators.

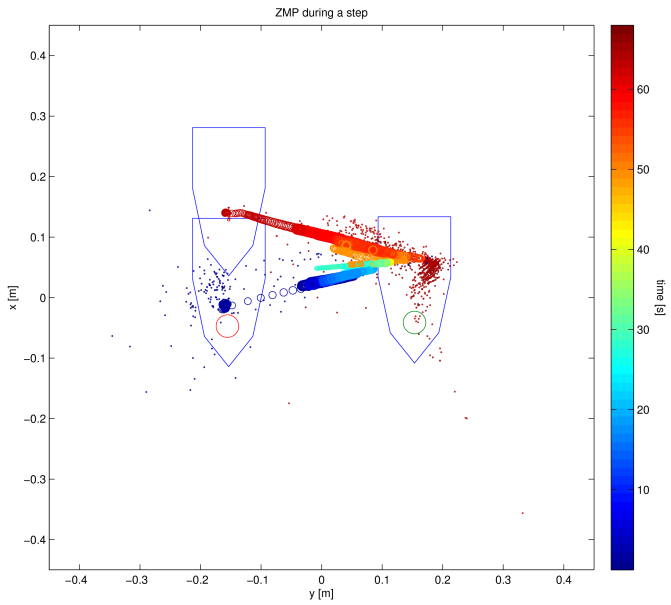


Figure 6.6: ZMP during the step. The size of the circles indicate the magnitude of the GRF, and the two large large circles is the placement of the FTS.

6.4.1 Forward and Inverse Kinematics Test.

The first test is a test, to see if the inverse kinematics can recreate input angles, when the only data available is the CoM, the ankles, and the hips locations. The setup is shown in Figure 6.7, and the results are shown in 6.8. In the results graphs each signal is added with the index vector number, to seperate them vizually from eachother. It is clear that the "Difference" graph in 6.8 is very close to being constant and only offsetted with the joint number. Hence there is very little deviations from zero.

In fact the worst case peak error is 0.002 [deg].

6.4.2 Kinematics in the Loop Test

The second test is to apply the two kinematics blocks in the control loop. For this purpose, the coordinate systems are changed to have an origin in one of the feet, before the reference for the CoM is generated, which consists of three sine waves around the inital CoM.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.10 \\ -0.02 \\ 0.60 \end{bmatrix} + \begin{bmatrix} 0.03 \\ -0.03 \\ .004 \end{bmatrix} \sin\left(\begin{bmatrix} \frac{4}{35} \\ \frac{4}{15} \\ \frac{2}{5} \end{bmatrix} t \right)$$

On figure 6.9 the resulting ZMP can be seen. When the system is moving slowly as in this experiment the real GCoM should be near the same point, and there indeed seem be a good correlation between the two points, and the using filtered feedback information from the ZMP a better tracking might even be possible.

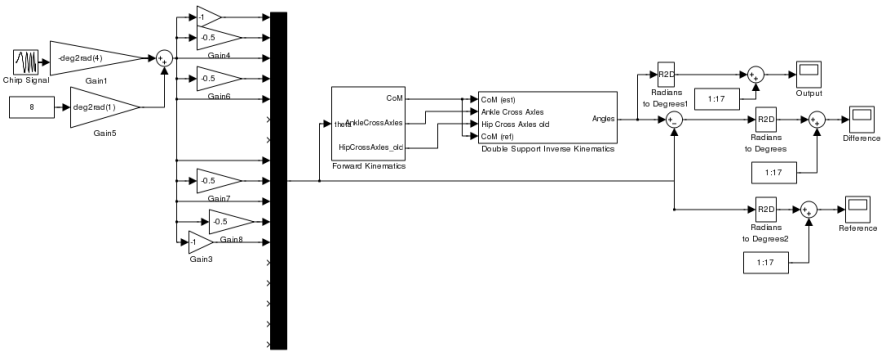


Figure 6.7: The model used for test of the Inverse Kinematics. The three graphs Output, Difference and Reference can be seen in Figure 6.8 .

The reason that the ZMP is not included in the feedback loop is that the bug report J.7.3 has not been solved. A control system should be able to cope with the FTS sometimes being in an errornous state, either by solving the bug so the errornous state is never entered or applying some fault detection that quickly can detect the problem and bring the system to a safe state.

6.5 Walk Test with Dynamic Reference

The second walk test was made using the full state machine, with inverse kinematics. The experiment would consist of two phases, one hanging from the wire and a second where it was standing.

When the the experiment was first executed the system was very quickly stopped since it became unstable, and a final run was made with lower proportional term on the knees. The results from the final run is the one that is seen on the figure 6.10 and 6.11.

From the graph of the knees it is clear that the system no longer can track the reference during the double support phase, and it overshoots during the single support phase, which indicates that it is not possible to track this reference without altering the structure of the posture controller, like adding a deriative term or changing to a controller that compensates for the cross coupling. Many of the other joints become unstable due to the cross couplings in the system.

Adding a deriative term to the posture controller could reduce the overshooting without making it possible to make a make aggresive controller for small pertubations, but without applying a filter would render the system unstable as well due to noise from discrete differentation of the input signal. Such a filter is not made during the timespan of the thesis writing since the problem was discovered too late.

It might be possible that if the robot had contact with the ground the system would have been stable due to the large natural dampening effect of frictions and heavier joint loads but the experiment was deemed to risky to preform while the robot was standing.

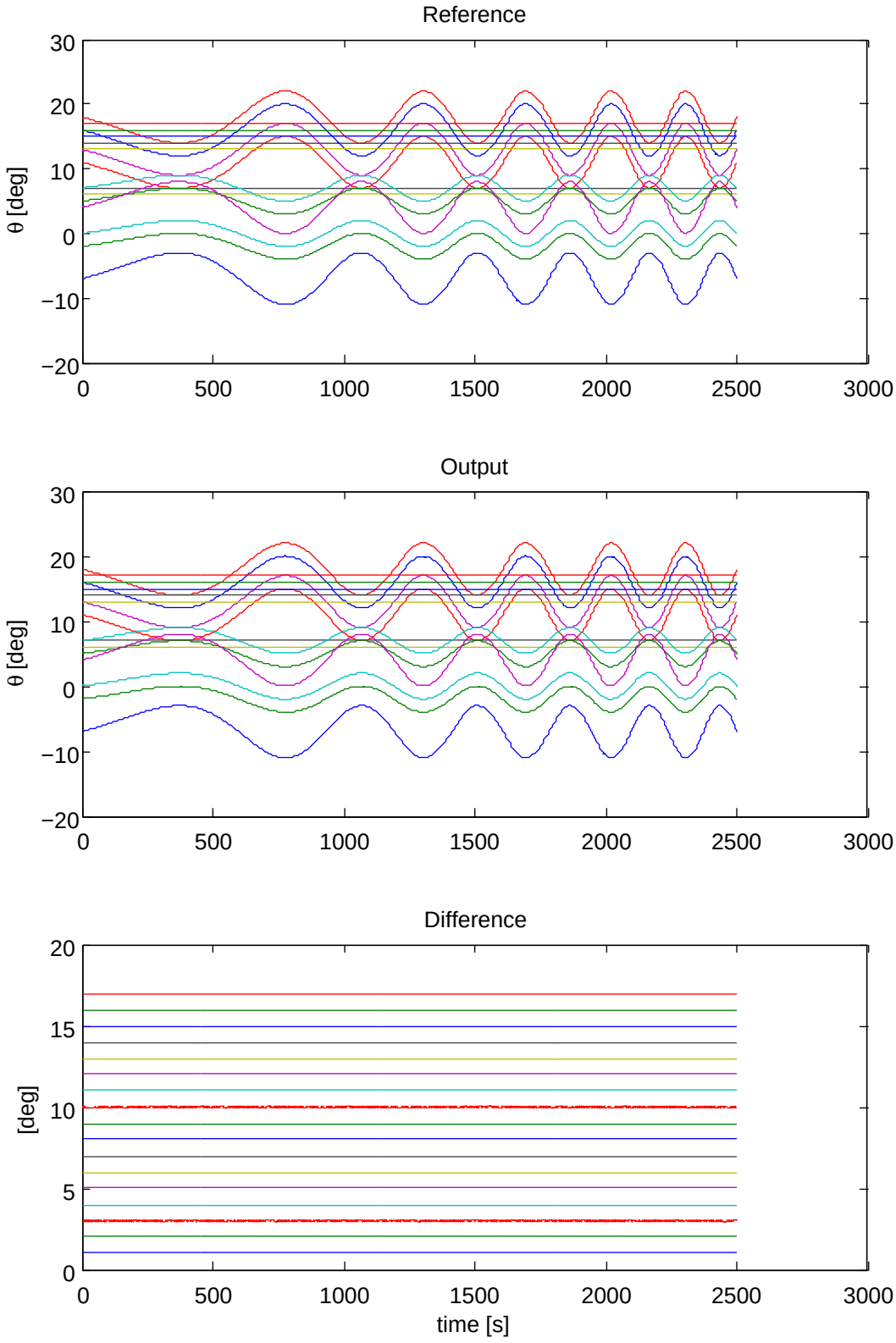

Reference
θ [deg]
30
20
10
0
−10
−20
0
500
1000
1500
2000
2500
3000
Output
θ [deg]
30
20
10
0
−10
−20
0
500
1000
1500
2000
2500
3000
Difference
[deg]
20
15
10
5
0
0
500
1000
1500
2000
2500
3000
time [s]

Figure 6.8:

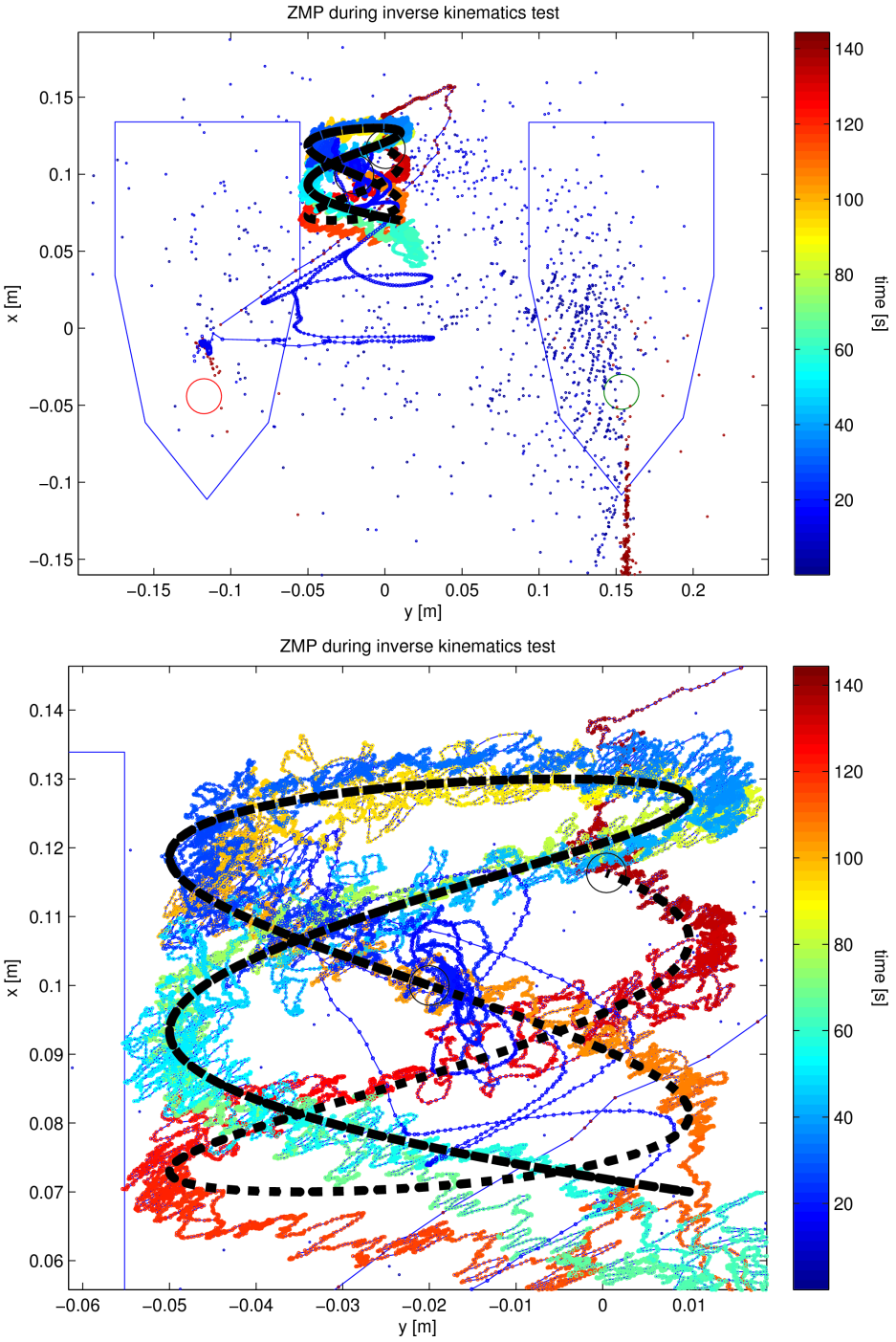


Figure 6.9: Two plots of the same data with different windows. Black is the GCoM reference, the coloured dots are the ZMP during the experiment. The straight blue line is the left foot. There is two black circles where the reference starts and ends. Note the the reference line from the sharp end in the bottom right is double as is travels both back and forth along this line. The unconnected ZMP dots is when the total ground reaction force is less than $10N$

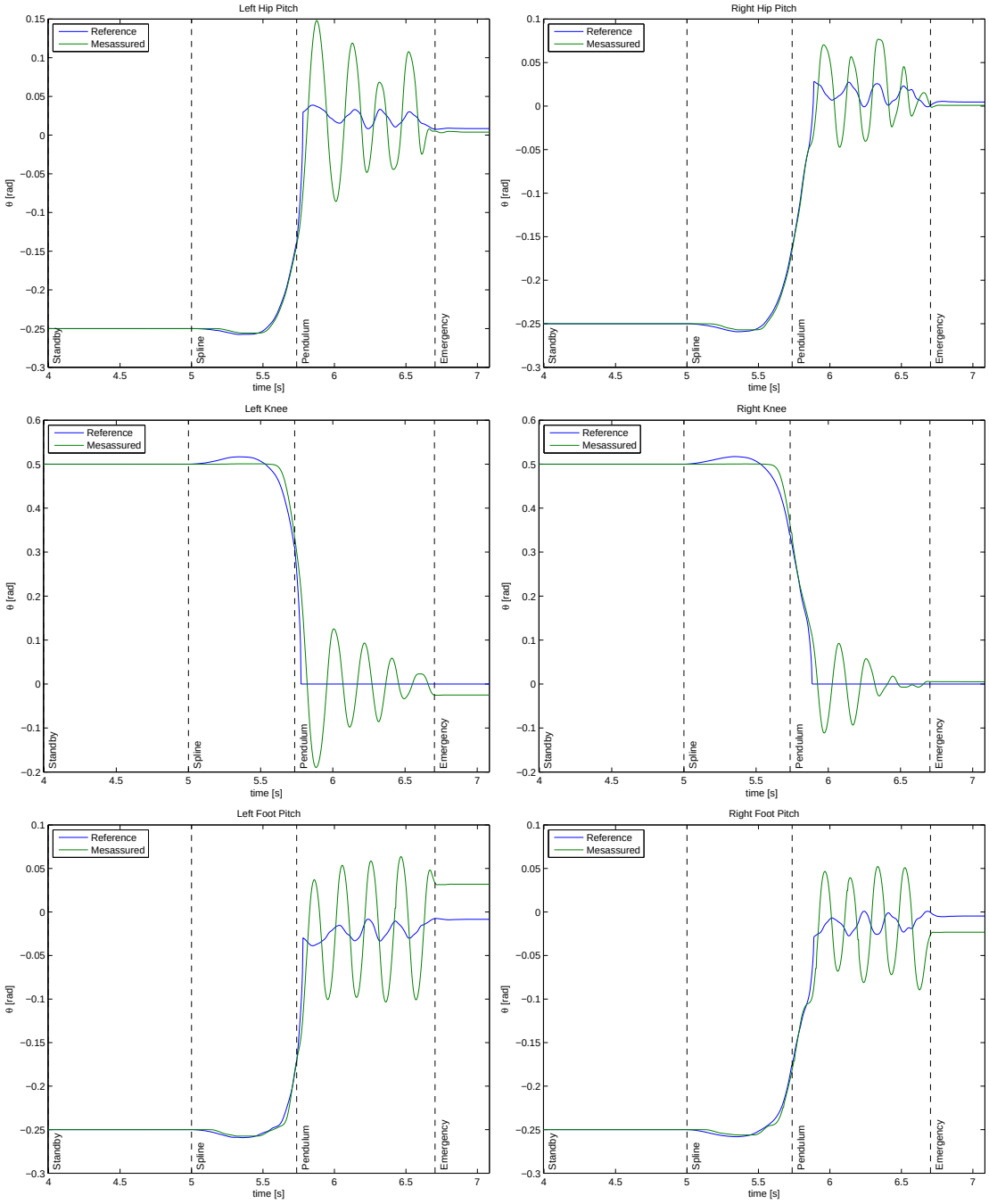


Figure 6.10: Pitch during experiment. The y-axis has different scales. The dotted lines represent state changes.

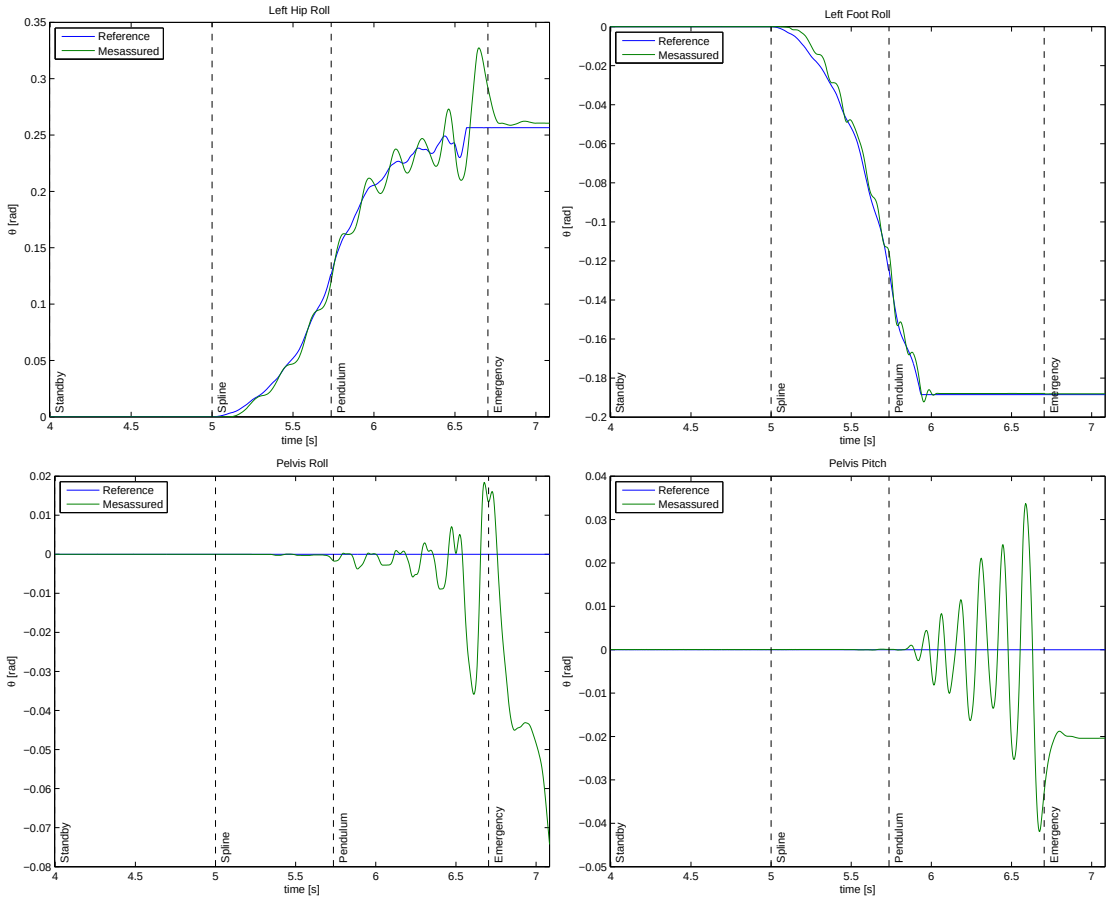


Figure 6.11: Roll angles during the experiment. The y-axis has different scales. The dotted lines represent state changes.

7

Epilogue

"Science is built up of facts, as a house is built of stones; but an accumulation of facts is no more a science than a heap of stones is a house."
—**Henri Poincaré**

7.1 Conclusion

It is has been shown in this thesis that it is possible to walk quasi-statically with AAU-BOT1. At the time of report completion, the robot has been ballistically thrown from a double support phase to a single support phase, by using the approximated dynamics model determined by an inverse pendulum.

Due to the simple posture controller design, hard ground impacts can render the inner loops unstable, and it could not be attempted to walk fully dynamically. Instead a soft landing is applied, which can be completed without invoking a hard impact, and a double support phase is regained.

The resulting gait is a hybrid between a static and a dynamic gait, as the swing up of the inverted pendulum is done dynamically, but the forward motion of the robot in the saggitale plane is done as a static trajectory. Such type of gait is called quasi-staticcally.

The inverted pendulum model has been extended to 3D to support a fully dynamic walk, both in prediction and in tracking. The necessesary motion for intering the inverted pendulum mode with correct speed, is ensured by a spline, and inverse kinematics. This has been applied to the hardware, but the posture controller was not able to track the references generated for some specific joints such as the knees.

Similar motions as predicted by the spline can be applied to the roll joints alone, and the quasi-static gait is achieved. This is interpreted as a proof of concept for using a spline and an inverted pendulum for model prediction on AAU-BOT1.

It is expected that a non-linear improvement of the posture controller, can compensate for the errors regarding a large step on the input, and make the robot able to walk fully dynamic.

The implemented system does not handle the push of, or the heel impact, but is designed for walking dynamically with a flat foot approach. In relation to human-like walking the most critical deviations is the lack of push off, as new research has shown that barefooted running avoids heel impacts altogether[11], and that it can also occur for barefooted walking.

The designed system does however treat the ankles as if they where passive joints, which is actually what would be found on the robot if the toes where used. A system with a passive joint in the loop, is called underactuated, and a robot like this is called a redundant manipulator system. Redundant systems are notoriously difficult to create inverse kinematics for, as there often exists multiple solutions, and underactuated systems are a challenge to control, as one or more of the states are not controlable, but may be stabilizeable.

The inverse kinematics applied must be modified to take into account the toe, if a human like push off phase is to be achieved, however the underactuated part, should already be handled by the implemented control scheme, as the current approach mostly assumes that the ankle is already acting as an underactuated joint.

It should though be noted that the current tracking of a virtual passive joint that is applied, actually aids in dampening effects of an error in the inverted pendulum entry velocities, where as a true passive joint, would not be so forgiving. However by utillizing the push off, the duration of the double support phase is extended, and thus the amplitude of an error in the entry velocity should be much smaller, as there is more time to track it.

A model was determined which could estimate the ZMP and CoP location, allowing for balance control. It was shown in multiple experiments with slow movements that the location of the GCoM which was determined using forward kinematics, correlated with the location of the ZMP. This makes both models plausible. The ZMP estimation is however not placed in the control loop, as the FTS data can be unreliable, and that the estimation places the ZMP at the wrong location if the foot is no longer planar with the floor. A better sensor for determining what the robot is doing, with regards to balance is the onboard IMU, as this is not prone to errors concerning foot and grond alignments.

7.2 Future Work

Even though a form of dynamic walking has been established and applied in this thesis, the robot is not walking anthromorphically at the time of completion. To do so requires a strategy for solving inverse kinematics when in the push off phase, and a better posture control strategy.

Before this is implemented it should also be realized that there are plenty of known software bugs, of which some could appear at a critical time, when all of the robot is in motion, and then destroy the robot as a consequence. Many of the bugs can be solved.

In this section the most critical suggestions for future work on AAU-BOT1 is presented.

7.2.1 CAN Driver Update

The current CAN driver is not optimal for a controller if it should contain any error handling. The EPOS reports errors back to the OBC such as positive or negative limit errors, or under voltage errors, that all indicates that the system should be stopped. Write errors are also reported, but many of the errors such as not responding, are not handled.

The driver only recognizes known errors, or “expected” errors which are then logged, and not more is done to it. This group has implemented a watcher that collects those reports from the log, and alerts the operator, and even if some of the errors such as CAN write error occurs repeatedly, shuts down the controller. This is not always preferable.

Consider the case where the robot is standing on one leg, and one of the joints i.e. the arm reports CAN write error, and this disengages the controller, and shuts off the actuators on the legs.

If the CAN driver was written as a callback system, errors could be fed into the controller, and handled directly, and it would be possible to determine the severity of the error. It would

also be possible to detect if an EPOS was unresponsive at start up, and then avoid initialization of a motion that could damage the robot.

7.2.2 FTS Driver Update

The current implementation of the FTS drivers have a multitude of known bugs. some of them reveals themselves in a manner that appears as hardware errors. This have made debugging of the controllers difficult. The driver is prone to a behavior that resembles what would be expected from an overflow on the data buffer, and have problems being restarted if the entire kernell is not rebooted.

Such problems should be solveable, if they where known at the beginning of a project. For this thesis the error was isolated to be a driver bug very late, and it was not possible to reserve time for updating the driver.

7.2.3 Change of Simulator Integrator Scheme

The integrator used for the MPG in single support, can also be applied for simulation of the robot. The integrator is better at handling impacts, and joint limits than the currently used SimMechanics integrator. However, a change of integration scheme may also require that SimMechanics is aborted as the simulator structure, so the change may take long time to implement.

It should be considered if this is desired to spend time on, just to gain a better simulation of the impacts. However the advantages of having the integration scheme under full control allows changing the entire environment from Simulink to other engines that might run faster, and in the long run it could prove to be a good choice.

7.2.4 Error Handling Within The Controller

A controller can currently with the implemented system be aborted if a known error occurs. Other errors simply trigger a warning lamp and sound for the operator. This is not the desired behavior. Multiple cases can be presented where shutting of the controller is not preferable, as it causes the robot to collapse, if it is not hoisted in a crane.

A better error handling routine should be implemented. This requires knowledge from the hardware drivers that in general is not currently passed to the controller.

An example is the “out of sync” error on the FTS amplifiers. This error is not crucial if it happens when standing in double stance, as the system can then often be forced to stop, and to return to zero position. However during walking, the FTS might be used for balancing, or for registering when ground contact has been reached, or similar.

If error handling is placed in a controller an error from them should engage some other means of describing a switching surface, and then force the robot to stop walking, and attempt a stand still. Even freezing the robot in it’s current angular references are better than the current solution which only alerts the operator that something is wrong.

In general there are three types of known errors that should be handled in a controller.

1. EPOS errors, a defective joint must be either restarted, or shut down. The remaining part of the robot should respond appropriately. It is often better to create a zero order hold on the angular references, than shutting the system down completely.

2. FTS errors are troublesome and should at any rate do something. Depending on the final controllers structure and dependence on the FTS the handling should be different. For the structure presented in this thesis where the FTS are merely used as switches, a new offset

and transformation matrix, could possibly reduce the symptoms of the error enough to be operational.

3. Dependency on Sensors which are shut off. Currently the controller has no direct way of knowing if any of the sensors are shut off. However the information is available from the drivers, so they could be rewritten to take commands from a controller, so that they can be started and configured from within the initialization of the controller. A consequence could be that a controller will remain in the safe waiting state until the robots sensory system is actually ready for operation.

7.2.5 Non-linear MIMO Posture Controller

The current posture controller implementation using PID controllers does not handle dynamic effects such as cross couplings, and have difficulties with handling the impulses created at impact. This was expected at the time of posture controller choice, and limits the gait that can be realized in practice to be quite slow, when compared to humans.

A faster and human-like gait will result in greater impulses, and can render the PID controllers unstable. a rate limited controller is better at stabilizing after the impulse, but the optimal choice would be some controller that compensates for other joints.

The impulses are event driven, and will create effects that cannot be handled easily with a linear controller. An non-linear controller will have to be very aggressive at the time just after impact to suppress some of the errors, especially the ones relating to the hyper flexion of the knees.

A completely different approach could be to scale the hardware better, and replace some of the DC motors. Again the knees have proven to be very close to the limit of what they can actually deliver, and they require a high current just to break free of the stiction. This makes linearization of the joint a challenge and requires some strategy for anti stiction. Minimizing the stiction further, will make the joint behave closer to a linear system, reducing the need for a non-linear friction compensation.

7.2.6 Kevlar Belts and Spur Gears

Decreasing the height of the teeth in the spur gears will lower the stiction, and possibly the Coulomb friction on that side of the HDG. For some joints such as the joints located at the hip, the friction is nearly linear, and the existing solution is not causing problems, but for the knees, and the ankle pitch in particular, the stiction are quite high, and the effects are not very well described with a classic friction model.

A controller can be tuned to suppress the error, but the result is that the controller have difficulties in handling larger steps in the reference without becoming unstable, a non-linear control model would have to recreate, and negate the stiction behavior, which could improve performance of a controller. To improve development speed it is however suggested to fix this problem directly in the hardware. The price on a new gear setup is low in comparison to other robot expenses.

A company was found that could deliver Kevlar belts and spur gears. The company is called Brd. Klee.

7.2.7 Slave Follower setup on EPOS

The EPOS can be configured to run with an intern rate limited PID controller. The controller is already used for zero positioning, and could increase performance on the system as a whole.

For the joints equipped with two motors, it is suggested to investigate the master/slave system that the EPOS can be configured to use. By reading the datasheets it was found that they can be configured to operate as slaves, copying the output that the master EPOS outputs. Another and perhaps more appropriate choice could be to reapply the analog EPOS that is available in the lab, and use them for slaves. This not only reduces CAN traffic, but also ensures that the EPOS with common joints can never be configured in diverging states.

If a non-linear control scheme is intended build in PID controllers does not present a problem, as the output of a non-linear controller can formed as a gain setting, and be sent to the EPOS. The technique might seem familiar as a variant of Gain scheduling, and for the controller structure proposed in this thesis, the placement of the posture controller in the EPOS would make it easy to change PID gains when states are changed.

7.2.8 Stability Analysis of Dynamic Gait

when a gait is achieved, the orbit which is formed in the state phase plot, can be analyzed sing known tools as moving Pointcaré map analysis, and thus used to determine how well a controller is at walking.

The benefit of the analysis is that it is revealed how fast a gait can be changed to change speed, with the implemented structure, and when it should be done, as overlapping of the orbits indicate that a shift can happen. Also the bifurcation properties of the gait should appear, and it can be determined if the gait can be optimized towards another ground impulse.

7.3 Proposals for AAU-BOT2

Some of the experience that was gained from working with implementing controllers on AAU-BOT1, relates to the design of the robot, which could be improved in some areas. The most discussed changes to the design of a future robot is included in this section. There has been no projects presented regarding possibilities of a new robot, but this group recommends that such a project is launched in the near future, as the original goal has not yet been met, and that much can be gained from having more than one robot.

In general the recommendations revolves a breakdown of the development phases into smaller achievable goals, allowing future students to iterate on control structures, where the individual parts of the controller, can be tested independently. To do so requires that the hardware can be subjected to some partial goals, which cannot be made on AAU-BOT1, such as static gaits, or plastic impacts.

7.3.1 1:1 Scales.

AAU-BOT1 has a 1:1 scaling. This means that it has the size of fully grown human.

The advantages of this size is that the mechanics can be kept so large that they are easy to work with, and that there is sufficient room, and sufficient weight ratios for a carrying the computational power required on board. This allows for a design of a controller that can use modern techniques, which can improve performance.

Even larger robots will off course make the dynamic effects of carrying a computer even less, and approach human weight distribution even better. However, if the robot weighs much more than a human it will be hard to operate it manually, which is necessary when controllers are developed. In fact the current design with regards to outer dimensions and weights is comfortable to work with.

7.3.2 Knee Caps

AAU-BOT1 can hyper extend the knee almost as much as it can flex them. This is not human-like behavior, and actually requires that the controller is quite aggressive to duplicate the joint behavior.

The knee impact, if such was available, is a dynamic feature that aids in sustaining walking, and a knee lock mechanism, could increase development speed, as the impacts could be assumed plastic, and a passive walker model could be applied. Such a gait could then be extended afterward to take into account the impact effects, by pre-flexing the knee, and then absorbing some of the energy.

7.3.3 Series Spring, Damper and Linear Actuation

As the AAU-BOT is intended to operate as humans the actuation system should be closer to that found in the human body. If actuators was chosen that has a spring/damper in series with a linear actuator, the impacts are reduced and absorbed in the actuators directly, elongating the impact, and thus aiding in reducing damages to the robot.

Another benefit of a configuration more like human muscles, is that the transferability from the models presented with passive walkers to the active walkers is much better. This has been shown by the Flame robot, and similar hybrids.

The benefits are also relating to the original goal of producing robots at AAU, as a more human like configuration, will make it possible for health science researchers to better relate their findings to the field of robotics and vice versa.

7.3.4 Protection of Fragile Mechanics.

AAU-BOT1 is for the majority of the joints, well designed, however when implementing a controller, some unfortunate features are revealed. Especially concerning the placement of the gears on the outside of the skeleton.

Consider the ankle pitch gears. The gears on the hips, and the knees are all placed on the outside of the robot, this keeps them somewhat isolated from the remaining mechanical parts, and they can only be damaged if the robot falls. The gears for the ankle pitch joint is placed on the inside of the leg, placing them in the position that can easiest be hit by the other leg. Should the legs ever hit each other (which is kinematically poassible) those gears are bound to take the impact, and be damaged. The probabillity of this happening is quite high, as dynamic gaits in humans usually have a step width that is very small, so a human-like gait should also display such a feature.

Further more, any experiment on a system, when tuning and testing controllers can result in some unexpected reaction, i.e. loss of communication with a motor controller. It can be seen on the robot harware, that previous groups have experienced problems under the initial tests, and that at least one of the experiments has actually damaged the gears, already.

Had the gears been protected by a fender, or had they been placed inside the robot frame, the robot should be more durable against errors in softwares.

7.3.5 Integration of Cables in the Design Phase

AAU-BOT1 suffers from poor design with respect to cables. had it been considered from the beginning how the wiring could be done, less of them would be in danger of being damaged when the robot moves. The current configuration has shown that cables that in any way can be damaged between to parts of a joint, will eventually be damaged. Even under normal operations.

For the next design it is suggested to include an electronics engineering student in the design team, to provide initial knowledge on networking, amplifiers, cable sizes and sensor noise. Especially networking knowledge, such as which networks are prone to noise, or latencies, is knowledge that could be used in the design phase.

7.3.6 Improved Configuration Space

While the robot has sufficient DoF for researching walking, and have a configuration space that should allow human like walking, it would be a great advantage to be able to walk statically first, as that particular problem can be reduced to optimizing a posture controller, and then track a stable trajectory. The keyword being stable. As the speed of the stable trajectory can be kept arbitrary low. An arbitrary low speed would allow for creating a posture controller which could handle impacts and cross couplings and test it, and iterate on it without having to rely on an untested trajectory generation and balance controller for avoid destroying the robot in an accidental fall.

7.3.7 Installing Accumulators

The accumulators on AAU-BOT1 is not yet installed. This lowers the CoM more than it was designed for, and thus the part of the CoM which is defined by the legs motions is relatively high. While this aids in stabilizing the robot, it also makes it difficult to move the CoM very far, and thus makes it difficult to achieve walking.

The addition of the power cables however adds unnecessary dynamics to the robot, which are especially annoying when the robot is standing on one foot, as very small forces can cause the robot to fall in this state, if the balance controller is not active.

In the development phase of a controller, and a control model, it would improve development time if the robot could be run on batteries for a while, until the balance controller was able to counter the effect from the cables.

7.3.8 Integration of Sensors in the Design.

The FTS are integrated well on the robot as a part of the frame. However potentiometers used for measuring angles, or the later addition of the IMU is not very well integrated, and actually reduces the configuration space further.

7.3.9 Build in Visual Outputs

AAU-BOT1 has the disadvantage in the current implementation, that the controller cannot communicate directly to the operators. An example of a solution could be to mount lights on the robot, so that it could report information about which state the controller was in and similar information.

This group has installed lights on the crane frame, which is connected to the robot via a cable, this choice has been made as there is little room on the robot for lights which can be emit information omnidirectional, which is important for operators.

Experience has shown that three colored lights are sufficient, but up til five different lights are preferred, if a similar solution should be chosen. Larger lights, like strobe lights has also shown to be better than e.g. a screen, as they are easier to see when something goes wrong in the system.

Bibliography

[1] Anatomical terms of location, 2010.
http://en.wikipedia.org/wiki/Anatomical_terms_of_location, 29/05-2010.

[2] Anatomical terms of motion, 2010.
http://en.wikipedia.org/wiki/Anatomical_terms_of_motion, 29/05-2010.

[3] Anthropomorphism, 2010.
http://en.wikipedia.org/wiki/Anthropomorphism, 29/05-2010.

[4] Gait, 2010.
http://en.wikipedia.org/wiki/Gait, 29/05-2010.

[5] Ahmad Reza Arshi Amir Abbas Zadpoor, Ali Asadi Nikooyan. A model-based parametric study of impact force during running. *Journal of Biomechanics*, 2006. *CD: /Bibliography/Litterature/Zadpoor,Nikooyan,Arshi;A_model-based_parametric_study_of_impact_force_during_running.pdf*.

[6] Steven H Collins Andy Ruina, Martijn Wisse. A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research*, 20(7):607–615, 2001. *CD: /Bibliography/Litterature/Collins,Wisse,Ruina;A_Three-Dimensional_Passive-Dynamic_Walking_Robot_with_Two_Legs_and_Knees.pdf*.

[7] Steven Hartley Collins Andy Ruina. A bipedal walking robot with efficient and human-like gait. *Proc. IEEE International Conference on Robotics & Automation*, pages 1983–1988, 2005. *CD: /Bibliography/Litterature/Collins,Ruina;A_Bipedal_Walking_Robot_with_Efficient_and_Human-like_Gait.pdf*.

[8] Yariv Bachar. Developing controllers for biped humanoid locomotion. Master's thesis, University of Edinburgh, 2004. *CD: /Bibliography/Litterature/Bachar;MscThesis:Developing_Controllers_for_Biped_Humanoid_Locomotion.pdf*.

[9] Remy Bohmer. Howto: Build an rt-application, 2009.
http://rt.wiki.kernel.org/index.php/HOWTO:_Build_an_RT-application, 04/03-2010.

[10] B. L. Davis C. L. Vaughan and J. C. O'Connor. *Dynamics of Human Gait*. Kiboho Publishers, 2. edition, 1992. ISBN: 0-620-23558-6.

[11] William A. Werbel Adam I. Daoud1 Susan D'Andrea Irene S. Davis Robert Ojiambo Mang'Eni Daniel E. Lieberman, Madhusudhan Venkadesan and Yannis Pitsiladis. Foot strike patterns and collision forces in habitually barefoot versus shod runners. *Nature*, 463, 2010. *CD: /Bibliography/Litterature/Lieberman,Venkadesan,Werbel,Daoud,Andrea,Davis,MangEniPirsiladis;Foot_strike_patterns_and_collision_forces_in_habitually.pdf*.

[12] Gerhard Wanner Ernst Hairer, Christian Lubich. Geometric numerical integration illustrated by the störmer-verlet method. *IEEE Transactions on Robotics and Automation*, 2003. *CD: /Bibliography/Litterature/Hairic,Lubich,Wanner;Geometric_numerical_integration_illustrated_by_the_Störmer-Verlet_method.pdf*.

[13] George T. Fallis. Walking toy, 1888. *CD: /Bibliography/Litterature/Fallis;Walking_Toy_Patent_January_17_1888.pdf*.

[14] Kenichi Ohara Yasushi Mae Gaku Takeo, Tomohito Takubo and Tatsuo Arai. Rotational operation of polygonal prism by multi-legged robot. *IEEE International Conference on Mechatronics and Automation*, 2009. *CD:* .

[15] Henry Gray. *Anatomy of the Human Body*. Lea & Febiger, 2. edition, 1918. ISBN: N/A.

[16] Joseph Hamill and Kathleen M. Knutzen. *Biomechanical Basis of Human Movement*. Lippincott Williams & Wilkins, 2. edition, 2003. ISBN: 0-7817-3405-3.

[17] William B. Heard. *Rigid Body Mechanics: Mathematics, Physics and Applications*. Wiley, 1. edition, 2005. ISBN: 978-3-527-40620-3.

[18] Daan Hobbelen. Flame, 2010. `http://www.3me.tudelft.nl/live/pagina.jsp?id=01f7247a-ae48-4328-a17f-631c0e8c38af&lang=en`, 30/05-2010.

[19] Yildirim Hurmuzlu and Osita D.I. Nwokah. *The Mechanical Systems Design Handbook: Modeling, Measurement, and Control (Electrical Engineering Handbook)*. CRC Press, 1. edition, 2001. ISBN: 978-0849385964.

[20] Arthur D. Kuo. A simple model of bipedal walking predicts the preferred speed-step length relationship. *ASME*, 123, 2001. *CD: /Bibliography/Litterature/Kuo;A_Simple_Model_of_Bipedal_Walking_Predicts_the_Preferred_Speed-Step_Length_Relationship.pdf*.

[21] James Ronald Leigh. *Control Theory - A Guided Tour*. IEE, 2. edition, 2005. ISBN: 0-86341-332-3.

[22] Ambarish Goswami Marko B. Popovic and Hugh Herr. Grond reference points in legged locomotion: Definitions, biological trajectories and control implications. *The International Journal of Robotics Research*, 24(12):1013–1032, 2005. *CD: /Bibliography/Litterature/Goswami,Popovic,Herr;Ground_Reference_Points_in_Legged_Locomotion:_Definitions,_Biological_Trajectories_and_Control_Implications.pdf*.

[23] Maxon. Application note; canopen basic information, 2004. *CD: /Bibliography/Datasheets/Maxon_EPOS_CAN_Note.pdf*.

[24] Tad McGeer. Passive dynamic walking. *The International Journal of Robotics Research*, 9(2):62–73, 1990. *CD: /Bibliography/Litterature/McGeer;Passive_Dynamic_Walking.pdf*.

[25] Tad McGeer. Passive walking with knees. *IEEE*, pages 1640–1645, 1990. *CD: /Bibliography/Litterature/McGeer;Passive_Walking_With_Knees.pdf*.

[26] Uwe Mettin. Principles for planning and analyzing motions of underactuated mechanical systems and redundant manipulators. Master's thesis, Umeå University, 2009. *CD:* .

[27] Allan Agerbo Nielsen Mikkel Melters Pedersen and Lars Fuglsang Christiansen. Design of biped robot aau-bot1. Master's thesis, Aalborg University, 2007. *CD: /Bibliography/Litterature/Pedersen,Nielsen,Christiansen;MscThesis: Design_of_Biped_Robot_AAU-BOT1.pdf*.

[28] Elliot Nicholls. Bipedal dynamic walking in robotics. Master's thesis, The University of Western Australia, 1998. *CD: /Dokumenter/aau/P9/AAU-BOT1/Bibliography/Litterature/Nicholls;Honours_Thesis:Bipedal_Dynamic_Walking_in_Robotics.pdf*.

[29] Jens Bo Nielsen. How we walk: Central control of muscle activity during human walking. *The Neuroscientist*, 2003. *CD: /Bibliography/Litterature/Nielsen;How_We_Walk:_Central_Control_of_Mscle_Activity_during_Human_Walking.pdf*.

[30] Michael Odgård Kuch Niss and Brian Thorarins Jensen. Modelling, simulation, and control of biped robot aau-bot1. Master's thesis, Aalborg University, 2009. *CD: /Bibliography/Litterature/Jensen,Niss;MscThesis:Modelling,_Simulation,_and_Control_of_Biped_Robot_AAU-BOT1.pdf*, owner = Henrik, timestamp = 2009.09.21.

[31] Mathias Garbus Per Kingo Jensen and Jan Vestergaard Knudsen. Instrumentation, modelling and control of aau-bot1. Master's thesis, Aalborg University, 2008. *CD: /Bibliography/Litterature/Pedersen,Nielsen,Christiansen;MscThesis:Design_of_Biped_Robot_AAU-BOT1.pdf*.

[32] John W. Peterson. Arc length parametization of spline curves. *Taligent Inc.* *CD: /Bibliography/Litterature/Peterson;Arc_Length_Parameterization_of_Spline_Curves.pdf*.

[33] John W. Jewett Jr. Raymond A. Serway. *Physics for Scientists and Engineers with Modern Physics*. Thomson brooks/cole, 6. edition, 2004. ISBN: 0-534-40949-0.

[34] A. Taneja B. Seth S. Mukherjee, V. Sangwan. Stability of an underactuated bipedal gait. *Science Direct - Bio systems 90*, 90, 2007. *CD: /Bibliography/Litterature/Mukherjee,Sangwan,Taneja;Stabillity_of_an_underactuated_bipedal_gait.pdf*.

[35] Fumio Kanehiro Kenji Kaneko Mitsuharu Morisawa Hirohisa Hirukawa Shin'ichiro Nakaoka, Atsushi Nakazawa and Katsushi Ikeuchi. Learning from observation paradigm: Leg task models for enabling a biped humanoid robot to imitate human dances. *The International Journal of Robotics Research*, 2007. *CD:* .

[36] Harry Skinner. *CURRENT Diagnosis & Treatment in Orthopedics*. McGraw-Hill Medical, 4. edition, 2006. 978-0071438339.

[37] Mark W. Spong and Seth Hutchinson. *Robot Modeling and Control*. Wiley, 1. edition, 2005. ISBN: 978-0-471-64990-8.

[38] Timothy D. Tuttle and Warren P. Seering. A nonlinear model of a harmonic drive gear transmission. *IEEE Transactions on Robotics and Automation*, 1996. *CD: /Bibliography/Litterature/Tuttle,Seering;A_Nonlinear_Model_of_a_Harmonic_Drive_Gear_Transmission.pdf*.

[39] Dirk Wollherr. Design and control aspects of humanoid walking robots. Master's thesis, Technische Universität München, 2005. *CD: /Bibliography/Litterature/Wollherr;Design_and_Control_Aspects_of_Humanoid_Walking_Robots.pdf*.

[40] Ting-Ying Wu and T.-J. Yeh. Optimal design and implementation of an energy-efficient biped walking in semi-active manner. *Robotica*, 27:841–852, 2009. *CD: /Bibliography/Litterature/Wu,Yeh;Optimal_design_and_implementation_of_an_energy-efficient_biped.pdf*.

[41] G. Cappellini F. Lacquaniti Y. P. Ivanenko, R. E. Poppele. Motor patterns in human walking and running. *Journal of Neurophysiology*, pages 3426–3437, 2006. *CD: /Bibliography/Litterature/Cappellini,Ivanenko,Poppele,Lacquaniti;Motor_Patterns_in_Human_Walking_and_Running.pdf*.

[42] F.Lacquaniti Y.P.Ivanenko, R.E. Poppele. Five basic muscle activation patterns account for muscle activity during human locomotion. *Journal of Physiology*, pages 267–282, 2004. *CD: /Bibliography/Litterature/Ivanenko,Poppele,Lacquaniti;Five_basic_muscle_activation_patterns_account_for_muscle_activity_during_human_locomotion.pdf*.

A

SimMechanics

In SimMechanics a model is set up using the following implementation rules.

- ▷ A ground reference coordinate system must be connected with a machine by a joint. The joint can be a 6DoF joint, and unconstrained, if the machine is flying or falling freely, but can also be a joint with constraints modelling an object connected with the ground.
- ▷ A machine is one or more rigid bodies connected to each other by mechanical joints.
- ▷ Mechanical joints is simply a set of constraints, which are connected through a mechanical link to at least two bodies.
  - On each body the mechanical link specifies a local coordinate system that is located somewhere geometrically on the body. The coordinate systems can be located relative to a body's Center of Gravity (CoG), to other body coordinate systems or to the worlds coordinate system. The coordinate systems can be rotated i.e. to align it's axis's with the geometric axis of a hinge in the physical world.
  - A library of joints exists which automatically sets up constraints i.e. a ball joint can be specified like "no linear motion is allowed", however in SimMechanics the focus is put on the DoF which allows motion.
  - Joints connects a body to a follower, while forces are transferred both ways, and the order is less important during simulation, it has some effects when initial conditions are applied. They relate the followers coordinate system to the body's.
  - Joint constraints does not handle friction, actuators or other applied forces, it only sets up geometric constraints
- ▷ It is assumed that the gravitational field is homogenic, and that bodies are solid, and thus CoM and CoG are always located at the same place.
- ▷ Sensors and actuators can be attached through a mechanical link to a joint or a body.
- ▷ Sensors allows Simulink to extract, both linear and angular position, velocity and acceleration measurements.
- ▷ Actuators allows Simulink to apply forces and torques at a joint, or at a coordinate system on a body.
- ▷ Coordinate systems are by default oriented with the gravity oriented in the y axis.

The blocks used in this project from SimMechanics is described below and can be seen in Figure A.1.

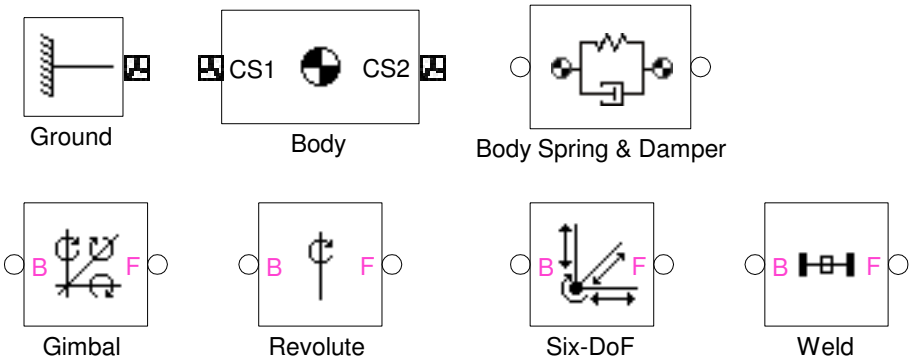


Figure A.1: SimMechanics blocks

Bodies

In SimMechanics a Rigid body is named a "body". For each body a mass, and an inertia matrix can be specified (or loaded from MATLAB workspace). For each body a coordinate system is defined named "CG" this is a short hand notation for Center of Gravity. However, in SimMechanics the rigid bodies are also convex, and solid, and gravity is always assumed to be a homogenic field, and thus the coordinate system should in fact have been named CoM, as that is where it is located. The CG coordinate system is actually a vector, locating the axis aligned CoM in world coordinates.

The ports on the body block defined by the coordinate systems set up in the block, can be used to apply forces and torques to the body.

Mechanical links

In relation to any defined coordinate system, a new coordinate systems can be specified. For each coordinate system specified in a body, a mechanic link can be attached to the body block. A mechanical link specifies the coordinates used transfer forces from one object to another. As an example to bodies can be connected to each other with a "Body Spring and Damper" block, this creates a linear force based on the euclidean distance of two coordinate systems, and a spring and damper model.

Joints

Two bodies can be connected to each other through a joint. The joint is a Simulink block with two ports for mechanical links, a (b) body and a (f) follower. The order which they are connected is not important though in the newer versions of SimMechanics. When one of the ports are connected to a body, it creates a constraint about where in relation to the body the joint must be located. If there is different bodies attached at each port, there is by default no constraints, two bodies coordinate systems are simply set up to have some transformation between them.

There exists multiple joint blocks of interest, each joint block sets up different extra constraints.

- ▷ A Ball Joint or a Gimbal Joint, is an example where a constraint which is easy to understand exists. It specifies that the euclidean distance between the two coordinate systems on the bodies must remain constant. Hence the bodies may rotate freely around the joint, but may not change their distance to the joint. The joint is automatically moved if the bodies are moved.

- ▷ A Revolute Joint, have the same constraints as a ball joint, but also has the constraints that rotations are only allowed about a single axis. The axis can be defined by a vector, if it is not axis aligned at initial position.

- ▷ A Six-DoF Joint, has no constraints, which allows both rotations and translatory movements.

- ▷ A Weld Joint, has the constraint that nothing must be changed in the relation between the bodies. The advantage of a weld instead of joining the two models in a single rigid body, is that the force necessary to respect the constraint can be calculated.

Ground

The Ground block is a special type of body, which defines the world coordinate frames. Hence all other bodies much in some way be connected with ground through a chain of joints and bodies. using a Six-DoF Joint will separate them dynamically, and only position the body in relation to ground. When this is used ground forces must be applied in some other manner.

Initial Conditions

Can be applied after modelling to a joint or a body, using a special IC block.

A.1 Example Implementation of a Falling Box Landing on a Spring and Damper.

A model is set up to illustrate the basic principles used in SimMechanics and in this thesis. The model is a dice falling vertically downwards on a spring/damper. It presents a prelude to the ground model used for the robot in the thesis, and thus the spring/damper is named ground in this example. An overview of the model is seen in Figure A.2.

The necessary SimMechanics ground reference block is connected through a 6DoF custom created joint to the dice. There is no constraints on the allowed motions in the joint, hence it is not restricting the free fall in any way. Initial conditions are provided to the joint, to lift the dice from the ground.

The dice is a cube, and has it's mass configured as a mass in CoM and the Inertia is set up as a Principal axis inertia matrix.

Eight local coordinate systems are defined, distributed evenly around the CoM to form the vertices of a cube. At each of the local coordinate systems a ground spring and damper system is set up.

A SimMechanics model can be directly visualized in the Virtual Reality Toolbox in Simulink, however for this project, this build in visualizer is not used. This is partly because it takes up alot of system resources on the simulation computer, and partly because it has an unintuitive camera interface. Hence this model also illustrates how to use The Palantir visualizer created for the openDE simulation, from Simulink.

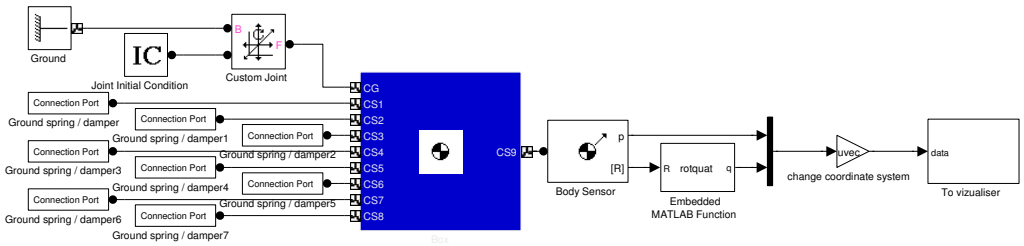


Figure A.2: A rigid body with eight vertices in the shape of a cubic box that falls on a spring and damper model of the ground, a sensor is attached to the body at a coordinate system which is located at CoG, this outputs the CoG's position and rotation matrix. This is transferred to Palantir as a position and quaternion, to allow visualization in the environment used to visualize AAU-BOT1.

If the model is tested without a running Palantir, delete the "To Visualizer" block, and enable the virtual reality system in Simulink.

A block is build that transfers the coordinate system mechanical link data, to Simulink data, and back, in this way a spring damper model can be created in a familiar fashion in Simulink, and applied to the vertices that collides with the ground see Figure A.3.

Notice that the rigid body model at all times has the vertices connected with the Simulink spring/damper force generator. This is because SimMechanics does not support switching of the mechanical links. However, when applying exactly zero force, this connection has no effect on the model. To ensure that forces are only applied when contact with the spring/damper exists geometrically, the implementation of some collision detection is needed.

The implementation of the collision detection for this model is quite simple, as the ground model is completely flat. Hence it can be done by very simply by a zero crossing detection for the vertices with 0. The resulting Spring/Damper model with collision detection can be seen in Figure A.4

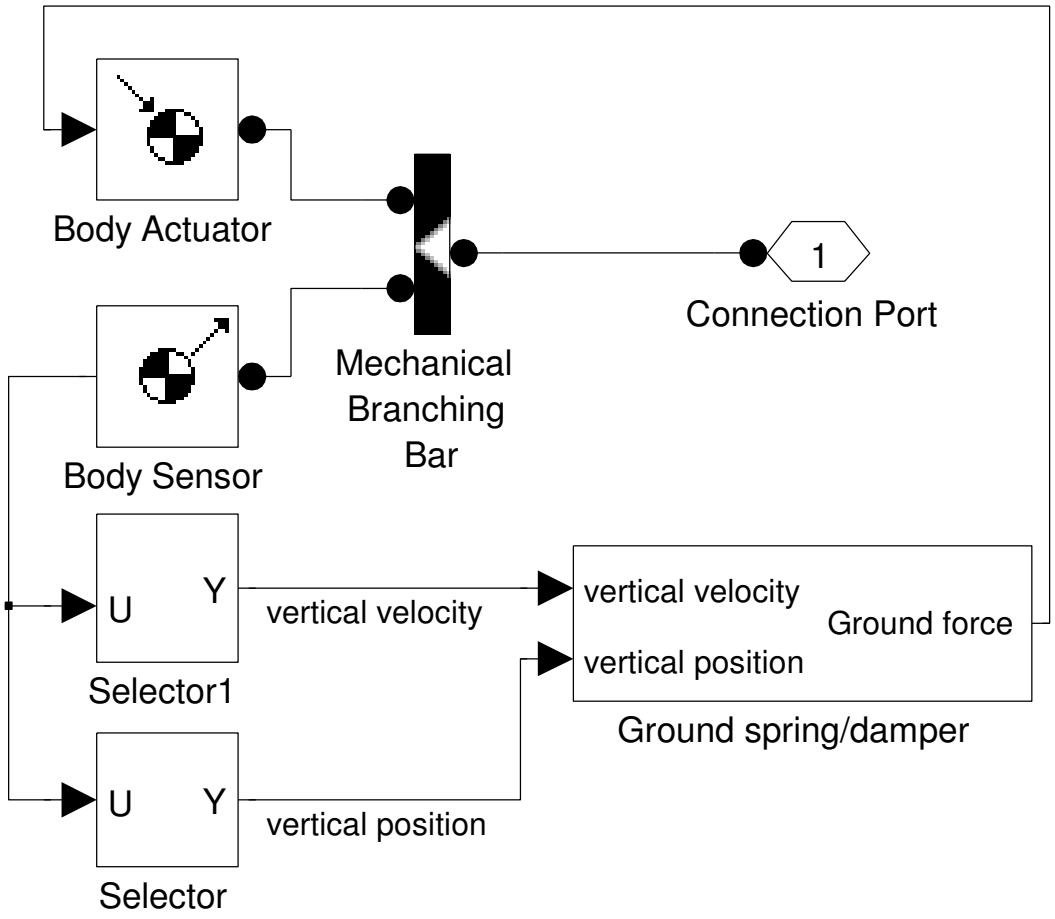


Figure A.3: The "Ground spring / damper" block in the overview just contains conversion to Simulink data.

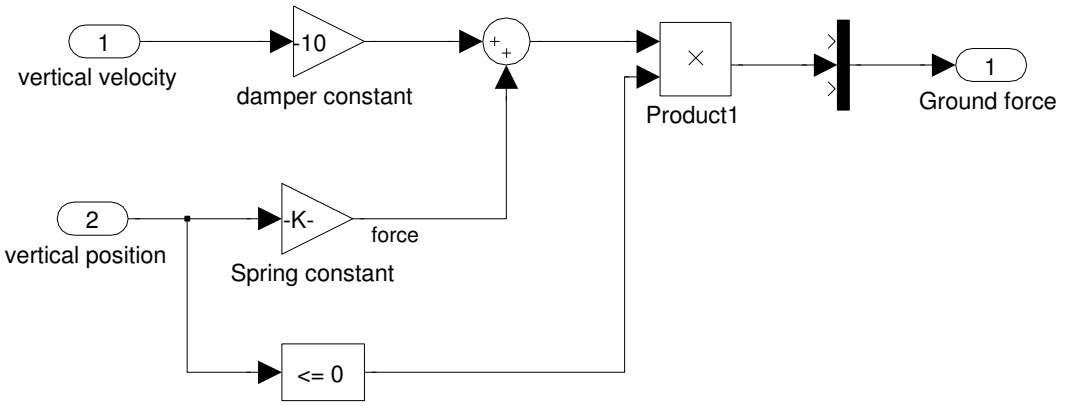


Figure A.4: The "Ground spring / damper" with collision detection, is implemented as a classical PD controller, and a zero crossing Boolean is multiplied to the result to ensure that zero force is applied when the cube is not touching the ground..

B

Derivation of Passive Walker Dynamics

Today's modern models of Passive Dynamics Based Walking Robots, are evolved from the 2d ramp walkers derived in 1990 by McGeer. Hence the derivation in this appendix is also described in 2d first, and afterward expanded to 3d. The purpose of this appendix is to produce a 3d dynamics model of a walking robot with dimensions as AAU-BOT1. The model is used to identify how such a walker can be actuated to stay in a stable gait, if it is walking at a ramp which does not drop, but instead climbs, or is level.

B.1 Definitions in this Appendix

The name convention used for the most common definitions in passive walking is adopted directly from McGeer, a few variables are selected otherwise to harmonize with modern literature. To avoid confusing the parameters are defined below:

γ	Is the ramp or slope angle, when positive, the walker goes "down hill".
l	Is the length of the leg, it is assumed that the two legs are identical in shape.
c	Is the distance from the point foot, to the center of mass of the leg measured along the leg.
w	Is the distance to the legs center of mass, measured perpendicular from the leg in walking direction.
R	Is the radius of the semicircular foot.
m	Is the leg mass, (located at the legs center of mass).
M	Is the hip mass.
g	Is the gravitational acceleration.
I	Is the moment of inertia.
r_{gyr}	Is the radius of gyration, in this case defined as: $(r_{gyr} = \sqrt{\frac{I}{mass}})$.
θ	Is the angle from the ramp normal, to the stance leg, in the direction of the ramp.
β	Is the angle between the legs.
L	Is the angular momentum.
ω	Is the angular velocity.
λ	Is viscous dampening

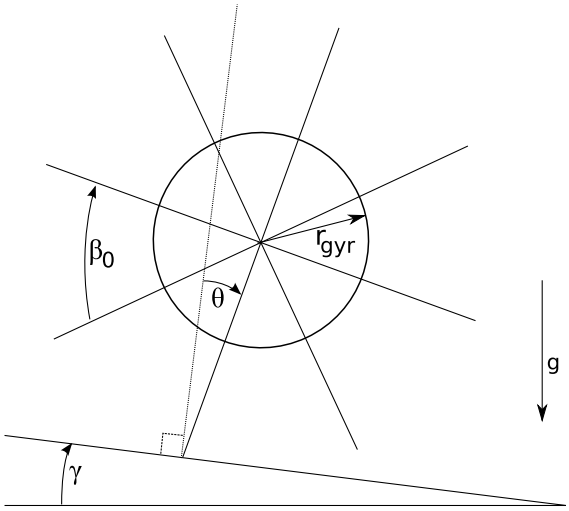


Figure B.1: A rimless wheel on a slope, with illustration of the angles used in the model and length of r_{gyr}

B.2 Rimless Wheel Model

The simplest walker model is a rimless wheel. It does not walk in the way that humans walk, however it does not rotate nicely as a wheel either. A rimless wheel can be seen on figure B.1.

B.2.1 Swing Phase of a Rimless Wheel Walking Model

Supposing that the wheel is already in motion, it will rotate around the stance spoke, in the same manner as an inverted pendulum. Hence the motion, relating to the swing phase can be modelled as an inverted pendulum attached on a slope, see Equation B.1.

$$\ddot{\theta} = (\frac{gl}{l^2 + r_{gyr}^2})(sin(\theta) + sin(\gamma)) \tag{B.1}$$

Hence it can be concluded that the rimless wheel accelerates it's mass downwards in a curve, as the stance spoke "falls". It can also be concluded that the potential energy delivered by gravity is converted to kinetic energy. However after some time, the wheel will have "fallen" so much that another spoke collides with the ramp. This is somewhat relating to the situation that happens as heel strike, and is investigated in the next section.

It happens at the exact moment in time when two spokes form a triangle with the ramp.

B.2.2 Heel strike of a Rimless Wheel Walking Model

If the collision at heel strike is treated as inelastic and impulsive, the angular momentum must be conserved about the impact point [33]. This seems quite appropriate for this purpose, hence the model of heel strike presented by McGeer is investigated. It uses angular momentum to calculate the loss in angular speed at heel strike. see Equation B.2.

Notice that the L^- denotes that it models the angular momentum just before impact, and that the L^+denotes that it models it just after impact.

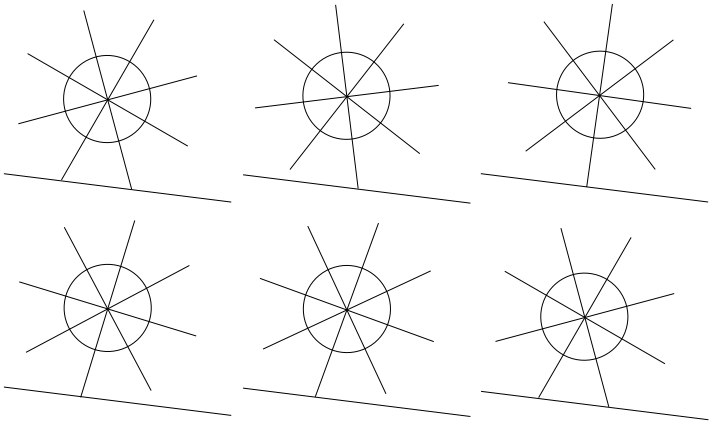


Figure B.2: The cartoon illustrates the motions of the rimless wheel. Notice that it most of the time stands on one spoke, and while that happens it resembles an inverted pendulum.

$$L^- = (cos(\beta_0) + r_{gyr}^2) M l^2 \omega^- \tag{B.2}$$

$$L^+ = (1 + r_{gyr}^2) M l^2 \omega^+$$

The conservation of the angular momentum yields that

$$L^- = L^+$$

Which leads to

$$\omega^+ = \omega^- (\frac{cos(\beta_0) + r_{gyr}^2}{1 + r_{gyr}^2}) \tag{B.3}$$

As $cos(\beta_0)$ tends to be < 1 (because there is usually $< \infty$ spokes on a rimless wheel) Equation B.3 defines that a loss of angular velocity ω occurs when a heel strike occurs.

B.2.3 Conclusions on the Rimless Wheel Walking Model

The model of a rimless wheel shows that it does exhibit motions comparable to a walk, see FigureB.2 for concept of walk.

To obtain a stable "walk" with a rimless wheel in 2d, the slope γ must be chosen large enough, to ensure that the build up of angular speed in the swing phase, is equal to the loss of angular speed in the heel strike phase.

Steeper slopes, will result in a faster gait. however the slope can be too steep and the rimless wheel model can reach speeds so high that the assumptions of impulsive collisions at heel strike does not hold in practice.

Shallower slopes will slow the walker down, and after a few "steps" it will stop walking, and reach a stand still.

To successfully initiate "walking", it can be seen from the inverse pendulum model of the swing phase that, it is necessary to enter the first swing phase with an angular speed that is at least high enough to allow the spoke to overcome the de-celleration that occurs before it reaches

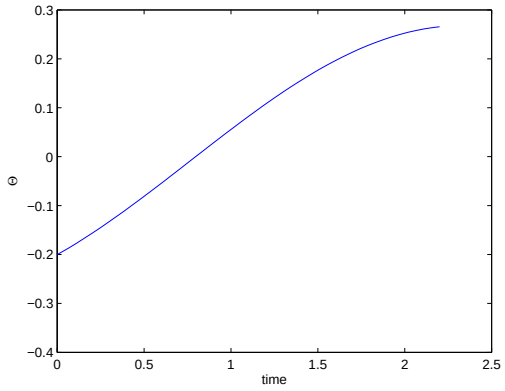


Figure B.3: The angle of the stance spoke while in swing phase.

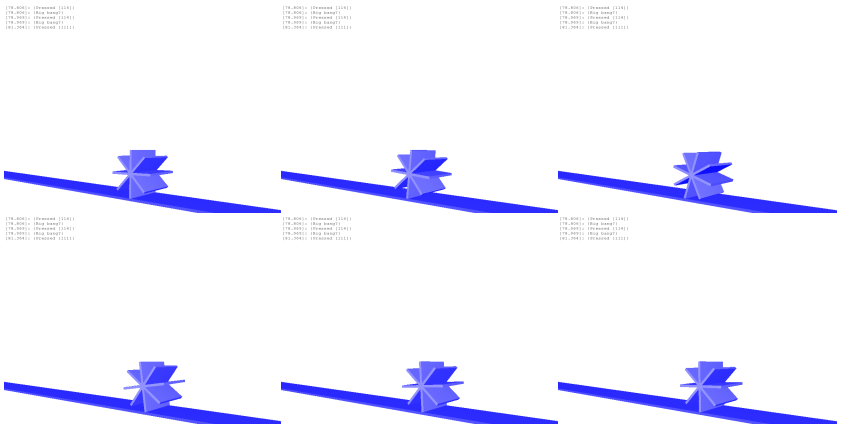


Figure B.4: Cartoon created from an early edition of the visualization tool. It illustrates motions of the simulated rimless wheel.

the vertical stance. Otherwise the walker will fall back down at the previous spoke and stand still, (at least for all reasonable slope angles).

A successful simulation done in the created openDE simulator, is visualized and shown as a cartoon in Figure B.4. When compared to the analytical case presented in the previous text, it is noteworthy that because of spoke width, there is multiple impacts, which requires a bit steeper slope to sustain speed in the openDE environment.

B.3 Straight Legged Biped Model

Supposing the rimless wheel could be cut out, so that only the two lower spokes remained. Then the first step would still succeed exactly as before, and the swing phase of the second step would initialize as supposed to. However, as no more spokes exists to carry the wheel, it will fall and land on the ground. This situation is illustrated in Figure B.5.

If the spoke which has ground clearance, is rotated while the wheel is in the swing phase, it can be rotated to the position of the next “missing” spoke. In relation to the walking motion,

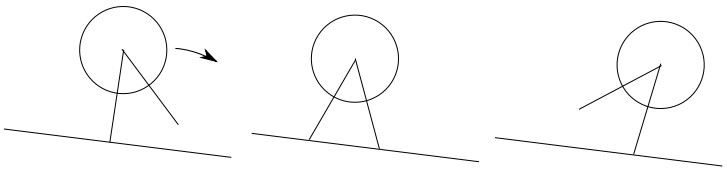


Figure B.5: A rimless wheel is reduced to having 2 spokes. It will be able to take the first step, but not the second, and it will fall.

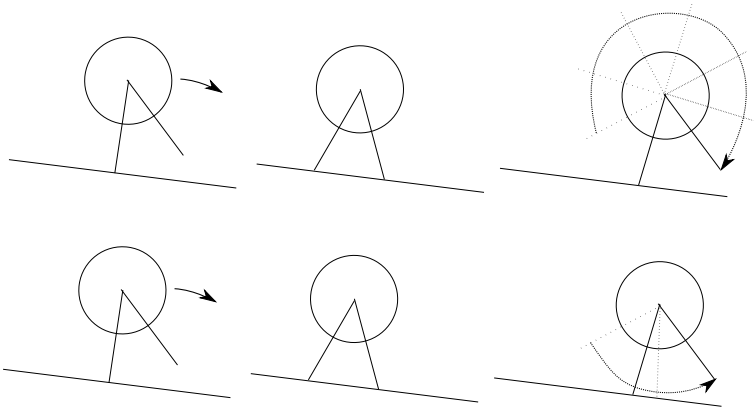


Figure B.6: If the spoke not being used for support is rotated to the position of the missing spoke, the wheel will keep walking. The rotation can be done in two ways shown here, the second solution is easy to identify as biped walk.

the resulting spoke configuration at heel strike becomes equivalent to the rimless wheels.

The rotation of the free spoke can be done in the two ways illustrated in Figure B.6. One option accelerates the spoke in the direction of the movement of the wheel. Hence it rotates faster than the wheel it self, and reaches the missing spokes position before it is used for support.

The other option swings the spoke forward as a pendulum. The second option resembles humanoid walk in concept and is hence modelled in the next section.

B.3.1 Swing Phase of a Straight Legged Biped Model

The second option for achieving walk with the two spoked rimless wheel, can be modelled as a double pendulum. One pendulum is attached to the ground, and then inverted, representing the stance leg, the other pendulum is attached to the top end of the stance leg (dubbed hip joint).

The two pendulums have similar lengths, but not necessarily similar masses. In fact it is generally so that the hip is much heavier than the feet on a walking robot.

A walking robot as the one modelled, will not be able to walk in the real world, as it will strike the ground during the swing phase. At least one solution to this problem has been observed B.7, which works reasonably well, and is also used in the openDE simulation. Pads are placed where the stance legs are expected to be, hence the robot gains clearance from the ground during the swing phase. This allows for a physical replication of the mathematical model, even though very few ramps are usually found prefabricated with suitable spaced pads.

As this model is not intended for realization, and a realization scheme already exists if such

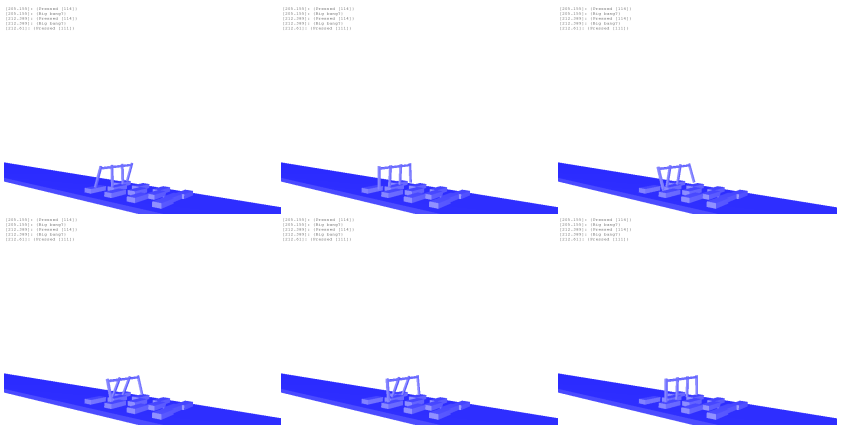


Figure B.7: The straight legged walker, walks down a ramp with pads attached on top, to provide ground clearance. The cartoon is extracted from an early edition of the visualization tool, and the simulation is run using openDE. The simulation model is stored in the standard Xode.

where needed, the mid swing ground contact is ignored in this model.

The geometry of a double pendulum is set up as shown on Figure B.8. Notice that $\hat{\theta}$ and $\hat{\phi}$ is used instead of θ and ϕ this is not intended to have a mathematical interpretation it is simply to avoid confusion with the previously defined θ. The relation between the two relates to the incline of the ramp, and can easily be injected as an angular offset.

$$x_1 = l \cdot sin(\hat{\theta}) \tag{B.4}$$

$$y_1 = l \cdot cos(\hat{\theta}) \tag{B.5}$$

$$x_2 = l \cdot (sin(\hat{\phi}) + sin(\hat{\theta})) \tag{B.6}$$

$$y_2 = l \cdot (cos(\hat{\phi}) + cos(\hat{\theta})) \tag{B.7}$$

The Lagrangian of a system, is the representation of a system of motion. It is an applicable method together with the Euler-Lagrangian method to determine the equations of motion when a system is conservative. The Langrian is defined as $L = K - P$. Where K is kinetic energy, and P is potential energy.

Potential Energy

The potential energy equations of an object is just the mass times the gravity times the altitude.

$$P = m \cdot g \cdot y \tag{B.8}$$

Hence if the pendulums are simple pendulums, the potential energies can be found as

$$P_1 = m_1 \cdot g \cdot y_1 \tag{B.9}$$

$$P_2 = m_2 \cdot g \cdot y_2 \tag{B.10}$$

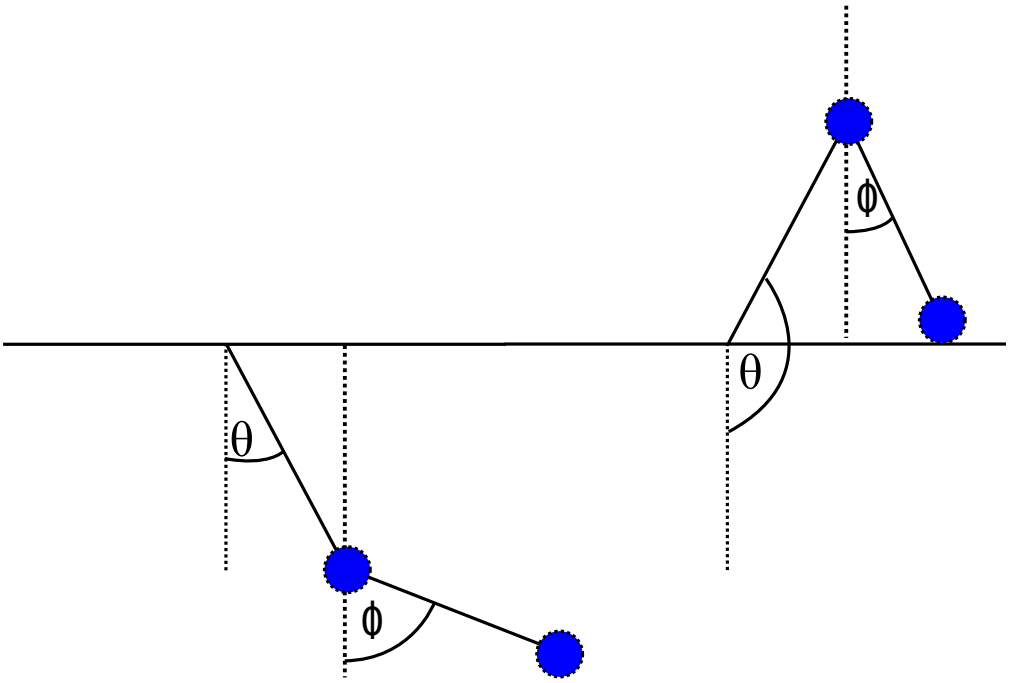


Figure B.8: Double Pendulum, reference angles, the same model also describes a biped, simply by inverting one of the pendulums.

Kinetic Energy

The kinetic energy equation is defined as

$$K = \frac{mv^2}{2} \tag{B.11}$$

The only unknown variable is v^2, it can be found as $v^2 = \dot{x}^2 + \dot{y}^2$, as it is the square of the sum of the speed in both axis's. Hence the speed in those directions are derived first.

$$\dot{x}_1 = l \cdot \dot{\hat{\theta}} cos(\hat{\theta}) \tag{B.12}$$

$$\dot{y}_1 = -l \cdot \dot{\hat{\theta}} sin(\hat{\theta}) \tag{B.13}$$

$$\dot{x}_2 = l \cdot \dot{\hat{\phi}} cos(\hat{\phi}) + \dot{x}_1 \tag{B.14}$$

$$\dot{y}_2 = -l \cdot \dot{\hat{\phi}} sin(\hat{\phi}) + \dot{y}_1 \tag{B.15}$$

Substituting into EquationB.11, and remembering $(cos^2(x) + sin^2(x) = 1)$ gives

$$K_1 = \frac{m_1(\dot{x}_1^2 + \dot{y}_1^2)}{2} = \frac{m_1(l^2 \cdot \dot{\hat{\theta}}^2[cos(\hat{\theta})^2 + sin(\hat{\theta})^2])}{2} = \frac{m_1}{2} l^2 \dot{\hat{\theta}}^2 \tag{B.16}$$

$$K_2 = \frac{m_2(\dot{x}_2^2 + \dot{y}_2^2)}{2} \tag{B.17}$$

$$K_2 = \frac{m_2}{2}\{[l\cdot\dot{\hat{\phi}}cos(\hat{\phi}) + l\cdot\dot{\hat{\theta}}cos(\hat{\theta})]^2 + [-l\cdot\dot{\hat{\phi}}sin(\hat{\phi}) - l\cdot\dot{\hat{\theta}}sin(\hat{\theta})]^2\}$$

$$K_2 = \frac{m_2}{2}\{[l\cdot\dot{\hat{\phi}}cos(\hat{\phi})]^2+[l\cdot\dot{\hat{\theta}}cos(\hat{\theta})]^2+2[l\dot{\hat{\phi}}cos(\hat{\phi})\cdot l\dot{\hat{\theta}}cos(\hat{\theta})]+[-l\cdot\dot{\hat{\phi}}sin(\hat{\phi})]^2+[-l\cdot\dot{\hat{\theta}}sin(\hat{\theta})]^2+2[l\dot{\hat{\phi}}sin(\hat{\phi})\cdot l\dot{\hat{\theta}}sin(\hat{\theta})]\}$$

$$K_2 = \frac{m_2}{2}\{[l^2\dot{\hat{\phi}}^2cos(\hat{\phi})^2]+[l^2\dot{\hat{\theta}}^2cos(\hat{\theta})^2]+[l^2\dot{\hat{\phi}}^2sin(\hat{\phi})^2]+[l^2\dot{\hat{\theta}}^2sin(\hat{\theta})^2]+2[l\dot{\hat{\phi}}cos(\hat{\phi})\cdot l\dot{\hat{\theta}}cos(\hat{\theta})]+2[l\dot{\hat{\phi}}sin(\hat{\phi})\cdot l\dot{\hat{\theta}}sin(\hat{\theta})]\}$$

$$K_2 = \frac{m_2}{2}\{l^2\dot{\hat{\phi}}^2 + l^2\dot{\hat{\theta}}^2 + 2[l\dot{\hat{\phi}}cos(\hat{\phi})\cdot l\dot{\hat{\theta}}cos(\hat{\theta})] + 2[l\dot{\hat{\phi}}sin(\hat{\phi})\cdot l\dot{\hat{\theta}}sin(\hat{\theta})]\}$$

$$K_2 = \frac{m_2}{2}\{l^2\dot{\hat{\phi}}^2 + l^2\dot{\hat{\theta}}^2 + 2l^2(\dot{\hat{\phi}}\dot{\hat{\theta}})[cos(\hat{\phi})cos(\hat{\theta}) + sin(\hat{\phi})sin(\hat{\theta})]\}$$

Remembering the trigonometric identity $cos(x-y) = cos(x)cos(y) + sin(x)sin(y)$:

$$K_2 = \frac{m_2}{2}\left(l^2\dot{\hat{\phi}}^2 + l^2\dot{\hat{\theta}}^2 + 2l^2(\dot{\hat{\phi}}\dot{\hat{\theta}})cos(\hat{\theta}-\hat{\phi})\right)$$

And then the Lagrangian is written as:

$$L = \left[\frac{m_2}{2}\left(l^2\dot{\hat{\phi}}^2 + l^2\dot{\hat{\theta}}^2 + 2l^2(\dot{\hat{\phi}}\dot{\hat{\theta}})cos(\hat{\theta}-\hat{\phi})\right) + \frac{m_1}{2}l^2\dot{\hat{\theta}}^2\right] - [m_1\cdot g\cdot y_1 + m_2\cdot g\cdot y_2]$$

$$L = \frac{1}{2}(m_1+m_2)\cdot l^2\dot{\hat{\theta}}^2+\frac{1}{2}l^2m_2\dot{\hat{\phi}}^2+m_2l^2\dot{\hat{\phi}}\dot{\hat{\theta}}cos(\hat{\theta}-\hat{\phi})+gl\left((m_1+m_2)cos(\hat{\theta}) + m_2cos(\hat{\phi})\right) \quad \text{(B.18)}$$

Euler Lagrange

The equations of motion is determined from the Lagrangian using the Euler Lagrange differential equation, (see Equation B.19).

$$\frac{d}{dt}\left(\frac{\delta L}{\delta\dot{\theta}}\right) - \frac{\delta L}{\delta\theta} = 0 \quad \text{(B.19)}$$

The equation is derived for$\hat{\theta}$, and as the same method is used for$\hat{\phi}$the result is provided without intermediate steps.

$$\frac{\delta L}{\delta\hat{\theta}} = -m_2l^2\dot{\hat{\phi}}\dot{\hat{\theta}}sin(\hat{\theta}-\hat{\phi}) - gl(m_1+m_2)sin(\hat{\theta}) \quad \text{(B.20)}$$

$$\frac{\delta L}{\delta\dot{\hat{\theta}}} = (m_1+m_2)l^2\dot{\hat{\theta}} + m_2l^2\dot{\hat{\phi}}cos(\hat{\theta}-\hat{\phi}) \quad \text{(B.21)}$$

$$\frac{d}{dt}\left(\frac{\delta L}{\delta\dot{\hat{\theta}}}\right) = (m_1+m_2)l^2\ddot{\hat{\theta}} + m_2l^2\ddot{\hat{\phi}}cos(\hat{\theta}-\hat{\phi}) - m_2l^2\dot{\hat{\phi}}sin(\hat{\theta}-\hat{\phi})(\dot{\hat{\theta}}-\dot{\hat{\phi}}) \quad \text{(B.22)}$$

$$\left((m_1+m_2)l^2\ddot{\hat{\theta}} + m_2l^2\ddot{\hat{\phi}}cos(\hat{\theta}-\hat{\phi}) - m_2l^2\dot{\hat{\phi}}sin(\hat{\theta}-\hat{\phi})(\dot{\hat{\theta}}-\dot{\hat{\phi}})\right)-\left(-m_2l^2\dot{\hat{\phi}}\dot{\hat{\theta}}sin(\hat{\theta}-\hat{\phi}) - gl(m_1+m_2)sin(\hat{\theta})\right) = 0$$

which finally becomes:

$$(m_1 + m_2)l^2\ddot{\hat{\theta}} + m_2l^2\ddot{\hat{\phi}}cos(\hat{\theta} - \hat{\phi}) + m_2l^2\dot{\hat{\phi}}^2sin(\hat{\theta} - \hat{\phi}) + gl(m_1 + m_2)sin(\hat{\theta}) = 0 \tag{B.23}$$

and similarly for $\hat{\phi}$:

$$m_2l^2\ddot{\hat{\phi}} + m_2l^2\ddot{\hat{\theta}}cos(\hat{\theta} - \hat{\phi}) - m_2l^2\dot{\hat{\theta}}^2sin(\hat{\theta} - \hat{\phi}) + glm_2sin(\hat{\phi}) = 0 \tag{B.24}$$

Equation B.23and B.24 describes the motions of a double pendulum. On the biped model, one of the pendulums are inversed at initialization, and the ramp has an angle γ. This can be put into the model by setting $\hat{\theta} = \pi - \theta - \gamma$ and $\hat{\phi} = \phi - \gamma$.

The two coupled differential equations derived, are less straight forward to understand than the simple inverse pendulum used for describing the rimless wheel. In fact a double pendulum can exhibit chaotic movements, which by nature is hard to describe. However, considering the example where $m_2$is negligible, the equations of motion (B.23 and B.24) reduces to a single equation:

$$\ddot{\hat{\theta}} = \frac{-g}{l}sin(\hat{\theta})$$

This is the standard equation for a simple pendulum, and somewhat similar to Equation B.1, except that in the rimless wheel model, the radius of gyration was included. Hence it can be concluded that if the mass $m_2$is much smaller than $m_1$the basic motion of the hip resembles the motions of the inverted pendulum, which was also expected.

It is less obvious how the attached pendulum will behave, however inspection of Equation B.24 reveals that the motions exhibited by the free leg can be roughly interpreted as the sum of two separate parts:

$$\ddot{\hat{\phi}} = \left[\frac{-g}{l}sin(\hat{\phi})\right] + \left[\ddot{\hat{\theta}}cos(\hat{\theta} - \hat{\phi}) - \dot{\hat{\theta}}^2sin(\hat{\theta} - \hat{\phi})\right] \tag{B.25}$$

The first part resembles a normal pendulum, and the second part relates directly to the movements of the hip, hence if the hip movements are very slow, the contribution from this motion becomes very small, and the swing leg acts as a regular pendulum.

This is also what is intuitively expected.

B.3.2 Heel strike of a Straight Legged Biped Model

If the masses $m_1$and $m_2$are chosen appropriately, the two coupled pendulums will exactly complete the swing phase, such that the position of the legs are similar to the position of the spokes on the rimless wheel at time of impact. Hence Equation B.3 is still very applicable as a model for the heel strike angular speed loss, however, as r_{gyr} has not been defined for the simple double pendulum derived, the equation reduces to:

$$\omega^+ = cos(2\theta^-)\omega^- \tag{B.26}$$

Again it is observed that some of the hips angular speed is lost due to the impact at heel strike.

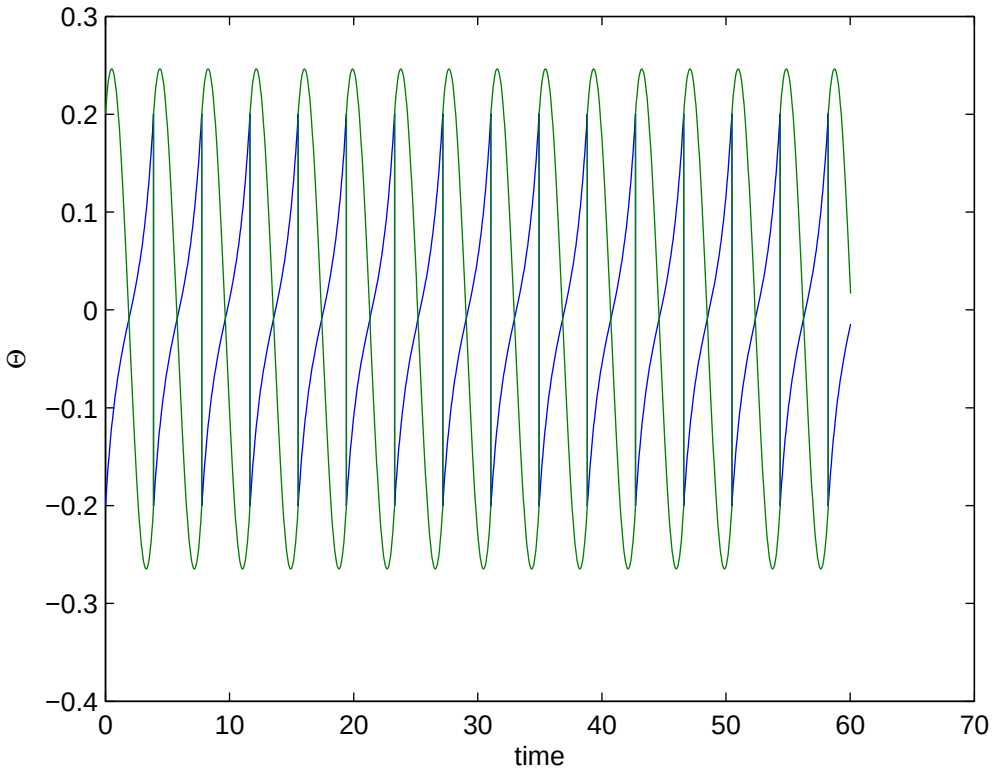


Figure B.9: The angle of the stance leg (blue) and the inter leg angle (green). The swing phase is shown for one step, and at time=4 the heel strike occurs, which is illustrated as a shift of the state of the legs. The simulation was based on Mariano Garcia's MATLAB implementation of a model similar to the one derived.

B.3.3 Conclusions on the Straight Legged Biped Model

When considering the example where the hip has a much larger mass than the feet, the motions of the hip and stance leg resembles that exhibited by the rimless wheel. Some dampening occurs in the swing phase due to the effect of adding a second pendulum, however the energy transfered to the added pendulum, results in a swinging motion that allows the biped to reestablish the angle between the legs before the biped falls over. Hence the system is able to walk down a slope.

The resulting dynamics resembles in principle the rimless wheel at impact, and is close to it in swing phase. Hence the observations regarding γ and the initial rotational velocity of the hip must be applicable for this model also.

For this model an additional initial condition must be fulfilled, which is that the swing leg must also be initialized at a position such that it can complete the first swing properly. However, if initialized properly, the biped will walk nicely down a ramp with raised support pads. This has been simulated and the results are shown in Figure B.9 and as a cartoon in Figure B.7

A controller has been implemented in python for the openDE simulation of the biped. It attempts walking on level slopes by regulating on the stance leg angle. For design and results

see Appendix G.

B.4 Kneed Passive Walker

If attempting to realize a biped walker, the need for the raised pads are troublesome, and inflexible in many ways. However the pads can be removed from the ramp by modifying the walking model. Robot developers have presented at least two different approaches which works reasonably well.

1. The swinging leg is shortened after it lifts from the ground, to avoid the ground, and the elongated just before heel strike occurs. This effects the equations of motion very little, if the retracting part of the leg is near mass less. Hence the mathematical model is already adequate in description of the motions.

2. The swinging pendulum is halved, and hinged together with a third pendulum. This gives the effect of a knee joint, and similar to human gaits the feet has ground clearance due to the extra swinging motion. This method complicates the equations of motion as a third coupled pendulum is introduced.

It is interesting to investigate the kneed model, as AAU-BOT1 which the final model should resemble, has knees.

B.4.1 Swing Phase of a Kneed Passive Walker

The model presented first is an extension of the one shown for the double pendulum, the method for deriving the equations of motion are exactly the same, however, for this derivation the indexes on the length have importance, so the geometry is re-defined:

$$x_1 = l_1 sin(\hat{\theta}) \tag{B.27}$$

$$y_1 = l_1 cos(\hat{\theta}) \tag{B.28}$$

$$x_2 = l_1 sin(\hat{\phi}) + l_2 sin(\hat{\theta}) \tag{B.29}$$

$$y_2 = l_1 cos(\hat{\phi}) + l_2 cos(\hat{\theta}) \tag{B.30}$$

$$x_3 = l_1 sin(\hat{\phi}) + l_2 sin(\hat{\theta}) + l_3 sin(\hat{\psi}) \tag{B.31}$$

$$y_3 = l_1 cos(\hat{\phi}) + l_2 cos(\hat{\theta}) + l_3 cos(\hat{\psi}) \tag{B.32}$$

Potential Energy

Equations B.9 and B.10 are already setup for potential energy, and are still valid as the geometry is re-defined, a third equation arises though:

$$P_3 = m_3 \cdot g \cdot y_3 \tag{B.33}$$

Kinetic Energy

The derivatives of the geometry is found as:

$$\dot{x}_1 = l_1\dot{\hat{\theta}}cos(\hat{\theta}) \tag{B.34}$$

$$\dot{y}_1 = -l_1\dot{\hat{\theta}}sin(\hat{\theta}) \tag{B.35}$$

$$\dot{x}_2 = l_2 \cdot \dot{\hat{\phi}}cos(\hat{\phi}) + \dot{x}_1 \tag{B.36}$$

$$\dot{y}_2 = -l_2 \cdot \dot{\hat{\phi}}sin(\hat{\phi}) + \dot{y}_1 \tag{B.37}$$

$$\dot{x}_3 = l_3 \cdot \dot{\hat{\psi}}cos(\hat{\psi}) + \dot{x}_2 \tag{B.38}$$

$$\dot{y}_3 = -l_3 \cdot \dot{\hat{\psi}}sin(\hat{\psi}) + \dot{y}_2 \tag{B.39}$$

and then the kinetic energy can be formed as:

$$K_1 = \frac{m_1(l_1^2 \cdot \dot{\hat{\theta}}^2[cos(\hat{\theta})^2 + sin(\hat{\theta})^2])}{2} = \frac{m_1}{2}l_1^2\dot{\hat{\theta}}^2 \tag{B.40}$$

$$K_2 = \frac{m_2}{2}\left(l_1^2\dot{\hat{\theta}}^2 + l_2^2\dot{\hat{\phi}}^2 + 2l_1l_2(\dot{\hat{\phi}}\dot{\hat{\theta}})cos(\hat{\theta} - \hat{\phi})\right) \tag{B.41}$$

$$K_3 = \frac{m_3}{2}\{[l_1 \cdot \dot{\hat{\theta}}cos(\hat{\theta}) + l_2 \cdot \dot{\hat{\phi}}cos(\hat{\phi}) + l_3 \cdot \dot{\hat{\psi}}cos(\hat{\psi})]^2 + [-l \cdot \dot{\hat{\phi}}sin(\hat{\phi}) - l \cdot \dot{\hat{\theta}}sin(\hat{\theta}) - l_3 \cdot \dot{\hat{\psi}}sin(\hat{\psi})]^2\}$$

expand

$$K_3 = \frac{m_3}{2}\{[l_1 \cdot \dot{\hat{\theta}}cos(\hat{\theta})]^2 + [l_2 \cdot \dot{\hat{\phi}}cos(\hat{\phi})]^2 + [l_3 \cdot \dot{\hat{\psi}}cos(\hat{\psi})]^2$$

$$+[-l_1 \cdot \dot{\hat{\theta}}sin(\hat{\theta})]^2 + [-l_2 \cdot \dot{\hat{\phi}}sin(\hat{\phi})]^2 + [-l_3 \cdot \dot{\hat{\psi}}sin(\hat{\psi})]^2$$

$$+2[l_1\dot{\hat{\theta}}l_2\hat{\phi}cos(\hat{\theta})cos(\hat{\phi})] + 2[l_1\dot{\hat{\theta}}l_2\hat{\psi}cos(\hat{\theta})cos(\hat{\psi})] + 2[l_1\dot{\hat{\phi}}l_2\hat{\psi}cos(\hat{\phi})cos(\hat{\psi})]$$

$$+2[l_1\dot{\hat{\theta}}l_2\hat{\phi}sin(\hat{\theta})sin(\hat{\phi})] + 2[l_1\dot{\hat{\theta}}l_2\hat{\psi}sin(\hat{\theta})sin(\hat{\psi})] + 2[l_1\dot{\hat{\phi}}l_2\hat{\psi}sin(\hat{\phi})sin(\hat{\psi})]\}$$

simplify

$$K_3 = \frac{m_3}{2}\{l_1^2\dot{\hat{\theta}}^2 + l_2^2\dot{\hat{\phi}}^2 + l_3^2\dot{\hat{\psi}}^2$$

$$+2([l_1\dot{\hat{\theta}}l_2\hat{\phi}cos(\hat{\theta})cos(\hat{\phi})] + [l_1\dot{\hat{\theta}}l_2\hat{\psi}cos(\hat{\theta})cos(\hat{\psi})] + [l_1\dot{\hat{\phi}}l_2\hat{\psi}cos(\hat{\phi})cos(\hat{\psi})]$$

$$+[l_1\dot{\hat{\theta}}l_2\hat{\phi}sin(\hat{\theta})sin(\hat{\phi})] + [l_1\dot{\hat{\theta}}l_2\hat{\psi}sin(\hat{\theta})sin(\hat{\psi})] + [l_1\dot{\hat{\phi}}l_2\hat{\psi}sin(\hat{\phi})sin(\hat{\psi})])\}$$

and simplifying again:

$$K_3 = \frac{m_3}{2}\{l_1^2\dot{\hat{\theta}}^2 + l_2^2\dot{\hat{\phi}}^2 + l_3^2\,\dot{\hat{\psi}}^2$$

$$+2(l_1\dot{\hat{\theta}}l_2\hat{\phi}[cos(\hat{\theta})cos(\hat{\phi})+sin(\hat{\theta})sin(\hat{\phi})]+l_1\dot{\hat{\theta}}l_2\hat{\psi}[cos(\hat{\theta})cos(\hat{\psi})+sin(\hat{\theta})sin(\hat{\psi})]+l_1\dot{\hat{\phi}}l_2\hat{\psi}[cos(\hat{\phi})cos(\hat{\psi})+sin(\hat{\phi})sin(\hat{\psi})]$$

and using trigonometry this reduces to:

$$K_3 = \frac{m_3}{2}\left(l_1^2\dot{\hat{\theta}}^2 + l_2^2\dot{\hat{\phi}}^2 + l_3^2\,\dot{\hat{\psi}}^2 + 2[l_1l_2(\dot{\hat{\phi}}\dot{\hat{\theta}})cos(\hat{\theta}-\hat{\phi}) + l_1l_3(\dot{\hat{\theta}}\dot{\hat{\psi}})cos(\hat{\theta}-\hat{\psi}) + l_2l_3(\dot{\hat{\phi}}\dot{\hat{\psi}})cos(\hat{\phi}-\hat{\psi})]\right) \tag{B.42}$$

The Lagrangian is now:

$$L = K - P = (K_1 + K_2 + K_3) - (P_1 + P_2 + P_2) \tag{B.43}$$

which expands to:

$$L = \frac{m_1}{2}l_1^2\dot{\hat{\theta}}^2 + \frac{m_2}{2}\left(l_1^2\dot{\hat{\theta}}^2 + l_2^2\dot{\hat{\phi}}^2 + 2l_1l_2(\dot{\hat{\phi}}\dot{\hat{\theta}})cos(\hat{\theta}-\hat{\phi})\right)$$

$$+\frac{m_3}{2}\left(l_1^2\dot{\hat{\theta}}^2 + l_2^2\dot{\hat{\phi}}^2 + l_3^2\,\dot{\hat{\psi}}^2 + 2[l_1l_2(\dot{\hat{\phi}}\dot{\hat{\theta}})cos(\hat{\theta}-\hat{\phi}) + l_1l_3(\dot{\hat{\theta}}\dot{\hat{\psi}})cos(\hat{\theta}-\hat{\psi}) + l_2l_3(\dot{\hat{\phi}}\dot{\hat{\psi}})cos(\hat{\phi}-\hat{\psi})]\right)$$

$$-\left(m_1gl_1cos(\hat{\theta}) + m_2g[l_1cos(\hat{\phi}) + l_2cos(\hat{\theta})] + m_3g[l_1cos(\hat{\phi}) + l_2cos(\hat{\theta}) + l_3cos(\hat{\psi})]\right)$$

and can be rewritten as:

$$L = \frac{1}{2}(m_1+m_2+m_3)l_1^2\dot{\hat{\theta}}^2 + \frac{1}{2}(m_2+m_3)l_2^2\dot{\hat{\phi}}^2 + \frac{1}{2}(m_3)l_3^2\,\dot{\hat{\psi}}^2$$

$$+m_2l_1l_2(\dot{\hat{\phi}}\dot{\hat{\theta}})cos(\hat{\theta}-\hat{\phi}) + m_3\left(l_1l_2(\dot{\hat{\phi}}\dot{\hat{\theta}})cos(\hat{\theta}-\hat{\phi}) + l_1l_3(\dot{\hat{\theta}}\dot{\hat{\psi}})cos(\hat{\theta}-\hat{\psi}) + l_2l_3(\dot{\hat{\phi}}\dot{\hat{\psi}})cos(\hat{\phi}-\hat{\psi})\right)$$

$$-m_1gl_1cos(\hat{\theta}) - m_2g[l_1cos(\hat{\phi}) + l_2cos(\hat{\theta})] - m_3g[l_1cos(\hat{\phi}) + l_2cos(\hat{\theta}) + l_3cos(\hat{\psi})]$$

which can be reduced to:

$$L = \frac{1}{2}(m_1+m_2+m_3)l_1^2\dot{\hat{\theta}}^2 + \frac{1}{2}(m_2+m_3)l_2^2\dot{\hat{\phi}}^2 + \frac{1}{2}(m_3)l_3^2\,\dot{\hat{\psi}}^2$$

$$+(m_2+m_3)l_1l_2(\dot{\hat{\phi}}\dot{\hat{\theta}})cos(\hat{\theta}-\hat{\phi}) + m_3\left(l_1l_3(\dot{\hat{\theta}}\dot{\hat{\psi}})cos(\hat{\theta}-\hat{\psi}) + l_2l_3(\dot{\hat{\phi}}\dot{\hat{\psi}})cos(\hat{\phi}-\hat{\psi})\right)$$

$$+g\left(-[m_1l_1 + m_2l_2 + m_3l_2]cos(\hat{\theta}) - [m_2l_1 + m_3l_1]cos(\hat{\phi}) - [m_3l_3]cos(\hat{\psi})\right)$$

Euler Lagrange

In the same manner as with the double pendulum, the equations of motion can be determined using Euler Lagrange see Equation B.19

$$\frac{\delta L}{\delta \hat{\theta}} = -(m_2+m_3)l_1l_2(\dot{\hat{\phi}}\dot{\hat{\theta}})sin(\hat{\theta}-\hat{\phi})-m_3l_1l_3(\dot{\hat{\theta}}\dot{\hat{\psi}})sin(\hat{\theta}-\hat{\psi})+g[m_1l_1+m_2l_2+m_3l_2]sin(\hat{\theta}) \quad \text{(B.44)}$$

$$\frac{\delta L}{\delta \dot{\hat{\theta}}} = (m_1+m_2+m_3)l_1^2\dot{\hat{\theta}} + (m_2+m_3)l_1l_2\dot{\hat{\phi}}cos(\hat{\theta}-\hat{\phi}) + m_3l_1l_3\dot{\hat{\psi}}cos(\hat{\theta}-\hat{\psi}) \quad \text{(B.45)}$$

$$\frac{d}{dt}\left(\frac{\delta L}{\delta \dot{\hat{\theta}}}\right) = (m_1+m_2+m_3)l_1^2\ddot{\hat{\theta}} + (m_2+m_3)l_1l_2\left(\ddot{\hat{\phi}}cos(\hat{\theta}-\hat{\phi}) - \dot{\hat{\phi}}sin(\hat{\theta}-\hat{\phi})(\dot{\hat{\theta}}-\dot{\hat{\phi}})\right) \quad \text{(B.46)}$$

$$+m_3l_1l_3\left(\ddot{\hat{\psi}}cos(\hat{\theta}-\hat{\psi}) - \dot{\hat{\psi}}sin(\hat{\theta}-\hat{\psi})(\dot{\hat{\theta}}-\dot{\hat{\psi}})\right)$$

remembering the Euler Lagrange:

$$\frac{d}{dt}\left(\frac{\delta L}{\delta \dot{\hat{\theta}}}\right) - \frac{\delta L}{\delta \hat{\theta}} = 0$$

Substituting the equation of motion relating to θ:

$$(m_1+m_2+m_3)l_1^2\ddot{\hat{\theta}} + (m_2+m_3)l_1l_2\left(\ddot{\hat{\phi}}cos(\hat{\theta}-\hat{\phi}) - \dot{\hat{\phi}}sin(\hat{\theta}-\hat{\phi})(\dot{\hat{\theta}}-\dot{\hat{\phi}})\right) \quad \text{(B.47)}$$

$$+m_3l_1l_3\left(\ddot{\hat{\psi}}cos(\hat{\theta}-\hat{\psi}) - \dot{\hat{\psi}}sin(\hat{\theta}-\hat{\psi})(\dot{\hat{\theta}}-\dot{\hat{\psi}})\right)$$

$$+(m_2+m_3)l_1l_2(\dot{\hat{\phi}}\dot{\hat{\theta}})sin(\hat{\theta}-\hat{\phi}) + m_3l_1l_3(\dot{\hat{\theta}}\dot{\hat{\psi}})sin(\hat{\theta}-\hat{\psi}) - g[m_1l_1+m_2l_2+m_3l_2]sin(\hat{\theta}) = 0$$

And for ϕ the Equation of motion becomes:

$$(m_2+m_3)l_2^2\ddot{\hat{\phi}} + (m_2+m_3)l_1l_2\left(\ddot{\hat{\theta}}cos(\hat{\theta}-\hat{\phi}) - \dot{\hat{\theta}}sin(\hat{\theta}-\hat{\phi})(\dot{\hat{\theta}}-\dot{\hat{\phi}})\right) \quad \text{(B.48)}$$

$$+m_3l_2l_3\left(\ddot{\hat{\psi}}cos(\hat{\phi}-\hat{\psi}) - \dot{\hat{\psi}}sin(\hat{\phi}-\hat{\psi})(\dot{\hat{\phi}}-\dot{\hat{\psi}})\right)$$

$$+(m_2+m_3)l_1l_2(\dot{\hat{\phi}}\dot{\hat{\theta}})sin(\hat{\theta}-\hat{\phi}) + m_3l_2l_3(\dot{\hat{\phi}}\dot{\hat{\psi}})sin(\hat{\phi}-\hat{\psi}) - g[m_2l_1+m_3l_1]sin(\hat{\phi}) = 0$$

Finally for ψit becomes:

$$m_3l_3^2\ddot{\hat{\psi}} + m_3l_1l_3\left(\ddot{\hat{\theta}}cos(\hat{\theta}-\hat{\psi}) - \dot{\hat{\theta}}sin(\hat{\theta}-\hat{\psi})(\dot{\hat{\theta}}-\dot{\hat{\psi}})\right) \quad \text{(B.49)}$$

$$+m_3l_2l_3\left(\ddot{\hat{\phi}}cos(\hat{\phi}-\hat{\psi}) - \dot{\hat{\phi}}sin(\hat{\phi}-\hat{\psi})(\dot{\hat{\phi}}-\dot{\hat{\psi}})\right)$$

$$+m_3l_1l_3(\dot{\hat{\theta}}\dot{\hat{\psi}})sin(\hat{\theta}-\hat{\psi}) + m_3l_2l_3(\dot{\hat{\phi}}\dot{\hat{\psi}})sin(\hat{\phi}-\hat{\psi}) - g[m_3l_3]sin(\hat{\psi}) = 0$$

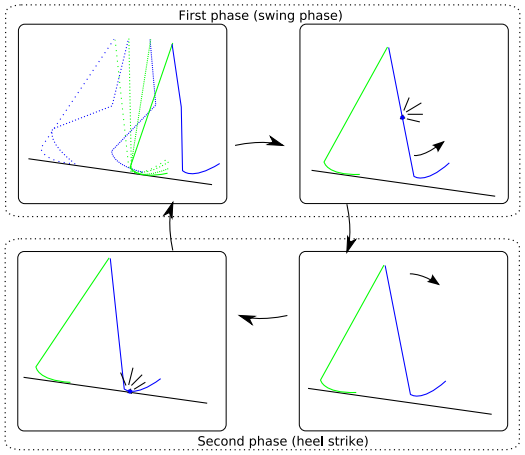


Figure B.10: The two phases of walking for the passive 2d walker with knees. Inspired by [34]. Radius for the feet is often chosen as the leg length.

The three equations of motion (B.47, B.48 and B.49), illustrates that as more pendulums are attached in a chain, the effects of the individual pendulum becomes increasingly harder to isolate from the other.

The double pendulum model was explored from the case where the hip was much heavier than the feet. This is expected by a walking robot, and using the same approach on this triple pendulum model it is still obvious, as shown in Equation B.50 that with very small masses m_2 and m_3, the behavior of the hip in the swing phase will resemble the inverted pendulum.

$$m_1 l_1^2 \ddot{\hat{\theta}} - g m_1 l_1 sin(\hat{\theta}) = 0 \tag{B.50}$$

The motions of the two suspended pendulums, are not easy to describe as separate movements, the equations of motion predict chaotic behavior, and the mass distribution of the two links are hard to consider as being extremely different from each other. If they where however, the two different mass distribution possibilities defines which pendulum is the dominating one.

In fact, the knee will hyper extend if this is the only model used. Hence it can be claimed that this model alone does not describe the entire swing phase, as the knee by design, limits the possible rotations in the knee joint.

This is handled in Section B.4.2.

B.4.2 Knee strike of a kneed passive walker

A model of the knee strike is in nature somewhat similar to the heel strike. However, when the knee strike is introduced, the phases of the model needs to be re-defined. The model is recreated from [34], as simulations indicate that the model is solid.

The swing phase now includes the knee strike, and the heel strike phase now includes the swing downwards from knee strike to heel strike. As visualized in Figure B.10

The model presented includes the location of theCoM, and thus also the moment of inertia. These parameters were not included in the simple derivation presented in the previous section. These are here introduced as the variables a and I . Also the viscous dampening effect $\lambda$is included in this model for the knee.

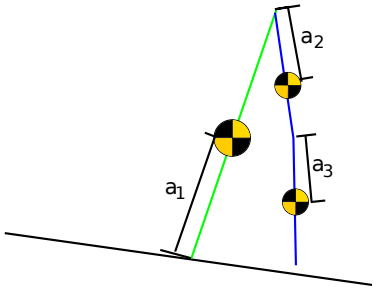


Figure B.11: The model of a kneed walker, with CoM locations drawed.

On Figure B.11 the location of the CoM is seen. The derivation is not shown here for the new setup, but the resulting equations of motion for the swing phase is shown below.

The Force Balance Equations on Link 3:

$$m_3a_3\left(l_1cos(\psi-\theta)\ddot{\theta}+l_2cos(\psi-\phi)\ddot{\phi}+\right)+(m_3a_3^2+I_3)\ddot{\psi} \tag{B.51}$$

$$+[m_3a_3l_1sin(\psi-\theta)\dot{\theta}]\dot{\theta}+[m_3a_3l_2sin(\psi-\phi)\dot{\phi}-\lambda_3]\dot{\phi}$$

$$+\lambda_3\dot{\psi}+m_3a_3g\cdot sin(\psi)=0$$

And the Moment Balance about center of mass of link 2:

$$l_1(m_3l_2+m_2a_2)cos(\phi-\theta)\ddot{\theta}+(I_2+m_2a_2^2+m_3l_2^2)\ddot{\phi}+m_3a_3l_2cos(\psi-\phi)\ddot{\psi} \tag{B.52}$$

$$+(m_3l_2+m_2a_2)l_1sin(\phi-\theta)\theta^2+\dot{\lambda}_3\dot{\phi}[-m_3a_3l_2sin(\psi-\phi)]\dot{\psi}-\lambda_3]\dot{\psi}$$

$$+(m_3l_2+m_2a_2)g\cdot sin(\phi)=T_2$$

Moment relation to the pivot point, where the stance leg touches the ground:

$$(I_1+m_1a_1^2+m_2l_1^2+m_3l_1^2)\ddot{\theta}+(m_3l_2+m_2a_2)l_1cos(\phi-\theta)\ddot{\phi}+m_3a_3l_1cos(\psi-\theta)\ddot{\psi} \tag{B.53}$$

$$-(m_3l_2+m_2a_2)l_1sin(\phi-\theta)\dot{\phi}^2-m_3a_3l_1sin(\psi-\theta)\dot{\psi}^2$$

$$+(m_1a_1+m_2l_1+m_3l_1)g\cdot sin(\theta)=-T_2$$

According to [34], writing the three equations on matrix form allows for easy expression of the knee strike effect.

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{23} & M_{32} & M_{33} \end{bmatrix}\begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \\ \ddot{\psi} \end{bmatrix}+\begin{bmatrix} 0 & C_{12}\dot{\phi} & C_{13}\dot{\psi} \\ -C_{12}\dot{\theta} & \lambda_3 & C_{23}\dot{\psi}-\lambda_3 \\ -C_{13}\dot{\theta} & -C_{23}\dot{\phi}-\lambda_3 & \lambda_3 \end{bmatrix}\begin{bmatrix} \theta \\ \phi \\ \psi \end{bmatrix}+\begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix}g=\begin{bmatrix} -T_2 \\ T_2 \\ 0 \end{bmatrix} \tag{B.54}$$

Where:

$$\begin{aligned}
M_{11} &= I_1 + m_1 a_1^2 + (m_2 + m_3) l_1^2 \\
M_{12} &= (m_2 a_2 + m_3 l_2) l_1 cos(\phi - \theta) \\
M_{13} &= m_3 a_3 l_1 cos(\psi - \theta) \\
M_{22} &= I_2 + m_2 a_2^2 + m_3 l_2^2 \\
M_{23} &= m_3 a_3 l_2 cos(\psi - \phi) \\
M_{33} &= I_3 + m_3 a_3^2 \\
C_{12} &= -(m_2 a_2 + m_3 l_2) l_1 sin(\phi - \theta) \\
C_{13} &= -m_3 a_3 l_1 sin(\psi - \theta) \\
C_{23} &= -m_3 a_3 l_2 sin(\psi - \phi) \\
K_1 &= (m_1 a_1 + (m_2 + m_3) l_1) sin(\theta) \\
K_2 &= (m_2 a_2 + m_3 l_1) sin(\phi) \\
K_3 &= (m_3 a_3) sin(\psi) \\
T_2 &= -k\psi
\end{aligned}$$

Now the knee strike can simply be modelled as:

$$\begin{bmatrix} \omega_{1^+} \\ \omega_2^+ \\ \omega_3^+ \end{bmatrix} = \begin{bmatrix} \omega_{1^-} \\ \omega_2^- \\ \omega_3^- \end{bmatrix} - M^{-1} \begin{bmatrix} 0 \\ -\tau \\ \tau \end{bmatrix}$$

where τ is the hip torque.

After the knee strike the knee is locked, and for the entire phase 2, the model can be described using the equations of motion derived for the straight legged biped model in Section B.3.

C

Muscle EMG Patterns During Walking, a Pedagoical Interpretation

This appendice treats the results which are presented in the articles [42] and [41]. While the methods and results are nicely presented in the two articles, and not questioned by the authors of this thesis, it can however become necessary for non-bio-mechanical engineers to introduce some human anatomy to understand the results.

As an example, consider Figure C.1, which is copied from [41]. It contains alot of information about the intensity of actuation signals recieved by a particular muscle. However, without understanding the Short-hand notation of the muscles, and without intuitive knowledge about where those muscles are located, the data is almost useless. The authors and the audience for the articles should have this knowledge, but the authors and audience for this thesis is not expected to have extensive knowledge about the human anatomy. Hence the need for this appendice.

The two articles treated are written by the same team of researchers, and claims that five major EMG patterns exists when humans are walking, and that a motor program which sequences the patterns can be constructed quite simply.

This has interesting perspectives for this thesis, as a similar approach can be used for actuating the robot, and thus control the basic pattern of the dynamic walk. The motor program presented in the articles is interpretable as a state-machine controller, and as mentioned in Section 2.3 passive walkers which achieves walking on level ground using a state-machine controller already exists.

Unfortunately the state machines used for those robots are presented in articles and documentation as a black boxes, and a new state machine had to be concieved for AAU-BOT1. It was natural to look at the EMG patterns measured in humans, as AAU-BOT1 was to achieve human like walk.

The acronyms used in the articles, can re-appear in this thesis, however as alot of other acronyms are also used, theese are prefixed by an "m" in the thesis.

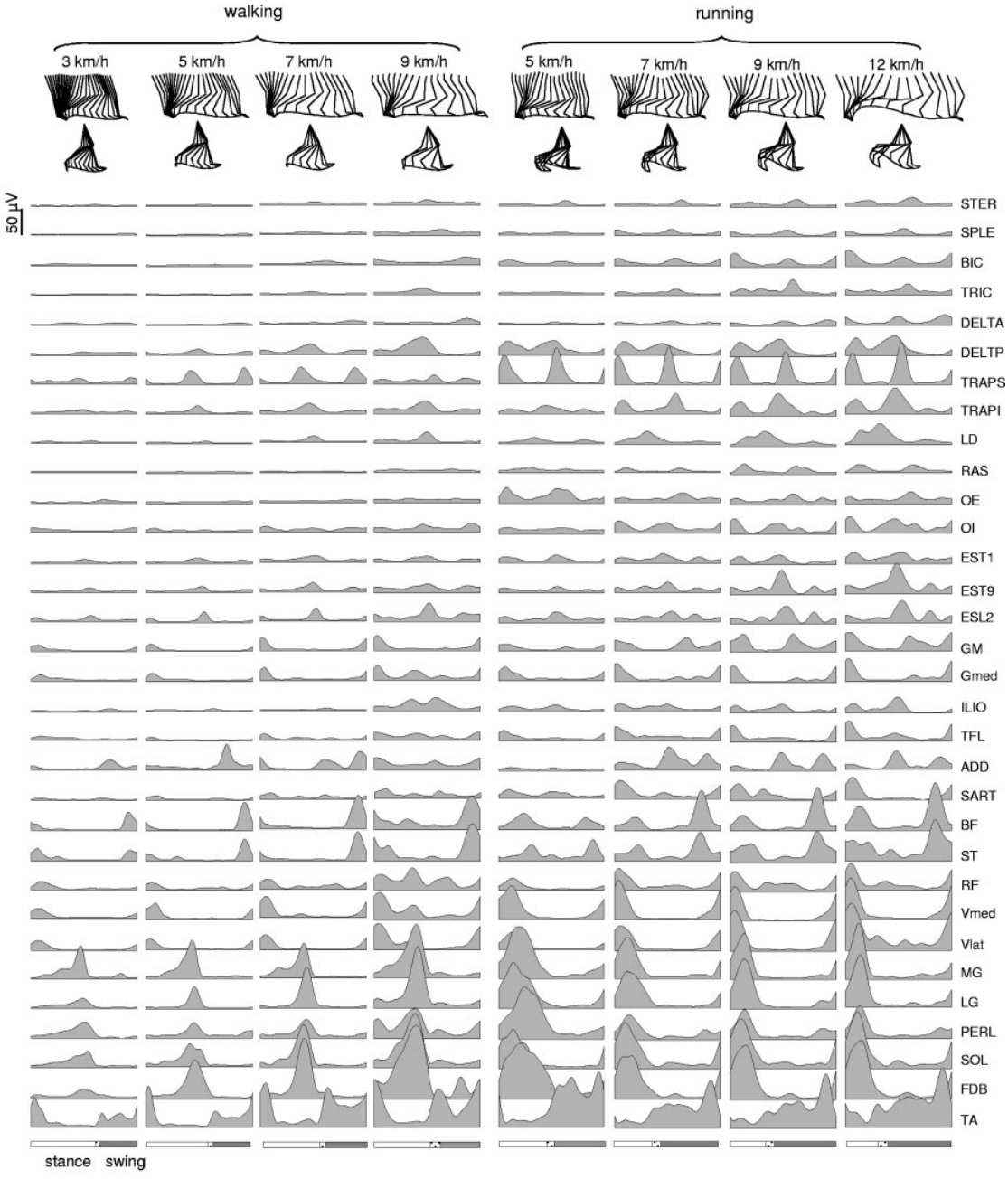
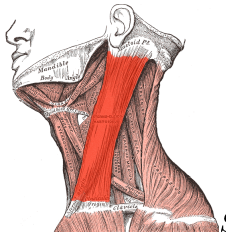


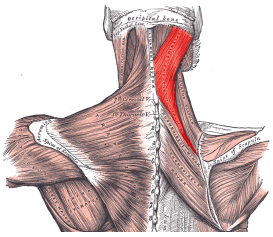
Figure C.1: Borrowed from [41], illustrates the normalized amount of actuation signals to each of the measured muscles in a timespan of one step.

C.1 Muscle Acronyms List and Definitions

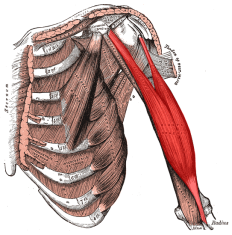
Images shown in this list, is borrowed from **Anatomy of the Human Body** by Henry Gray (1918), if nothing else is mentioned explicitly.



STER *Sternocleido Mastoideus.* When acting alone, it tilts head to its own side and in the same time rotates it so the face is turned towards the opposite side. Acting together, flexes the neck, raises the breast bone and assists in forced inspiration.

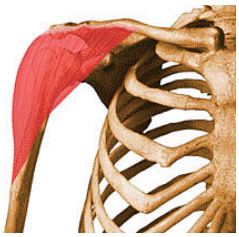


SPLE *Splenious.* Acting alone tilts the head to it's own side and rotates the head so the face is turned towards it's own side. Acting together pitches the head in relation to the spine.

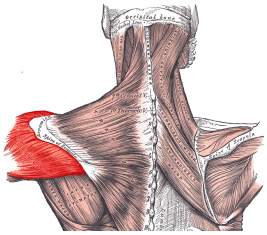


BIC *Biceps Brachii.* Attached to the shoulder and the inner(front) side of the elbow, flexes elbow and supinates forearm.

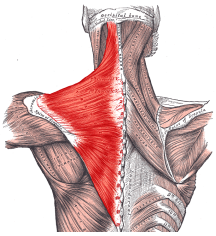
TRIC *Triceps Brachii.* Attached to the shoulder and the outer(back) side of the elbow, extends forearm by pulling in the shoulder



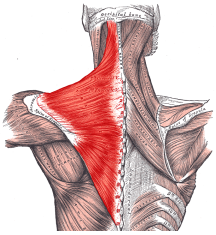
DELTA *Deltoideus Anterior.* Attached to the shoulder, and the upper arm bone, front. Responsible for shoulder abduction, flexion and extension.



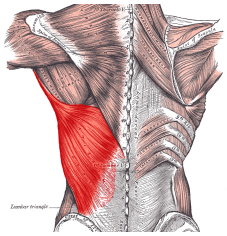
DELTP *Deltoideus Posterior.* Attached to the shoulder, and the upper arm bone. Responsible for shoulder abduction, flexion and extension.



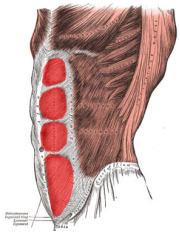
TRAPS *Trapezius Superior.* Connects shoulder to spinal cord. Works on retraction of scapula (shoulderblades)



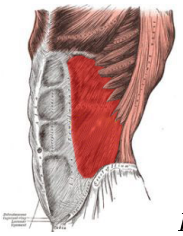
TRAPI *Trapezius Inferior.* Connects shoulder to spinal cord. Works on retraction of scapula (shoulderblades)



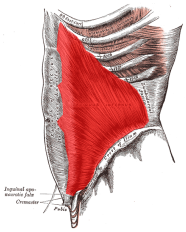
LD *Latissimus Dorsi.* Crosses from lower spine to arm, twists the torso. Pulls the forelimb towards the spine



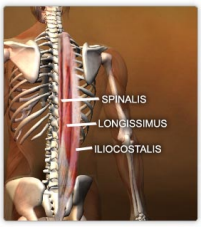
RAS *Rectus Abdominus.* It is responsible for flexing the lumbar spine, as when doing a "crunch". The rectus abdominis assists with breathing and plays an important role in respiration



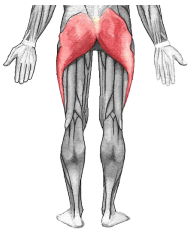
OE *External Oblique.* Rotates torso.



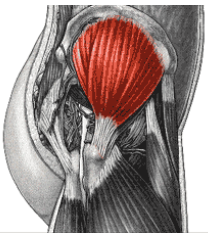
OI *Internal Oblique.* Compresses abdomen and rotates vertebral column.



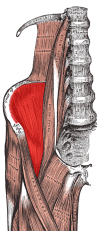
EST *Erector Spinae*, also known as sacrospinalis, the numbers indicate where on the spinal cord measurements are done. Flexes and stabilizes the spine



GM *Gluteus Maximus.* Connects thighs with rear pelvis. Does external rotation and extension of the hip joint, supports the extended knee, and chief antigravity muscle in sitting

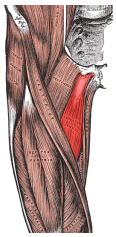


Gmed *Gluteus Medius.* Connects pelvis with spinal cord. Aids in abduction of the hip; and preventing adduction of the hip. Performs medial rotation of thigh



ILIO *Iliopsoas.* Connects inner side of hip to thigh. Creates flexion of the hip, and spine rotation

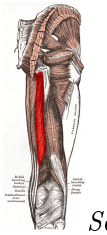
TFL *Tensor Fascia Latae.* Connect outer side of hip with knee, moves legs outwards by flexion in the hip.



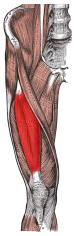
ADD *Adductor Longus.* Inside of thigh, moves legs inwards (adduction of thigh)

SART *Sartorius.* Rotates leg and aids in flexion, combined with other muscles, it aids in a multitude of knee/thigh flex and rotations.

BF *Biceps Femoris.* Connects on the rear of the thigh to the tibia, and flexes the knee joint, and extends hip joint.

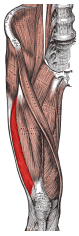


ST *Semitendinosus.* Connects on the rear of the thigh to the tibia, and flexes the knee, and extends hip joint.



RF *Rectus Femoris.* Connects front of hip with knee, stretches knee (knee extension) and flexes in the hip

Vmed *Vastus Medialis.* Extends the knee.

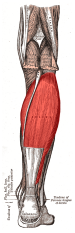


Vlat *Vastus Lateralis.* Extends and stabilizes the knee.

MG *Gastrocnemius Medialis* (and Lateralis) bot connects to the achilles tendron, and to the knee. Plantar flexes (bendes foot towards shin), and does some flexing of the knee.

LG *Gastrocnemius Lateralis.* see **MG**

PERL *Peroneus Longus.* Twists the ankle in the frontal plane, and Plantar flexes.



SOL *Soleus.* Plantar Flexes, and supports the achilles tendron.

FDB *Flexor Digitorum Brevis.* Flexes the toes.

TA *Tibialis Anterior. Dorsiflexes*, (stretches the foot) and inverts the foot

C.2 Physical Interpretation of the EMG Signals

The data is presented both for running and walking, AAU-BOT1 is supposed to be walking at 1 m/s which is equivalent to 3.6 km/h. It is chosen to illustrate the active muscles, by focussing on the EMG measurements for 3km/h.

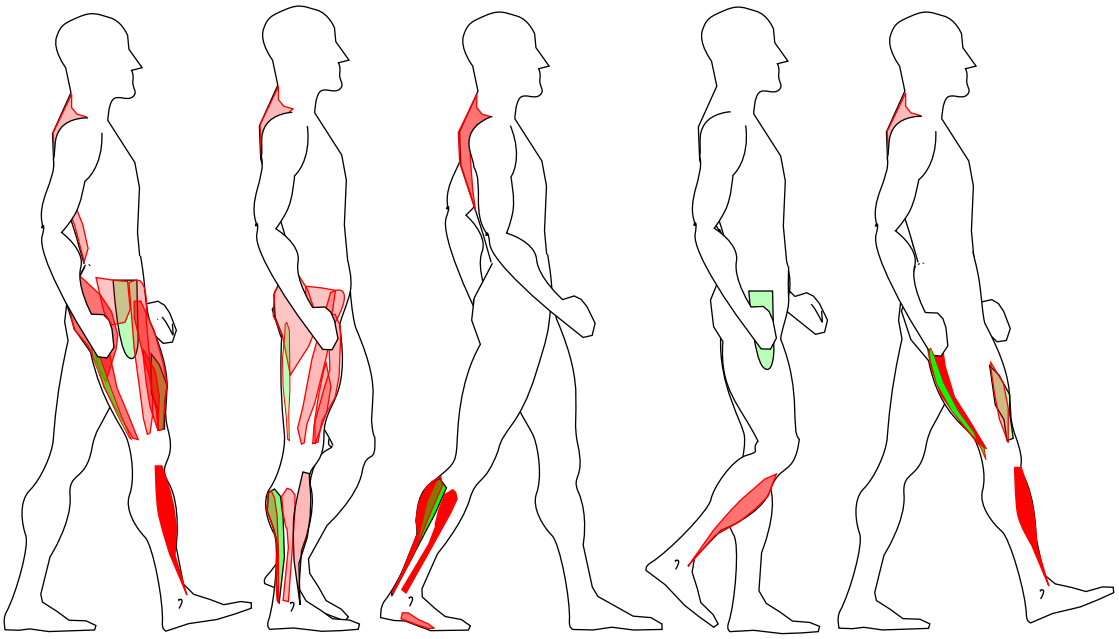


Figure C.2: Larger muscles which have a significantly high EMG signal, in comparison to noise level. Green muscles are inside of leg, Red is front, back or outside of leg.

Without considering the muscles ability to translate the actuation signals into force, some physical understanding can still be extracted. The data has not been made directly available to this study, and hence the only knowledge which can be extracted from the available graphs, are which muscles are active at a given time. The amount of force they deliver cannot be estimated very accurately from the presented dataset, fortunately the excact figures for the force/signal relations are irrellevant to this thesis, as a robot and not a human body is to be controlled. Further more biomechanical studies has so far not given a clear relationship between EMG signals and muscle force, it has though proven that the occurence of an EMG signal relates as an on/off indicator of the activity of a muscle[16].

The knowledge about which muscles are activated, can be used to conceptualise when and where to actuate on the trajectory generating robot model. The amount of force which is to be applied can then afterwards be determined specifically for each task.

The Muscle Acronyms List presented in subsection C.1 provides the necessary knowledge about the anatomy to place the EMG measurements as a colorizasiont of the relating muscles in a stylized human. The timescale for the data allows for identifying the walking phases, and thus the posture of the human, as this can be extracted from other sources i.e [16] and [10]. The result of this method for locating the active muscles is presented in Figure C.2, and is indicated in the following lists of active muscles for each phase. Muscles appear in the list, where a signal is within thresholds, and is followed by a letter indicating which threshold. (s,m,l) relating to quantities (small, medium, large). In Figure C.2, this letter relates to the intensity of the color, such that pink is small, and dark red is large.

Heel Strike TRAPS (s), EST2(s), GM(m), Gmed(s), TFL(s), ADD(s), BF(m), ST(m), RF(m), Vmed(m), Vlat(m), TA(l).

Midstance TRAPS (s), GM(s), Gmed(s), TFL(s), ST(s), RF(s), Vlat(s), MG(m), PERL(s), SOL(s),TA(s).

Heel Off TRAPS(m), MG(l), LG(m), PERL(l), SOL(l), FDB(m).

Toe Off ADD(m), TA(m).

Swing Phase acc ADD(m), TA(m).

Swing Phase decc TRAPS(s), BF(l), ST(l), Vmed(s), Vlat(s), TA(l).

From the list, and the figure, it is quite clear that alot of muscle control happens at heel strike, presumeably to absorb the sudden impact with the ground. Toe off and the acceleration phase in the swing phase, have almost similar muscle groups activated, which leads to a five phase model of the signal sequencing found in the article.

D

Impact at Heel Strike

To avoid damaging the robot, the forces and torques experienced by robot, and especially the FTS in the feet, must not exceed their limits.

During swing phase, this is plainly un-interesting as the swinging foot experiences allmost no load. The forces and torques experienced by the stance foot, during the single support can be found by a variety of models, as the ground contact is continous. However at heel strike the foot suddenly acquires contact, and this has dramatic effects.

If assuming that the impact is inelastic, or plastic, then an impact is not defined with forces, but only as an instantaneous velocity change to zero, as a result the kinetic energy suddenly drops. If assuming totally elastic impact, the energy is conserved, but the direction of motion is suddenly changed. To realise an elastic impact requires infinite force from the floor at the excact time of impact.

None of the two classic impact models are really modelling the impact very well, as it in both cases it is impossible to handle the requirements to a maximal impact force in any reasonable way.

In [30], a model of the ground was implemented based on springs. Such a ground model is interesting for impact modelling, as the impact with an ideal spring does not occur instantaneous, but instead a duration of time passes while the spring is compressed, and during this time the force provided by the spring grows from zero to maximum.

Such a model can be used to determine the forces and torques which the feet experiences during an impact, and thus be used to test different schemes for absorbing the impact force. If the heel strike is modelled as the expected impact while walking, the model parameters cannot be verified before the walk is achieved. This poses a problem, as the parameters for the ground springs are unknown, and thus the outputs of the model are not reliable.

To determine suitable parameters for the ground springs, an experiment is set up which also verifies the FTS dynamics. A more specific describtion of the experiment can be found in Appendice E.2, the results found in the experiments are used to determine the simulation model parameters.

However as the simulink simulation model uses zero crossing, and openDE does not, the results are interpreted below to generate the best possible fit for both engines.

The model of the impact is basicly modelled as illustrated in Figure D.1.

The model to predict the impact force experienced by the foot is a rigid body that models the foot, that experiences a vertically downwards force from the gravity working on the rest of AAU-BOT1 which pushes the rigid body down into the ground. The rigid body also experiences the force from the springs located at the foot vertices, pushing vertically upwards.

Both forces affect the motions of the rigid body CoM, and creates torques around it's CoM causing it to move. If $\vec{x}$ denotes the vector which locates CoM of the foot, then the equations of motion are:

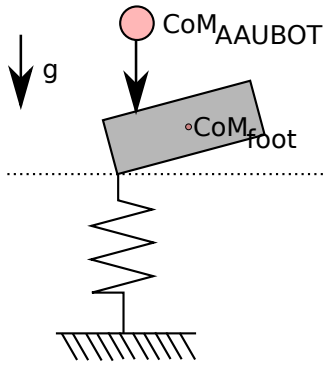


Figure D.1: Simple foot model of a single rigid body, which experiences forces from the spring used for modelling the ground contact, and the AAU-BOT1's CoM and gravity.

$$\begin{aligned}
\dot{\vec{x}} &= \vec{v} \\
\dot{\vec{v}} &= \frac{\vec{F}}{m} \\
\dot{R(t)} &= skew[\omega(t)] \cdot R(t) \\
\dot{\vec{\omega}} &= I^{-1}[\vec{\tau} - \vec{\omega} \times I\vec{\omega}]
\end{aligned}$$

Where $\vec{F}$ is the net sum of all the forces and $\vec{\tau}$ is the net sum of all the torques.

The system is set up in openDE, where springs are automatically attached to each of the four vertices that have contact with the ground, and the spring constant is chosen to match the impact measurements as best possible.

The ground model is afterwards used to predict the impact force at different impact speeds and leg configurations, as it is much faster than the simulink simulation. The forces are applied to the rigid body as:

$$\begin{aligned}
\vec{F}(t) &= \vec{F_g}(t) + \vec{F}_{AAU-BOT1}(t) + \vec{F_{Spring}}(t) \\
\vec{\tau}_{Spring}(t) &= \vec{r} \times \vec{F_{Spring}}(t) \\
\vec{\tau}_{AAU-BOT1}(t) &= \vec{r} \times \vec{F}_{AAU-BOT1}(t) \\
\vec{\tau}(t) &= \vec{\tau}_{Spring}(t) + \vec{\tau}_{AAU-BOT1}(t)
\end{aligned}$$

The foot is not allowed to turn or move through the floor, as a spring is attached at the toes excactly when the surface is touched. The effect of this is triviel to illustrate. The interesting force to retrieve from this model is $\vec{F_{Spring}}$, this is so because the foot is not deformed, and thus the FTS on top of the foot must experience the same force as the foot does.

Through experiments it was shown that the duration of the impact is below 1 sample, and as a result the recorded amplitude of the impact force behaves in a manner close to random. Hence estimating the Ground spring and Damper coeffecients is almost impossible with the available equipment.

E

Journals

E.1 Journal Foot Spring Parameter Estimation

The purpose of this appendice is to present the model, the applied experiments, and the experiment data, used for estimating the foot spring parameters.

Previous attempts at walking with AAU-BOT1 has been focussed on static walking, and thus it has been assumed so far that the foot is a solid unit. However for the task of walking like humans, a heel - toe approach to ground contact should be applied, and thus the toe will sometimes be the only part of the foot touching the ground.

The toe is mounted to the foot at the top of the toe plate by a simple hinge joint, and a spring is attached between the foot and the toe. The springs purpose is to ensure that it returns to the initial rotation when ground contact is lost. The spring will effect the robots dynamics when it stands on the toe, and it will determine how fast the toe settles. The effect of the foot spring is expected to be very small, as the spring it self is small. Still it is not known how small, and it is necessesary to have a spring in simulation to ensure that the toe actually do return to initial positions when in the air, hence the parameters are estimated by experiments.

E.1.1 Test Setup

In Figure E.1 a sketch of the test rig is presented. The spring model applied in the simulation assumes a spring constant, and a damper coeffecient. Hence a test setup is presented which attempts to determine those parameters.

A mass is attached to the spring, and it is release from a fully contracted spring, while the position is recorded. In the model the spring constant results in a sine wave motion in the vertical axis, and the damper coeffecient results in a dampening of the amplitude which finally stops the motion.

The data is recorded using a Sony DSC-P200 Digital Camera which takes an image of the position of the mass at approximately 20Hz, the spring it self is attached to a static rig. Behind the spring and mass, a simple ruler is placed, to vizually confirm the image analysis tool.

The mass used was a 1.81 Kg mass.
The image analysis tool used was able to extract features regarding the mass and attachement wire. A frame from the camera can be seen on Figure E.2. The points detected are transferred to a dataformat that MATLAB accepts, and then converted from pixels to meters using the photographed ruler for reference.

E.1.2 Results

The resulting motion is fitted using senstools on a standard 2nd order model.

The result of the fit is shown in Figure E.3.

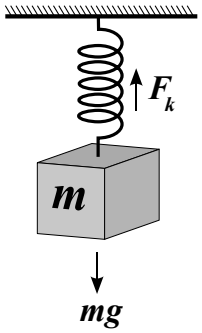


Figure E.1: Stylised sketch of the test setup.

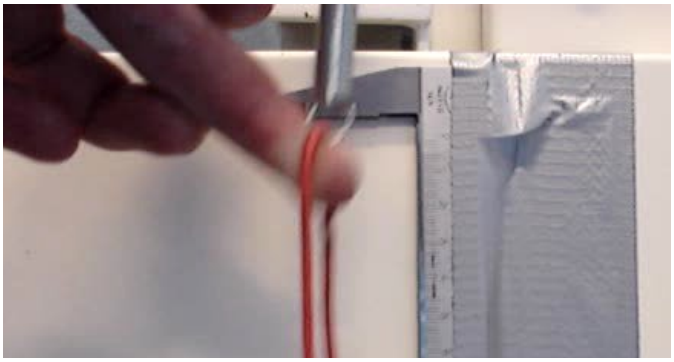


Figure E.2: Example sample image taken from the image stream, just before initialisation of the experiment.

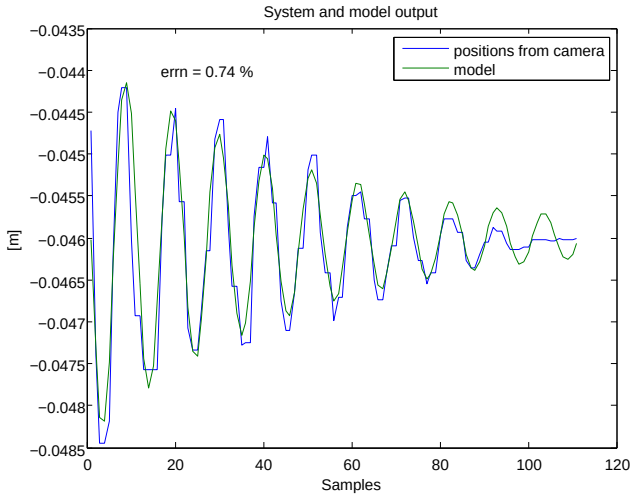


Figure E.3: The recorded data together with the estimated model behaviour.

The spring constant was determined as 403.9202, and the damper coeffecient was determined as 1.8164.

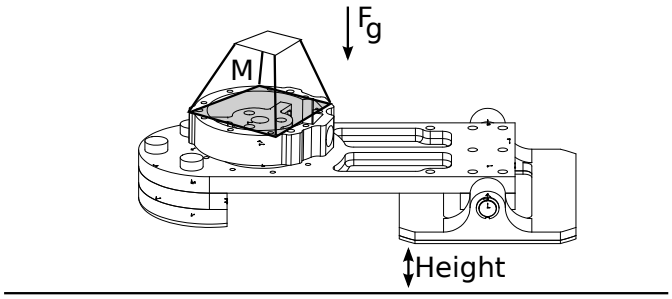


Figure E.4: Stylised sketch of the test setup.

E.2 Journal Impact Measurement

The purpose of this appendice is to present the method used for verifying the dynamics of the FTS and the simulation model of the sensor, and the ground, in relation to estimating the forces which arises at impacts with the ground. After the method is presented the measurement data is shown and compared with the simulation model.

The experiment did not yeild the desired results, and the results was that only steady state information could be used in the modelling of ground and FTS.

The calibration of the FTS has been done by previous groups, by utillising steady state measurements of the FTS in a static load. Hence the steady state resolution and accuracy of the FTS has been well established. It has not however been tested how fast the FTS performs during an impact with the ground.

By using the foot and toe assembly to generate a series of impacts, (using a free fall from a known altitude), and recording the FTS data, a test is presented which can be repeated in simulations, see Figure E.4. The testrig is simplified by replacing the ankle and thus the robot with a simple shaped mass, to reduce the amount of dynamic parameters and reduce errors which can occur in the dynamic model. The altitude of the free falling drop is chosen to be as low as possible to reduce the influence from the cables required to retrieve the FTS data.

The simulation model can be seen in E.5.

The foot has been fitted with the IMU sensor for reference measurements of the accelerations, as this can be used to determine how well the FTS performs, and how well the ground impact model performs.

It is deductable from the experimental results that the calibration is adequate for steadystate, but that the impact is done within a single sample for all the experiments carried out. This unfortunately means that the peak amplitude is not thrustworthy for calibration of ground models.

The simulation box parametres was: H = 1.995cm, L = 6.095cm, W = 8.03cm.

The inertia of a box is found as: $I_{xx} = \frac{1}{12}m(b^2+c^2); I_{yy} = \frac{1}{12}m(a^2+c^2); I_{zz} = \frac{1}{12}m(a^2+b^2)$

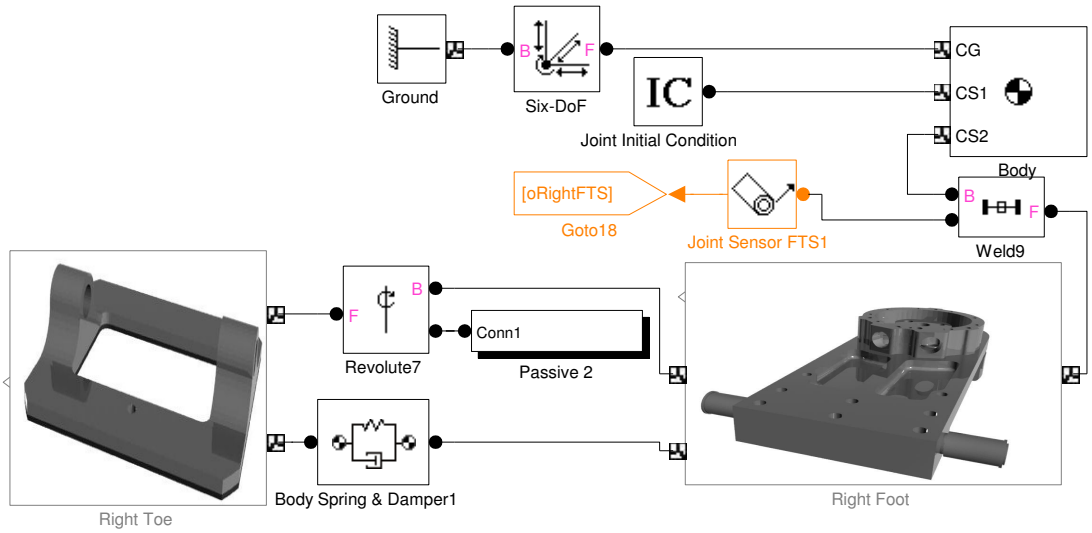


Figure E.5: The simulink simulation model used for sensor comparison..

E.3 Journal Ground Friction Measurement

The purpose of this appendice is to present the method used for determining the parameters for the Ground friction model.

The classical model is composed of a stiction and a viscous friction parameter. The ground which the robot walks on has a tendency to change friction behavior as it becomes dirty, wet, or simply worn. Even more important is that the robot can encounter different ground materials, which in nature behaves different.

Thus a parameterized model can at best, only model a particular area of the ground, and only at a particular time, it is however expected by the group that this known error in model parameters is small, and can be neglegted for the purpose of simulating walking, hence it is assumed that the parameters can be approximated roughly, and still provide a reasonable estimate of the interaction with the ground.

As a result of this assumption the parameters are determined from the simplest possible experiments, utillizing a classic Newtonmeter and a student to create the force. To gain the resolution and the accuracy of the newton meter, an electrical scale is used as a newtonmeter which measures how many kilo grams of pulling strength it experiences.

To convert into newtons use equation E.1, where S is the strength measured in Kg.

$$F = g * S \tag{E.1}$$

E.3.1 Stiction

By maintaining a position on the floor (requiring a velocity of $V = V_0 = 0$) the stiction force can be found as the largest force which can be applied without creating movement on the foot, see Figure E.6 for reference. The foot has been rigged with three different masses (M) to provide an estimated relation between the normal force and the stiction.

It is observed from the figures in Table E.1 that the stiction is dependent on the direction in which the force is applied, it is observed during the experiment that the toe is rotated slightly

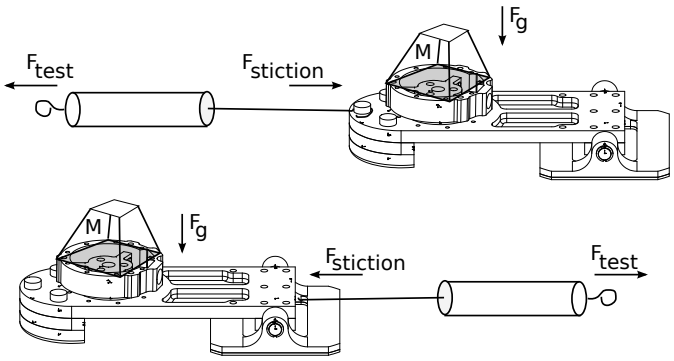


Figure E.6: Test setup for determining the Stiction parameter, the test is run in both directions

Mass [kg]	floor	$S_{heelside}$[Kg]	$S_{toeside}$[Kg]
1.28 (foot only)	roller	1.36	1.00
2.28	roller	2.20	1.82
3.28	roller	3.02	2.64
7.16	roller	5.82	6.20
1.28 (foot only)	linolium	0.86	0.70
2.28	linolium	1.36	1.46
3.28	linolium	2.16	2.10
7.16	linolium	3.68	4.98

Table E.1: Table with test results of the ground stiction test

when the force is applied at the heel side.

It is assumed that this rotation is caused by the friction with the ground, and that the resulting rotation creates and edge between the toe and the ground which for low masses provides an even higher stiction than a flat surface would. At higher masses where the toe cannot be rotated due to the weight of the mass, the effect of applying force at the heel side seems to create a higher stiction than when applying it at the toe side. However on the laboratory floor, this effect seems neglible. It is assumed that this is due to level difference in the toe and the heel height.

The statistic matlab tool "regress()" assumes a model ($y = bx$) where x is input and y is output, using this linear relation regression on the data b was determined, see Table and Figure .

E.3.2 Viscous Friction

By maintaining a constant velocity of $V = C$, the viscous friction force can be found as the force which is applied to maintain the velocity, see Figure E.7 for reference. The foot has been rigged with three different masses (M) to provide an estimated relation between the normal force and the stiction. To ensure constant velocity this experiment is only carried out on the roller, the foot is placed on the rolling band, and the newtonmeter is attached to the frame. This provides a control of the experiment, and it is assumed that the ratio of stiction between the types of floor is also appliccable to the viscous friction.

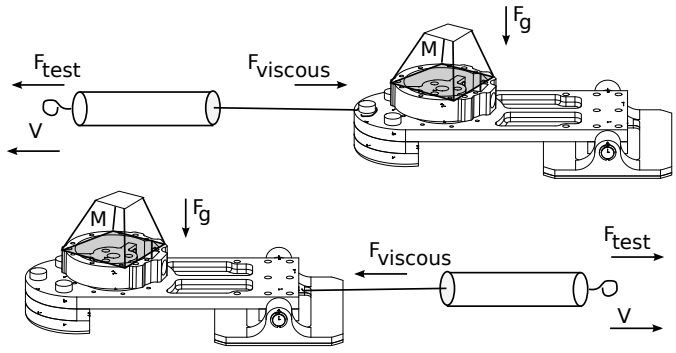


Figure E.7: Test setup for determining the viscous friction parameter.

Mass [kg]	floor	$S_{heelside}$[Kg] (V = 0.14 m/s)	[Kg] (V = 0.36 m/s)	$S_{toeside}$[Kg](V = 0.14 m/s)	[Kg](V = 0
1.28 (foot only)	roller	1.48 - 1.70	1.50-1.68	1.16 - 1.20	1.28 -
2.28	roller	2.60 - 2.80	2.50 - 3.00	2.06 - 2.16	2.40 -
3.28	roller	3.50 - 3.90	3.56 - 4.00	3.18 - 3.30	3.60 -
7.16	roller	7.50	7.50	7.50	7.74

Table E.2: Table with test results of the ground viscous friction test, notice that some measurements are given as a range, this is due to the dualistic nature of the roller band surface. The lower range is measured at the glued assembly part, the higher is measured outside the assembly part.

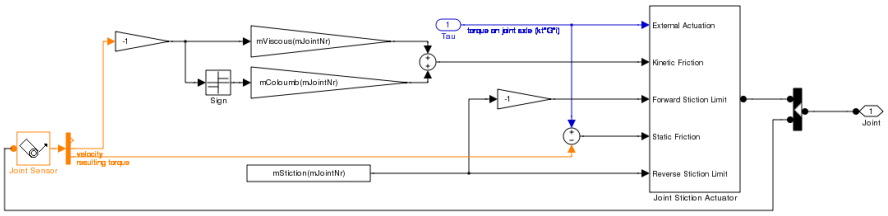


Figure E.8: The friction model used in the SimMechanics simulator.

E.4 Journal Joint Frictions

This appendice describes the joint friction model, and the experiment used for determining the parameters.

The model used in this project is a state based version of the classical friction model, that has a stiction, coloumb and a viscous parameter. The model is build in simulink as shown in Figure E.8. The "Joint Stiction Actuator" is the state machine, which switches between a locked and unlocked joint, depending on the speed and the joint computed torques. If joint torques and speeds are below limits, the joint remains locked.

E.4.1 Test Setup

The model was torso mounted as it was the case when the PI regulator was tuned. See Figure 4.3 in Section 4.2. In the described rig, the torso can be assumed to be rigidly connected with ground.

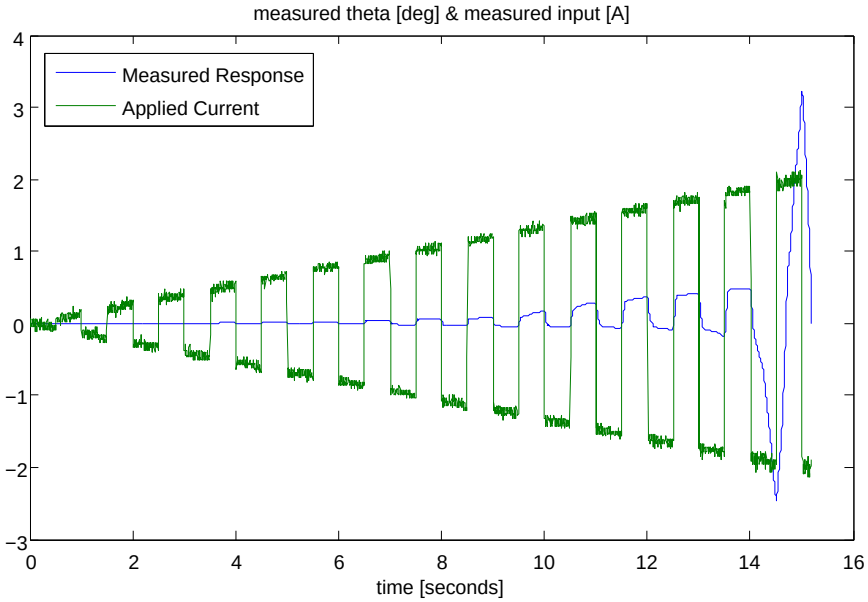


Figure E.9: The signal applied to the actuator by the EPOS, and the measured response for the right knee. A similar signal was sed on all the joints, but the responses varies. Notice how the joint notoriously gets stuck at certain angles, if the speed is too low. This effect is due to the non-linear friction in the HDG, and is hard to replicate with the simple model chosen.

Each joint was moved seperated in time, by applying a zig-zag signal with a very low amplitude, this ensures that the effect from gravity, and thus the dynamics not due to friction are kept very small. The result is an experiment which is designed for stiction and coloumb friction estimation, but provides only a small amount of knowledge concerning viscous friction.

The signal applied is shown in Figure E.9, together with an example response. This is taken from the right knee, as it is one of the joints that most clearly shows that the actual gears have more complex dynamics than a simple classic friction model. When this is mentioned, parameters can be found which approximates the joint behaviour, well enough for the posture controller to be tuned against.

The individual joints was dislinked from the simulator model, and welded to the simulator ground. This produces a compound pendulum model in SimMechanics, with the estimated mass and inertia matrice.

The toolbox Senstools is used to parameter estimate, but as that tool is designed for continous models, and uses a gauss-newton search algorithm, it is not well suited for determining parameters such as a state based stiction. The tool is in this experiments only usefull if the initial guesses are chosen sufficiently well, and with a distance to the discontinuities inflicted by the state based model.

Also the cost function used by the Senstools is a standard mean of the norm function. While this is proper for many functions, it means that it for this problem has a tendency to minimise the amplitude by setting the viscousity very high. This is unfortunate, and gives bad models.

Usually it is expected that a stiction parameters is either the parameter which is added to

the coloumb friction to provide a spike, but as the implementation of the model shows, the coloumb and the stiction parameters are here provided in [Nm]. For many joints they are even counter intuitive, with a stiction that is lower than the coloumb friction. This behaviour is due to the state nature of the model, as this configuration of parameters can produce some of the rectangular spikes seen in the measurements. Normal configurations of parameters is also found, but they have less tendency to prodce the recktangular spikes and are thus worse at fitting the data using this model.

E.4.2 Results

The results where not as promising as was hoped. This is a consequence of a model which is too simple to accurately predict the non-linear HDG frictions, and that a better optimizing algortihm could have been chosen than Senstools. It is suggested that future groups attempt a genetic algorithm, and a friction model that is nonlinearly dependent on speed, and angles.

LeftFrictionColoumb = 12.6226[Nm]
LeftFrictionViscous =3.4904[*]
LeftFrictionStiction =10.7889[Nm]

RightFrictionColoumb =13.7594[Nm]
RightFrictionViscous =9.4260[*]
RightFrictionStiction =10.7889[Nm]

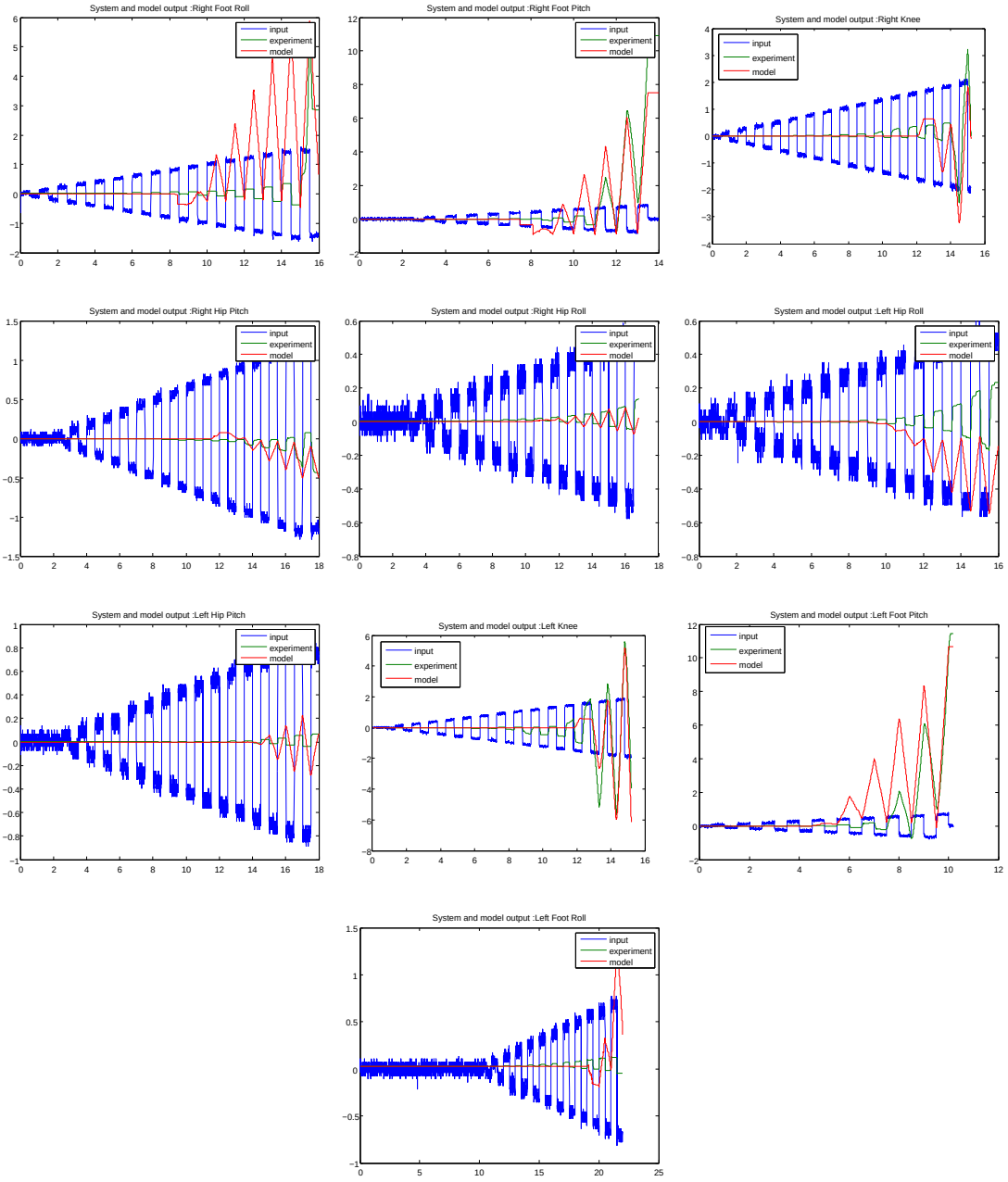


Figure E.10: Comparisons between the models performance, and the actual measurements. The model does not encompas the joint friction very well at theese low speeds. No yaw joints, or arms where actually measured, as they are not used much, and are assmed to have low friction.

Joint	Positive Limit Reason	Negative Limit Reason
Right Foot Roll	Foot bracket, HDG	Fts, Potmeter
Right Foot Pitch	Foot bracket, shin	Potmeter, shin
Right Knee	Epos,Epos	Internal potmeter
Right Hip Pitch	thigh,HDG	Potmeter,bracket
Right Hip Roll	Potmeter,Pelvis*	HDG,bracket
Right Hip Yaw	Motor,Pelvis	Belt,Belt
Left Hip Yaw	Belt,Belt	Motor,Pelvis
Left Hip Roll	HDG,bracket	Potmeter,Pelvis*
Left Hip Pitch	thigh,HDG	Potmeter,bracket
Left Knee	Epos,Epos	Internal potmeter
Left Foot Pitch	Foot bracket, shin	Potmeter, shin
Left Foot Roll	Fts, Potmeter	Foot bracket, HDG
Pelvis Yaw	Motor,bracket	Spinal Coord extension
Pelvis Pitch	Lack of Manpower	Lack of Manpower
Pelvis Roll	Lack of Manpower	bracket,torso
Right Arm	Lack of patience	Lack of patience
Left Arm	Lack of patience	Lack of patience

Table E.3: Reasons for joint limits.

E.5 Journal Joint Min/Max

In this appendice the limits of the joints are presented. The limits was found through experiments, where the EPOS was put in read only state. The point where either physical limitations, or sensory information made it impossible to attain further movements, where recorded by the EPOS, and then analyzed.

E.5.1 Method

The EPOS was allowed to read sensory information, but not apply currents. The robot was then maniplated manually to obtain the limitations of the joints.

E.5.2 Results

There are multiple reasons for the joint limits determined. They vary from sensory data, to physical constraints when aluminum hits aluminium. The reasons for the limits are shown in Table E.3. The limits determined are all relative to zero position and are provided in Table E.4.

Joint	Positive Limit	Negative Limit	Range	Orientation
Right Foot Roll	12.5263	-46.5789	59.1053	-1
Right Foot Pitch	34.5395	-23.7281	58.2675	-1
Right Knee	74.8684	-9.1053	83.9737	1
Right Hip Pitch	17.6023	-45.1901	62.7924	-1
Right Hip Roll	26.8787	-16.3918	43.2705	1
Right Hip Yaw	147.6316	-27.9605	175.5921	-1
Left Hip Yaw	29.1301	-146.5789	175.7091	-1
Left Hip Roll	15.5146	-28.5161	44.0307	1
Left Hip Pitch	17.2880	-42.3246	59.6126	1
Left Knee	73.2456	-17.3026	90.5482	-1
Left Foot Pitch	36.7763	-26.7763	63.5526	1
Left Foot Roll	33.6184	-11.3655	44.9839	-1
Pelvis Yaw	59.6930	-82.2515	141.9444	-1
Pelvis Pitch	24.2763	-35.8333	60.1096	-1
Pelvis Roll	31.1184	-22.1857	53.3041	1
Right Arm	∞	$-\infty$	∞	1
Left Arm	∞	$-\infty$	∞	-1

Table E.4: The determined limits in degrees.

E.6 Journal Mass and Breakaway Friction Estimation Measurements

This appendice presents

F

Balance Schemes

To avoid damaging the robot, and to prolong the timespan of a walk or a standstill for as long as possible, it is necessesary to avoid falling over. Retaining balance is not a new field of research to robotics, and it has been shown multiple times that the ZMP alone provides a usefull reference for a balance controller under initial conditions which are statically stable.

Even when this is not the case, humans can often recover balance in some manner, i.e. by moving the feet to new positions. Hence it should be possible to implement a system for a robot which does a similar thing. To provide an insight in to the schemes considered in this project a few concepts for balancing is presented in this section.

In general for small disturbances, simply shifting the CoP will change the tangential GRF, which again directly affects the motion of the CoM. This is a governing principle, which is used in many existing balance control strategies. However as the location of the CoP is limited to be under the feet, a second strategy that can be applied in addition, is to create a moment about the CoM, creating a momentarily larger tangential GRF. It is when the sum of theese two strategies fail to regain balance, that the robot will fall if it does not shift the location of the feet.

It is the balance controllers task to detect when to do what, and to ensure that the balance is stabilized.

F.1 GCoM Balancing

The GCoM is only relating to the kinematics and mass distribution of the robot, and is independent of the current velocity or accellerations. Hence it is interesting to look at the GCoM when the robot should remain in a pose for a longer time. The logic is quite simple, if the GCoM is not within the PoS the robot cannot remain in the position without generating a torque about the point of ground contact.

As the contact with the ground is not infinitely strong, the torques may be required to arise from a flywheel effect, i.e. accellerating the arms, and determining how much torque is needed from the GCoM to sustain a pose is difficult.

The GCoM can however be used for systems with large PoS as a very simple balance control, if very slow movements are applied and the effects of the dynamic properties are kept very small. While this is usually only interesting for static systems as buildings, or near static systems as hexapod robots walking in tripedal gaits, it does in fact also have some relevance in double support.

- ▷ The robot may only be shut off if the GCoM and the dynamic points as ZMP are all within the PoS.
- ▷ If the robot is nearly static (moving very slowly) it is sufficient to ensure that the GCoM is within the PoS, and balancing can be handled kinematically.

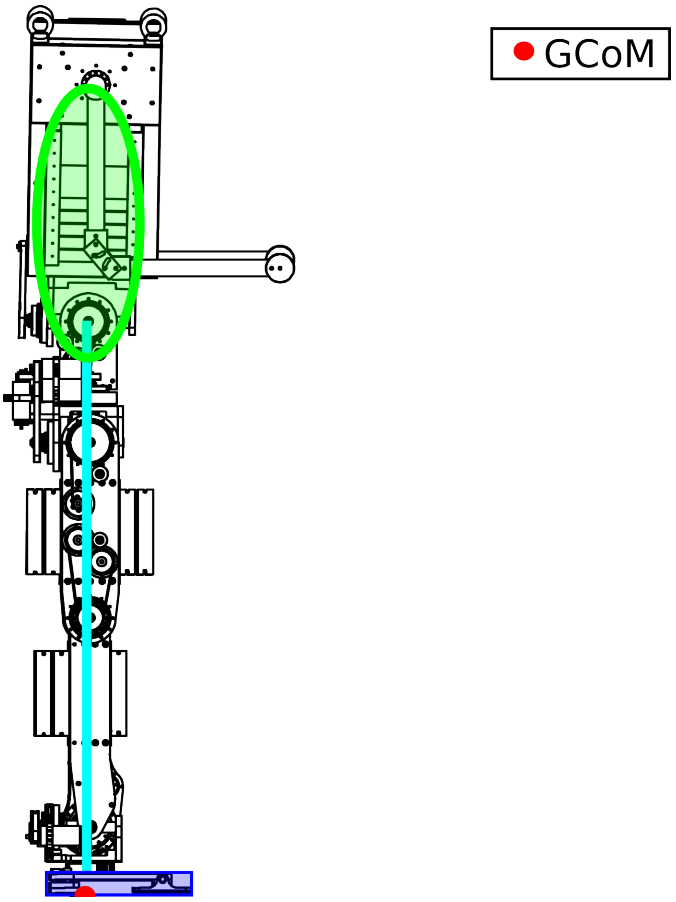


Figure F.1: GCoM is directly beneath the CoM, and when the robot is not moving, it is necessesary to ensure that GCoM is within the PoS. The PoS is the area beneath the foot (blue).

F.2 ZMP / CoP Balancing

The ZMP and the CoP for that matter, is located at the same geometric point while the robot is statically stable. Hence the double name convention for this scheme, it is not two seperate points which are used!

The concept is quite simple in its overall structure:

- ▷ The desired location of the ZMP is calculated or chosen, usually in an offline situation, which is within the PoS, and has a margin to the edges.
  For standing upright this is often configured as a zero reference for the robot.
- ▷ Some joints are chosen to react on the error of the current ZMP location (using feedback and a linear controller).

Often a small torque is provided at the ankles as the feet torques relate directly to the location of the CoP, and thus is able to manipulate it.

To enforce the balance scheme the torso is sometimes chosen to rotate along to create larger GRF. The reason for the popular choice of the torso is that it can be somewhat dynamically

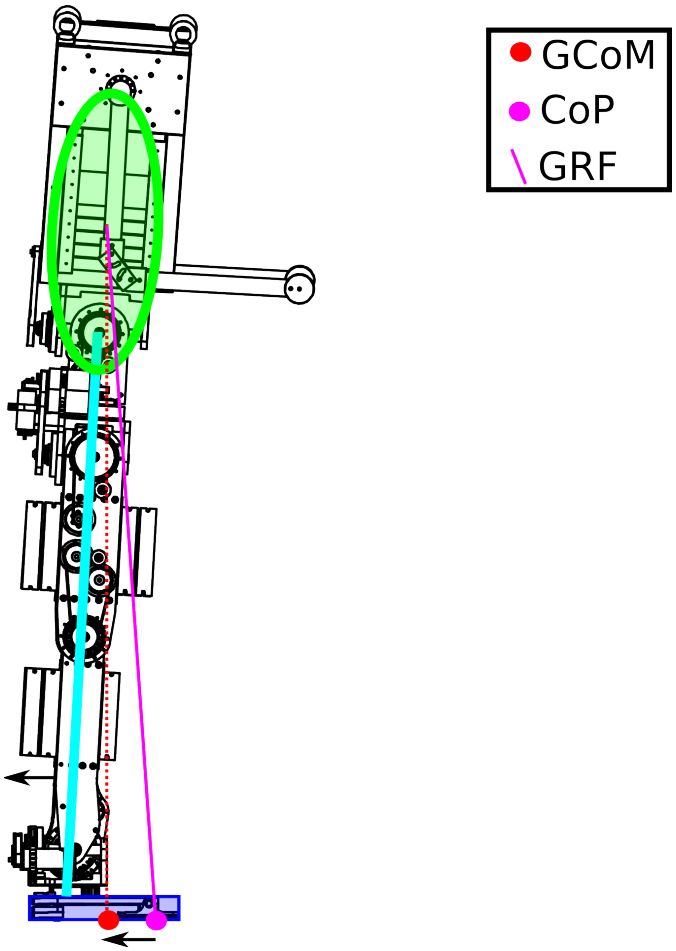


Figure F.2: The CoP is within the PoS, and it is expected that balance can be recovered by adding torque at the ankles. The direction from the GCoM to the CoP indicates the direction of the torque. In the figure this is oppositely drawn, as the arrows here indicate the desired motion in relation to the ground. The GRF is the ground reaction force, which is determined in the CoP.

decoupled from the lower limbs, the disadvantage is that by using this enforcement strategy the moments generated are not continous, and is larger than the ones directly induced at the ankle, and can cause instabillity.

The result of ZMP control is that while the ZMP is kept within the PoS the robot remains in a statically stable configuration, and hence in balance. It is in concept somewhat comparable to balancing an inverted pendulum, as torques are applied directly at the ankle, and it is often assumed that foot to ground contact is preserved.

It requires a well determined CoP or ZMP, and as CoP is obtainable from measurement data, the strategy is often used for robots which have multiple FTS mounted in the feet. It can be difficult to obtain this if passive joints are placed between the ground and the FTS.

If computational power is available, the iZMP have been succesfully used to stabilize a dynamic walk, as it can be determined even if the ZMP leaves the PoS.

F.3 LIPM and AMPM Balance Control

When in need of fast and powerfull balance control, where the CoP or ZMP can difficult to obtain directly, the presented method based on theese points can with some advantage be exchanged with an observer based controller. Considering the LIPM as an observer allows a control system to do this.

The assumptions used for the LIPM is that the foot remains in the same position during the span of the ground contact, and that the angle in relation to the ground can somehow be obtained. The observer model is, as suggested by the name, a linearised 1DoF model of the robot. With the single DoF located at the ankle. Thus it can require the robot to remain close to the kinematic refence in which it was linearised.

The model is often dubbed "cart-on-a-table" as it is close to this well known control problem in conception.

The observer is used to estimate the ZMP location, or simply the movements of CoM and thus the feedback can be provided to apply torques on the ankles.

If computational power is available, the Angular Momentum Pendulum Model (AMPM) is a more powerfull model for the observer, as it in addition to torque at the ankle allows a mass to be rotated about the CoM of the linearised pendulum. This allows the controller to rotate one or more joints on the robot to create torque. Often the pelvis is chosen as rotates the torso, and often the observer would be even better if the free leg is added also to weigh the CoM of the rotated objects close to the linearised models CoM.

F.4 CMP Balance Control

The CMP is much like the CoP in the manner that it can be determined from measuring ground reaction forces. The advantage of this is that it is obtainable quite fast, and that it can be fed directly back into a controller. The point is not restricted to the PoS and is thus suitable for dynamic walking, given that there are no passive joints between the FTS and the robot.

A CMP control strategy is much like the ZMP control based on a reference point, and an error, and a linear controller, and thus enherits the advantages of a simple structure, but does not suffer from the limitations of the location of the ZMP point. However where the ZMP control can be primarily considered an ankle strategy, the CMP control relates more to the ground torques and forces created by using the free parts of the robot body as a flywheel mounted at the hip.

The CMP balance controller is thus best suited for larger corrections, and can be hard to stabilise at small errors. Hence creating the torque which recovers balance becomes a matter of accellerating the masses not currently used for support.

F.5 Change of Support Balancing

A simple way of regaining and ensuring balance is to simply ignore that standing still is energy conserving. If this is not important, balancing can be considered a matter of ensuring that the upper body is allways pointing upwards.

For a robot with legs, this strategy alone can result in a "pogo stick" behavior of allways shifting balance between the legs, and thus rarely stands still. Calculating where and when to place the feet to ensure a fast stabilization, can be a challenge.

As energy is allways lost at impacts, a simple (but slow) way to ensure that balance eventually will be regained (assuming no walls appears in the path) is simply to determine the current angle of the stance leg (in the direction of the fall), and then double it and apply it as an interleg angle.

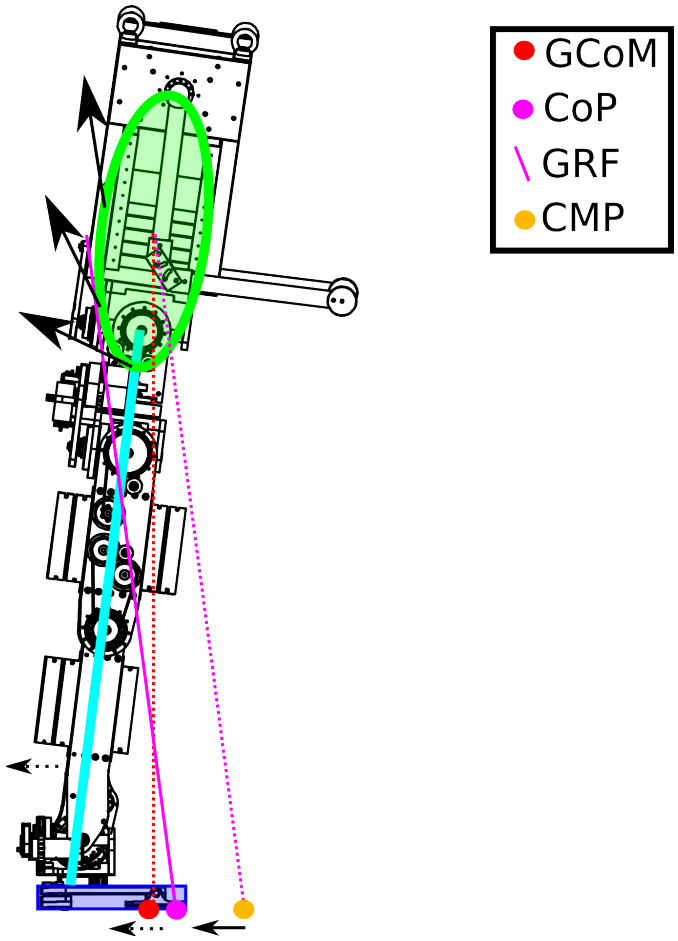


Figure F.3: The CoP is at the edge of the PoS, and when determining the GRF it is not going through the CoM of the robot. However if the GRF vector is extended from the CoM the CMP point is located on the ground. The relationship between the GCoM and the CMP provides information about how to accellerate the free masses, to generate the desired torques, and recover balance.

If the interleg angle becomes larger than the double stance leg angle, the robot stops it's falling using fewer steps, and as a result extends the PoS, and while this is preferable for standing it might not be preferrable in terms of energy consumption, as a stance which deviates much from the zero position of the joints requires more torque and thus more power to sustain.

If the interleg angle is smaller the robot may have to use more steps than otherwise, and ultimately if very small, the balance might not be recovered at all.

G

Control of a 2d Straight Legged Biped

To enable the 2d Straight Legged Biped Passive Walker to walk on a level or upwards sloping ramp, it is necessary to add extra torques to the walker. The basic model of the passive walker derived in Appendice B.3 is able to walk down a ramp, if the slope and the initial conditions are tuned to the weight distribution.

The dominating energy loss which degrades walking, is the loss of angular velocity at heel strike, this cannot be directly removed by a controller, but fortunately this is not necessary, as the introduction of a slope angle allows the passive walker model to rebuild the lost energy within the step.

However with a slope angle at zero it is necessary to add torques about the ground pivot point to ensure that the stance leg swings at a rate which allows continued walking.

Feet allows the addition of torques to the ankle, as rotations of the feet is equivalent to moving the CoP away from the pivot point in the model, there is offcourse an upper limit, as the CoP may not leave the PoS, but assming large feet, this can be ignored. To provide a simple realisation of this concept, massless feet are added in simulation, this have the effect that as long as the feet remains on the ground with infinite friction during stance, the model derived in B.3is still applicable.

If rotation about the angle on the stance leg, is affected by a controller it will according to Euation B.25 also effect the swing leg. This can potentially effect the stride length, and thus create a situation where more energy is needed to sustain the gait than would be needed for the optimal gait. Hence a controller needs to compensate for the slope of the ramp to ensure that the interleg angle and stance leg angular rate both ends up close to the situation which occur for the passive walker on the optimal slope.

It is usually a good idea to apply the simplest possible controller first, and then only when this cannot be done succesful expand the design, as such a strategy lowers developement time. The PID controller is a well known linear controller, which in relation to implementation is quite simple, it is also a SISO controller, which infers the need for a SISO control strategy.

A strategy could be to choose the stance leg angle reference as a function of the slope angle, and then regulate on the ankle joint only. Such a strategy is proposed in Figure G.1.

Tuning the PID controller to handle this is not straight forward. However, in this case where the primary goal is to regain the lost angular velocity, and a reference angle is chosen to be close to the final angle, the proportional gain alone should be able to recreate the rise time needed to sustain the gait.

An example is used for setting up a structure, with the parameters hipmass M=1, leglength=1, footmass=0, γ_0=0.009 radians.

It can be seen from Figure G.2 (a) that the example passive dynamics ramp walking biped, turns the stance leg about the ground pivot point, at a rate close to an inverted pendulum. From Figure G.2 (b) it is seen that the excact same walker initiated with the excact same conditions

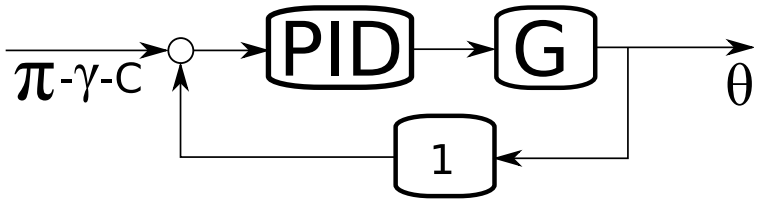


Figure G.1: A simple strategy for controlling the biped during the swing phase. As $\frac{1}{2}\pi - \gamma - c$ is a function of the slope angle, the controller reacts different for different slopes.

cannot walk on a level surface, in fact the stance leg does not even pass vertical.

If controlled by a proportional controller and if the reference angle is chosen to be 0.2 radians, (close to the termination angle of the stance leg at the tuned passive walk), the controller will add more torque in the beginnning than in the end of the swing phase. It becomes possible by using a small gain, to cross zero at a reasonable time, and from there on the inverse pendulum dynamics will ensure that the biped continues it's step, if not disturbed by the controller.

It is clear that the controller will be actuating very little, as the dynamics of the closed loop system should resemble the uncontrolled plant behaviour at the optimal slope angle. Optimally it should have a controller gain at zero at the optimal slope angle. This somewhat relates to moving the working point of a linear model.

As a solution of the two Ordinary Differential Equation (ODE)s provides little new knowledge about the system, and solving them is a very time consuming task, a P controller gain is found for a stable walk on a level surface using a simple binary search algorithm.

Stable walks can be achieved on level surfaces for the example system with a range of gain values. However within theese values, the closed loop system is unable to walk at the original slope, where the passive dynamics walker alone could walk down the ramp.

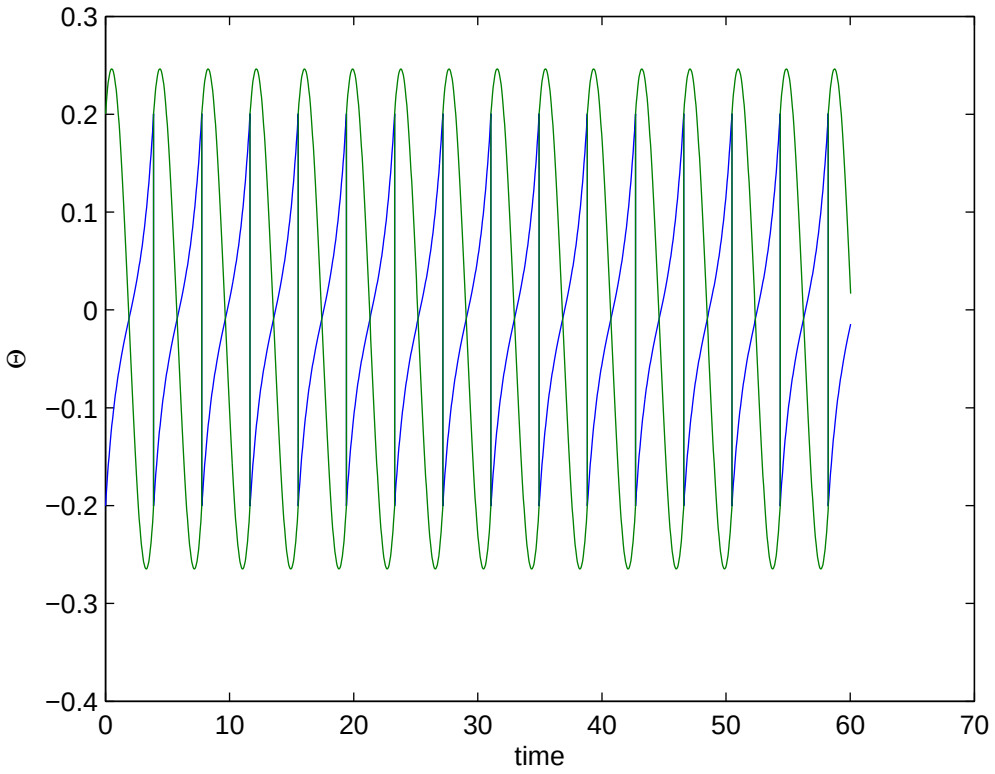
As this seems like an unfortunate property of a controller, and the effect of setting the reference as a function of the ramp slope, has little effect in this example, a different approach to including the slope angle is considered.

It is attempted to let the controllers parameters be determined from the slope angle. A naive proposal for such a parameter selection could be to choose $P = P_{level} - (P_{level}/\gamma_0) \cdot \gamma$, as this would ensure a gain equal to zero at the original ramp, and equal to the found stable P_{level} for walking at a levelled ramp.

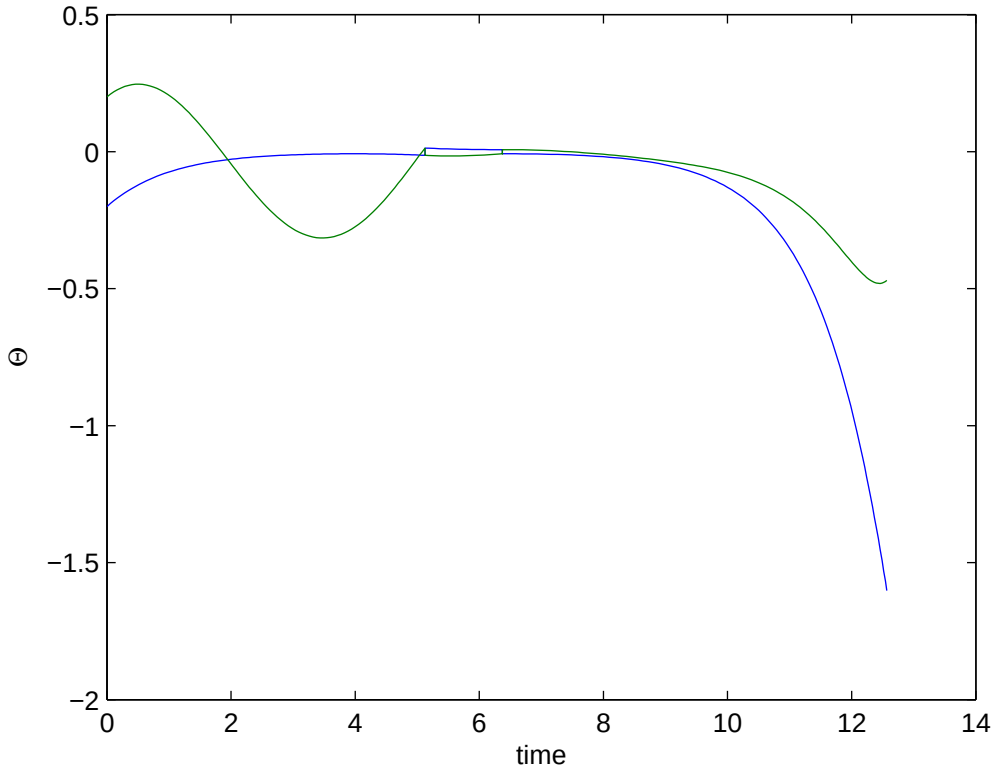
This does in fact, provide a margin of stable walks in the test example ranging from -0.011 radians to +0.001 radians.

This example shows that it is possible to achieve stable controlled walk at a level surface, without changing the original system properties using a modified version of a proportional controller. The resulting walk is shown in Figure G.3.

It should be noted though that the controlled biped can walk up ramps steeper than +0.001, although the walker then slowly converges towards a stride length of zero, and eventually falls. It is also noteworthy that the assumption of very large feet may not in fact be applicable in real life, as the CoP may be forced out to the rim of the foot, making the effect at best useless.



a



b

Figure G.2: The angle of the stance leg (blue) in relation to vertical, and the angle between the legs (green) during the swing phase of an uncontrolled biped. (a) shows the stabile ramp walk, and (b) shows the unstable level surface walk.

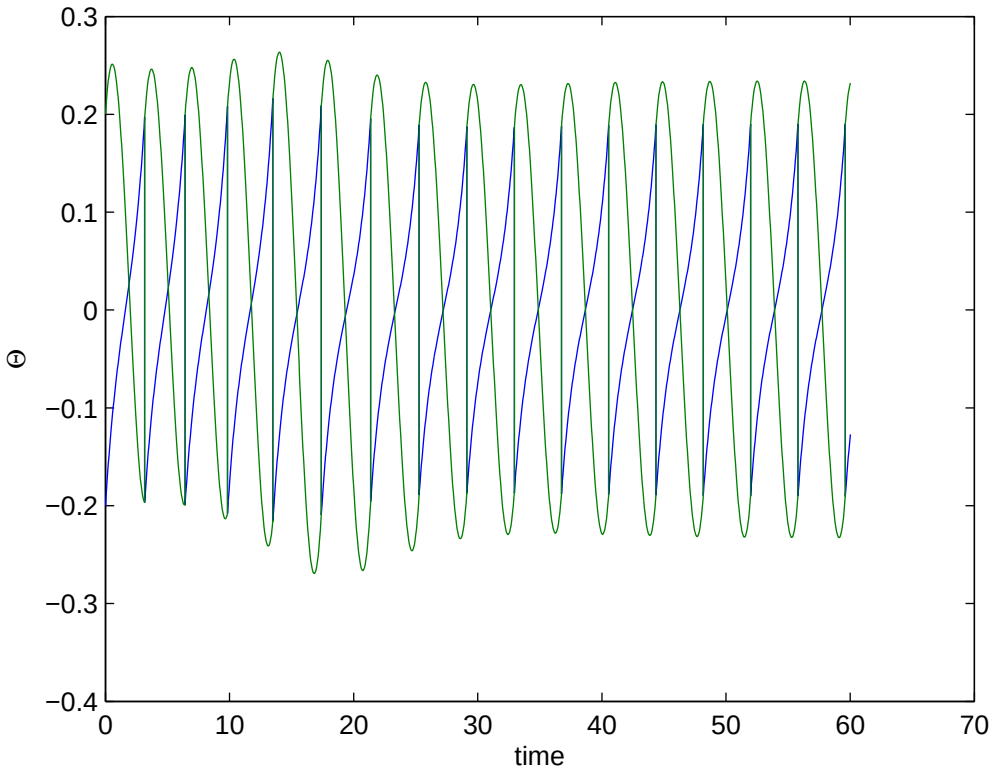


Figure G.3: The angle of the stance leg (blue) and the angle between the legs (green) are shown, as regulated by the Proportional controller at a level surface. Notice the settling time before the gait is stable.

H

Application Note for Hammer Industries

This document illustrates how the AAU-BOT1 weapons platform can be fitted with the Hammer Ind. CPS 2000 MK2 HB Assualt Rifle[1].

1.1 AAU-BOT1 Platform Basics

The AAU-BOT1 weapons platform®, is a versatile stabile all terrain vehicle for deployment of anti-infantry equipment. The AAU-BOT1 features bipedal walk, and can be camouflaged using an ordinary military uniform size XXL. It is completely autonomous and determines the path from base to deployment site.

The AAU-BOT series comes with a variety of configurations, from remote sniper configuration, the remote drone configuration, and the fully autonomous hunter killer mode [2]. The Configuration options are listed in Table_H.1.

Setting	Jumper Setting	Reboot Required
Remote Sniper	1000	
Remote Drone	0100	*
Stealth Mode	0010	
Paramedic	0001	*
Hunter Killer	0000	

Table H.1: Configuration Settings for the Jumpers. Jumpers are located within the OBC Casing to avoid reconfiguration during operations.

1.1.1 Instrumentation and Interfacing

The AAU-BOT1 is equipped wit advanced sensors, that allows it to orient it self in unknown locations. It can determine its heading by means of compass, and GPS. It can compensate for accellerations and jerks, and uses sensor fusion to estimate the wind speed. The estimation of wind speed is not used for orientation, or locomotion, but only serves as raw data for the targeting system.

Seismic equipment in the feet allows AAU-BOT1 to detect the approach of heavy battlefield equipment, and it will either camouflage, or take cover, if it estimates the apporaching vehicle as a threat.

[1] This Appendix contains sarcasm, and is placed in the thesis as an easter egg. It is a noteworthy bad example of Danish humour.

[2] It should be noted that the Hunter Killer mode is not recommended in situations where friendly fire should be avoided.

Digital video and audio recording equipment can be installed easily by connecting it to one of the available OBC USB ports. The AAU-BOT1 Firmware will detect the sensory equipment, and place it into the sensor array for better navigation and targeting. Audio recording equipment should as a minimum be in the 40~16kHz, with a SNR of at least 7dB. If cameras are connected, be aware that better precision can be achieved by removing the infared filters[3].

1.1.2 Coorporative Advantages

Multiple AAU-BOT1 can be configured to coorporate on joint operations. The internal communcation between the robots is ensured by a short range and a long range secured wireless network. A base system software can be configured and installed on standard x86 architectures, which facillitates a possible Master / Slave configuration.

If AAU-BOT1 looses contact with the base, it will automatically go into Autonomous Mode, and attempt to return to base. The base system can require multiple cores to analyze battlefield data, and if computer power is available it can be set up to take strategic counter measures to any potential threat. To ensure data integrity, and deployabillity the software is preconfigured to gain access to all available computer power within network range, that is compatible with the x86 architecture.

1.2 Installation of CPS 2000 MK2

While AAU-BOT1 is not equipped with hands[4], weapons can be mounted directly on the arms.

The 3D joints in the Waist is automatically combined with the shoulder joint to provide a stable platform for weapons firing. AAU-BOT1 detects the existence of weapons and runs a self test on the targeting system when a new weapon is detected.

CPS 2000 MK2 is a low weight rifle, that is mountable directly with the examplary brackets that is delivered with AAU-BOT1. It is mounted in the following order:

1. Insert Bolt A, in slots 1 and 2.
2. Insert Bolt B, in slots 3 and 4.
3. Mount bracket on each side of arm, and tighten bolts A, and B, with approximately 23 [Nm]
4. Attach clamp on hand cylinder.
5. The bracket contains internal electronics that will inform AAU-BOT1 of the presence of a weapon.
6. Should this not happen, installing video equipment that allows AAU-BOT1 to vizually identify the weapon, can solve the problem.

AAU-BOT1 should now take some test shots, to access the weapons power, and calibrate the targeting system to the available sensors. Upon succesful completion AAU-BOT1 is ready to deploy in operations.

[3] For night vision or simply better color filtering, it is recommended to use the AAU-BOT1 red backlight camera sockets.

[4] AAU-BOT2 is expected to have hands.

Palantir Protocol

This appendix describes the the protocol for visualization in sufficient detail that a new client or server could be written from this.

I.1 Design Requirement

- ▷ Minimal overhead on the OBC
- ▷ Flexible
- ▷ Extendable

I.2 Basic Structure

The Palantir protocol uses a simple stream based protocol where communication only goes from the client to the server, and as a non-critical system any faults can be ignored.

To transmit the stream both UDP or TCP can be used. Both requires a port number, and from the recommendations from IANA a private port number, 61001, is chosen as default.

The main process in the protocol is first to use a bit of time to setup the protocol and afterwards stream the raw data efficiently.

Since it would be undesireable to wait for the entire stream to be transmitted, seperation into logical work units is needed. Mainly two methods exists for such a problem. One solution is to seperate the data with an indentifier like newline, which gives the advantages, that it is easy to seperate and easy to find the start of a logical unit from anywhere in a stream. The disadvantages is that your data either cannot contain the seperator or you have to escape the seperator and escape the escape value which leads to pre- and postprocessing of the data.

The other method is lenght prefixing, in where each logical unit is prefixed with the lenght of the data. This makes it hard to find the next logical unit without knowledge of the previous, but any data can be included without processing it.

The lenght prefixing method is chosen to minimize the overhead on the on board computer, which then doesn't have to care about the bit representation of the data.

To add flexibility each data block is also prefixed with an identifier describing what kind of data is in the data block. In case the receiver does not regonize the identifier it can skip the entire block and resume reading from the next logical block.

As common in network protocols data lenght is in most significant byte order.

Byte Offset	0	1	2	3	4	5	6	7	8	9	...
0	Identifier				datalenght				data		

This structure is called a Palantir command and data represention is futher defined for each identifier in section I.4.

I.3 Palantir references

Instead of writing which variables that gets updated in each timestep, or hardcoding the order, the Palantir protocol uses references defined earlier in the stream. There exists two kind of references in the system.

There is references to objects, which represent a reference to a named object in the 3d visualization in a flat namespace, and is a string of any lenght.

List references is used to reference a previously defined order of the parameters and must be a string of excatly 4 chars.

I.4 Palantir Specific Commands

I.4.1 Load model

Loads a model file into the visulization. Any named object inside the model file should be available to use as an object reference. In this example the file "aaubot.bam" is loaded, and in this several hierarchical objects is defined in forward kinimatic manner. For the following examples it is assumed that this file contains objects named "Torso" and "LeftArm".

Byte Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	'L'	'M'	'D'	'L'	0	0	0	11	'A'	'a'	'u'	'b'	'o'	't'	'1'	'.'
16	'b'	'a'	'm'													

I.4.2 Make Center Of Mass Object

This creates a single object that is the representation of the center of mass. The data contains the object name that should be set on the newly created object so it can be referenced later.

This examples creates a Center of Mass Object with the name "Name1"

Byte Offset	0	1	2	3	4	5	6	7	8	9	10	11	12
0	'M'	'C'	'O'	'M'	0	0	0	5	'N'	'a'	'm'	'e'	'1'

I.4.3 Create update list

The data in this package is the name of the list followed by space delimited string of pairs of value types and object tags.

In case the object tag contains spaces Unix style escaping should be used. Binary64 is defined in the IEEE 754 standard, and C like languages it is commonly known as double.

Value Type	Meaning	Value data structure
Pos	Posistion of object	3 binary64 (x,y,z)
Quat	Rotation of object as a quaternion	4 binary64 (qw,qx,qy,qz)
PosQuat	Combination of the two above	7 binary64 (x,y,z,qw,qx,qy,qz)
H(Heading)	Yaw of the object	1 binary64
P	Pitch of the object	1 binary64
R	Roll of the object	1 binary64

This example creates a list named "LST1" with can update the position and rotation of the Torso as well as the pitch of the arm. Total lenght is 28 which is 0.0.1.11 in dotted decimal or 0x1B in hex.

Byte Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	'C'	'L'	'S'	'T'	0	0	1	11	'L'	'S'	'T'	'1'	'P'	'o'	's'	'Q'
16	'u'	'a'	't'	' '	'T'	'o'	'r'	's'	'o'	' '	'P'	' '	'L'	'e'	'f'	't'
32	'a'	'r'	'm'													

I.4.4 Update list

This command is used for adding a new keyframe in the animation of the object in the visualization. It consists of a list name, optinal zero padding, a timestamp and finally the data. The timestamps must be monotonically increasing and is measured in seconds. The padding is made for easy integration with matlab for which each sample can be seen as vector of doubles in which the two first is constant.

In this example the list from before is updated. The packet has 4 byte padding after the name to align the bytes.

After 0.2 seconds the torso is moved 3units forward,5units up, 7units left and is rotated 90degrees about the y axis.

The "LeftArm" is rotated 90degrees in the opposite direction

Byte Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	'U'	'L'	'S'	'T'	0	0	6	8	'L'	'S'	'T'	'1'	0	0	0	0
16	0.2e0								3.0e0							
32	5.0e0								7.0e0							
52	0.7071e0								0.0e0							
64	0.7071e0								0.0e0							
80	0.7071e0								0.0e0							
96	-0.7071e0								0.0e0							

J

Bug Reports

This appendice contains bugs experienced while working in the laboratory, the origin of the bug when found, and the result of the debugging. The document is created to minimize the work concerning future debugging, if a bug re-occurs.

The formatting of a bug is

- Title shortly describing the bug
  A description of the bug symptoms
- Origin
  Is a title which must allways be present, even when the origin of the bug is unknown. A short and precise summation of propable the origin is allways appended, if it is found.
  - Analysis
    Steps taken to identify the origin
  - Conclusion
    If the origin of the bug is likely to be found, the conclusion is put here.
- Solution / Workaround
  Contains the steps taken to debug and remove or simply contain the bug, and a note on the experienced succes with the workaround.

J.1 OBC Network Adapter Error

The SSH connection to the OBC is lost and cannot be re established. The OBC does not respond to a ping. And if screen and keyboard is connected to the OBC running

```
>ifconfig -a
```

on the OBC indicates that the network adapter is not present.

J.1.1 Origin

Improper encapsulation of OBC causes the it to invoke a hardware error on the Network Adapter

Analysis

The OBC was rebooted, which suppresed the bug. When the bug reappeared suspescion was put on the hardware, and it was removed from the temporary cardboard casing. This removed the bug, but the bug reappeared after reinserting it into the cardboard box. This was repeated 3 times with similar symptoms.

Conclusion

The cardboard box, have collected dust, or the anti-static padding have attracted material which allows a static charge to build up.

J.1.2 Solution / Workaround

The OBC mainboard was equipped with spacers, and placed on the table without a cabinet.
The bug has not reappeared since the workaround was initialised.

J.2 OBC make cannot locate aaubot_io.h

When building a controller, the process indicates that an include error happens with aaubot_io.h

J.2.1 Origin

The path is hardcoded and the build is not done from the hardcoded path.

Analysis

Location of the file was done using grep, and the presence was confirmed. The path was different from usual build path, which suggested a hardcoded path issue. The path to the aaubot code is defined in rtai.tmf, and when changed the build succeeded.

Conclusion

The hardcoded reference to the aaubot directory containing header files causes the build to fail.

J.2.2 Solution / Workaround

Change the hardcoded path in rtai.tmf to reflect where the build can locate the files.

```
AAUBOT_DIR = /pathdir/path
```

J.3 OBC is irradic after Re-assembly.

When OBC has been disassembled, or moved, it chrases when the aaubot module is modprobed. SSH connections are at best random.

Analysis

The bug was found after an attempt to fit the OBC into the new cover. This required the I/O interfaces to be shifted. The drivers seem to autodetect the location of the cards but at the initialisation of the aaubot module interacts with the I/O interfaces. It was still suspected of raising the bug.

Shifting the I/O interface cards back resolved the bug.

Conclusion

The I/O modules is not responding properly.

J.3.1 Solution / Workaround

Check the I/O boards, and check the sockets for foreign objects. Also verify that the CAN I/O interface card is be placed in the PCI socket closest to the CPU. Ensure that the FTS and IMU interface card is placed in the PCI socket furthest away from the CPU.

J.4 Permission Denied during compile

When running aaubot_engage, the compiler on the local machine reports permission denied, and other errors.

Analysis

The rtw build procedure tries to engage the builded code on the loca machine, or has the wrong realtime target configured.

Conclusion

The simlink model is not configured for the build procedure.

J.4.1 Solution / Workaround

open simulink model configurations, choose the:

- ▷ "language" as "c"
- ▷ "system target file" as "rtai.tlc"
- ▷ and enable "generate code only"

Now that model should be compileable with aaubot_engage.

J.5 initialization commands cannot be evaluated during compile

When running aaubot_engage, the MATLAB build cannot evaluate some of the initialization commands

Analysis

No analysis was carried out, this bug was simply observed.

Conclusion

simlink has known bugs

J.5.1 Solution / Workaround

remove break points

J.6 Segmentation Fault During Compile

When running aaubot_engage, the compiler on the OBC reports segmentation fault.

Analysis

The only change made to the system was the placement of a "to workspace" block in the simulink model. This was removed again, and it solved the problem.

Conclusion

The simulink model must not contain a "to workspace" block.

J.6.0.1 Solution / Workaround

Remove all "to workspace" blocks from the simulink model.

J.7 OBC Aborts initial position, and hangs during compile

When running aaubot_engage, the compiler on the OBC never completes, instead a red warning light appears.

Analysis

When initial conditions are chosen such that the physical joint limits are limits, the EPOS will shut off during movement to initial conditions. This happens as the last part of the compile, but the software cannot handle this error, and shuts down.

Conclusion

reference initial and zero positions must be legal for the compile procedure to complete.

J.7.0.2 Solution / Workaround

Set initial conditions within joint limits.

J.7.1 Weird errors during compile

When running aaubot_engage, the compiler on the OBC reports permission denied.

Analysis

This happens sometimes when obccontrol or aaubot_control is runnning on the OBC, the reason for this is not isolated, but can be related to rtai. It is expected that other rtai dependent applications can provoke the same bug

Conclusion

The bug is not critical as it relates to compile time only. This happens rarely, but should be investigated in future projects.

J.7.1.1 Solution / Workaround

reboot OBC, kill all aaubot related applications on the OBC, and rerun aaubot_engage again.

J.7.2 No sensor data recorded with sensor read blocks

When runinng a controller, no data is recieved from EPOS.

Analysis

This was a known bug, from investigating software.

Conclusion

EPOS are initialised with the actuator block. Hence it does not sample data without an initialisation.

J.7.2.1 Solution / Workaround

Add an actator block with no inputs, as this initialises the EPOS, without moving the actuators.

J.7.3 FTS Measurement Errors

While running a controller, the FTS measurements suddenly becomes strange.

Analysis

The bug occured while running an inverse kinematics test, that did not rely on FTS, but used it for state changes. The experiment was aborted as suggested, and at abortion the emergency state was not entered. This was of no importance to safety, but triggered the attention of the team.

The recorded FTS data was investigated, and as seen on Figure .... the data revealed a sudden change in the levels recorded by the FTS amplifiers. The error log on the OBC revealed that the FTS channel 4 had reported an out of sync error. The bug has reappeared with random intervals.

```
aaubot1 kernel: [  505.732032] AAUBOT-I/O(401) - Channel 4 out of sync! (old:0x08
    new:0x02)
```

It was not possible to reengage the FTS communication with channel 4 after the experiment was stopped. A reboot of the OBC solved the problem.

The behaviour of the data has been found in old measurement data also, where the FTS has no been used actively, but nevertheless has been recording during the experiment.

Conclusion

The FTS amplifier drivers are prone to unknown errors.

J.7.3.1 Solution / Workaround

The proper solution is to rewrite the FTS drivers. In the current system no workaround has been found, the error detection seems stable, and now alerts the operator by igniting the red warning light.

J.7.4 Measurements from FTS is off by a factor of ~ -10

When runinng a controller, the FTS data was suddenly very wrong on the FTS on the left foot.

Analysis

This FTS has been reported by previous groups to have been prone to shortcircuits, and other bugs. The wiring was moved about to prevent this problem.

The FTS amplifiers did not reply with an ack, except channel 5. Hence it was not expected to be the analogue part of the FTS that failed. The FTS was soft rebooted, and the FTS was placed in read mode.

The FTS amplifiers was still streaming at OBC reboot.

Conclusion

FTS driver bugs caused the error in FTS data.

J.7.4.1 Solution / Workaround

Shut down everything and reboot all digital systems, including FTS amplifiers.

J.7.5 OBC is not responding after re-assembly

The Cooler for the CPU initiates, but the screen remains blank, and nothing happens even if the power button is reset or pressed down.

Analysis

The lack of activity suggests that the CPU or BIOS is malfunctioning. The CPU has a seperate power chord. This was not plugged in in this case and caused the error.

Conclusion

All power chords must be plugged in to the OBC before turning it on.

J.7.5.1 Solution / Workaround

All power chords must be plugged in to the OBC before turning it on.

J.7.6 Timeout on CAN write 0 and 1

The timeout message is raised when the connection to the EPOS on the CAN is not existing.

Analysis

As the bus id's are defined from 1 and upwards on aau-bot, the error is assumed to indicate that CAN bus id 1 and 2 is not connected or the EPOS are shut off. A quick test of this verified the thesis.

Conclusion

Probably the CAN cables are not connected

J.7.6.1 Solution / Workaround

plug in appropriate CAN cables.

J.7.7 EPOS not Responding.

The EPOS on the front of the right lower leg is inresposive to both configuration attempts and activation codes.

The status indicator LED on the EPOS blinks green.

The slave (inner) EPOS on the back side of the leg is indicating with a red diode that an error has occured.

J.7.7.1 Origin

Faulty CAN cable between the master and the slave on the back side of the right lower leg.

Analysis

The EPOS on the front side of the leg was visually inspected, and no immidiate hardware bugs was visible. The reset routine was run with no errors reported. CAN was disconnected to the leg, which resulted in similar behaviour on the remaining EPOS. CAN was connected directly to "malfunctioning" EPOS on the front of the leg, which allowed configuration and activation. The CAN cable connecting the the front of the lower leg with the back was tested without errors, while reassembling the slave EPOS was found to indicate an error. The CAN was routed around the slave EPOS with the error indication, which reestablished connection to the EPOS on the front. The CAN cables on the slave EPOS was replaced, and communication was reestablished to all EPOS.

Conclusion

Faulty CAN cable between master and slave EPOS on the backside of the lower right leg caused the bug

J.7.7.2 Solution / Workaround

Replaced faulty CAN cable.

The bug has not reappeared since the workaround was initialised.

J.7.8 EPOS allways on

The EPOS on left ankle roll, and .... does not react on the binary error signal from the angular limit circuit.

The binary indication is present, when masured by multimeter. The EPOS continues to measure the angles by means of potentiometer..

J.7.8.1 Origin

Probable cause is the EPOS configuration, or the EPOS socket for the cable.

Analysis

This bug is not investigated yet, as it does not cause problems during nominal operation.

Conclusion

Probable cause is the EPOS configuration

J.7.8.2 Solution / Workaround

TBD.

J.7.9 Screen Flicker on OBC Terminal

The terminal text and windows flickers

J.7.9.1 Origin

unknown

Analysis

The OBC PSU voltages are noisy, this can result in timing errors on the VGA output circuitry. When flickering is worse than usual, it is accompanied by an audio noise generated by the OBC PSU.

Conclusion

Unknown cause

J.7.9.2 Solution / Workaround

symptoms ignored

J.7.10 Left Knee Current Mode Move Error

When in current mode (as in controlled mode) the left knee only moves slightly and then stops.

J.7.10.1 Origin

Primary EPOS DIP switch!

Analysis

When in position mode, the knee can be moved, and thus zero positioning works. When in Current mode, the knee does not move after a very short while.

The EPOS for the knee is a master/slave configuration. EPOS (1) is dubbed Master, as it measures angles, and is engaged in position mode.

The knee is controlled from a controller in current mode, and the motion described is that it accellerates shortly, then bumps backwards and then settles.

The data is analysed, and the Master current seems to be close to an inverted signal of the slave, the knowledge gained is that the two EPOS are attempting to move the joint towards eachother with increasing torque, thus straining the rubber band.

To isolate the error, cables from EPOS(2) "slave" to DC-MOTOR is unplugged and controller is re-started. No motion occurs on the knee, and when data is fetched, very little current is provided.

Then cables from EPOS(1) "master" to DC-MOTOR is unplugged, and controller is re-started, Now motion occurs on the knee, and the reference is tracked.

All cables are plugged in, and the "position mode" is tested, and by means of unpluggin cables it is confirmed that it is the master, and only the master that is active in "position mode".

It is now concluded that all cables are functioning, as communication is active, and both motors can be moved in the two modes. However, something fails when in current mode. Slaves are allways in current mode, and operates fine when actively controlled, and is nicely passive when not. Hence suspescion is put on the Master.

All cables are plugged in, and a small reference is given to a controller, which is re-started, the jpoint behaves as before, and data is retrieved. Focus is put on the behavior of the currents in the early stages of motion. It is apparent from the data that the Slave moves as predicted by the controller, and after a small motion occurs, the Master builds up the opposite torque, untill the system settles. The steady state position indicates that the system settles with an offset.

It is concluded that the Master is using a controller to keep the zero position of the knee, and that the Slave is attempting to track the reference. The PI parts of the position mode (internal), and the current mode(matlab) controllers are very different, resulting in a Slave which builds up torque more aggressively than the Master, which explains the steadystate error. The experiment was not run long enough to see if an Integral part would eventually prove the conclusion.

The interface PC is started in windows, and the EPOS interface application is started and connected with RS232 to the Master EPOS. The mode is position while all other EPOS on CAN4 is in Current mode. The EPOS is succesfully forced in current mode from the interface software. The control interface on the OBC is started, and it is selected that the joint should be moved in position mode. It is registerred that all other EPOS changes to "Profile Position mode", but the left knee Master EPOS remains in "Current mode".

The RS232 is moved to another EPOS, and the knee Master EPOS is setup via CAN. From this interface the modes are changed succesfully.

It is attempted to change mode from the AAU-BOT interface, here all other EPOS than the left knee Master changes modes appropriately. The Master EPOS remains in the mode which it

was put in from the windows application.

Conclusion

PDO commands are recieved since current / position references are recieved, when mode is properly set. Direct commands fail. Adresses are wrong on DIP switch. Someone changed a DIP switch setting.

J.7.10.2 Solution / Workaround

Set the DIP switch properly.
- An automated detector has been put in the TODO list

J.7.11 CAN Write Error (timeout)

Initially when this bug appeared the robot jerks, and lost control for short period of time. However the bug has later returned with a continous CAN write error.

J.7.11.1 Origin

Hardware interrupts take too long in the OBC hardware.

Analysis

While PID controller was tested, the robot suddenly behaved badly, and temporarily lost control.

It regained control immideately afterwards, and if the controller was reloaded the problem did not re-occur.

The bug re-appeared three times during the tuning of the PID regulator.

While testing other parts of the User interface, where the useabillity and the communication should have been improved, the "CAN write error" re-appeared sporadically, and it was not clear why this happened. However more graphics on the terminal caused the error to be generated more frequently.

Logs was inspected, and usually the "CAN write error" was found in the log near a "sample time violation" from the rt_main program which executes the controller.

A lower priority of the GUI initially reduced the frequency of the problem, but did not remove it.

By analysing the device drivers written by the previous group, it was found that the "CAN write error" only appears when a semaphore is not released by the CAN I/O card interrupt handler routine within a specified time. The duration of this timeout can be set from the initialisation, but it can also be set to zero, which will cause the device driver to hang untill the interrupt routine releases the semaphore.

While this restores communication it became clear that the sample time was sometimes violated by as much as 300ms, which is similar to dropping more than 80 samples in the controller.

The code base used for execution of the controller and the I/O card drivers was studied closely. It was found that the only probable cause was that

1. Some hardware interrupt hangs the system and causes a sampling time violation.
2. Sampling time violation logs an error, and writes it on screen
3. The VGA interrupt is run with higher priority than the CAN I/O card interrupts, which sometimes causes a CAN timeout.
4. CAN timeouts is logged and written on screen, which invokes the VGA interrupt again.

Conclusion

When the VGA interrupt is invoked, the system can easily be forced into a state of timeout errors, which often results in an infinite loop of errors.

The errors shuts down communication due to timeouts.

Other high prioritized hardware interrupts which are time demanding can cause the same error as the VGA interrupt, this list includes the SMI (System Management Interrupt) and the DMA interrupt [9], which are both interrupts that cannot be avoided when on an x86 architecture.

Handling graphics (such as text in a terminal) output on the OBC, can cause the system to violate the sampling times, and thus drop CAN frames.

J.7.11.2 Solution / Workaround

The CAN write errors results in a lack of execution of the transmit routines, and as the EPOS remains in the last configured state, this becomes dangerous when in velocity or current mode.

As the hardware architecture was not replaceable within the timeframe of the thesis, the bug could not be solved. Things was done to reduce the severity of the error:

- VGA interrupts was disabled, screen outputs now is updated using Xserver from linux, the advantage is that X uses the frame buffers, and thus avoids invoking the VGA interrupts. The priority of updating the screen is greatly reduced, but in relation to the user experience it can be positively noted that updating frame buffers is faster than using the legacy VGA interrupt mode.
- USB input handling was disabled in BIOS, and is now only handled by linux
- rtmain was changed, now the controller is shut down if the sample times are ever violated, the shutdown frees time, and regains CAN control to the device driver, the last tx message in the CAN buffer is set to disable the EPOS. This has some consequences:
  - The EPOS are killed as fast as possible (faster than human reaction times), and the robot does not rotate joints uncontrollable
  - The Controller and the EPOS might be killed when the robot is dynamically unstable, or when a joint is starined beyond the stiction limits. Hence the robot might fall and get damaged if not caught by personnel. To warn the operators of the shutoff, a red flashing strobe light is automatically turned on when a sampling time error is logged.
  - A controller is not forgiven and allowed to continue afterwards if the computation time gets close to the sample time. This reduces the possible complexity of a controller structure.

J.7.12 Absolute angle Measurements are Strange

The angles obtained when measuring absolute angles, does only partly comply with actual movements, or completely fails to change, when movement occurs. The relative angle obtained updates as expected.

J.7.12.1 Origin

Mechanical link between axle and potentiometer has loosened from axle. Or measurements are obtained from a joint without absolute odometer.

Analysis

Before this problem is analysed, it is known that angles are measured using potentiometers where possible. The axle which is measured is the ball bearing axis opposite of the gears. The potentiometer axles are fastened into the bearing axles with an universal joint, and one axis-aligned screw per joint. The universal joint must be tightly fitted to the screw, and the screw must be tightened to the axle.

Rotate the universal joint by hand, if this requires more torque than expected from a potentiometer alone, the above knowledge dictates that the screw is the probable cause. The bug has only appeared during calibration of the potentiometers.

Conclusion

If an angular measurement starts behaving badly, the propable cause is that the unviseral joint to aaubot joint axle screw has come loose.

J.7.12.2 Solution / Workaround

Re-thightned screw by turning the universal joint between the potentiometer and the aaubot joint axle, and afterwards recalibrate zero positions. Use tacho meters in motors to detect this error.

The bug has not occured since initial calibration.

J.7.13 Bad behavior during Zero Positioning

The robot moves past, or away from the zero position on some joints, and possible with rapid speeds.

Origin

Either a controller is still engaged, or zero position reference is wrong.

Analysis

The behavior of overshooting, or movement away from the zero position can occur, if the zero position has been reset at a bad time. The internal position controllers operate on the Tacho, and thus needs to be reset when in zero position to gain an absolute reference. If the motion is accellerating or looks unstable the cause may be that the controller previously run was not shut off, in that case disengage the controller, as it might contain integral parts, which then can cause it to be unstable.

Conclusion

multiple causes can exhibit theese symptoms

J.7.13.1 Solution / Workaround

Hit emergency stop type (c) shut off controllers that might be loaded. unlock emergency stops, and move joints manually to zero position, reset EPOS and reset zero positions.

J.7.14 Electrical Schock when Touching Robot.

The operator experiences an electrical schock when touching the frame of the robot. If operator is standing on the rubber band of the roller the shock is within acceptable range of pain. If touching the frame of the roller, it is dependent on clothing.

J.7.14.1 Origin

The roller has no ground, and dischrages through personel or equipment.

Analysis

The schock is described best as mild, or a tingleling sensation, (and a surprising effect). It has been repeatedly observed over a period of a few hours. No permanent damage was found on the robot or the handler. The experience is described by the operators as similar to a DC shock or low amplitude continous AC schock.

The roller was powered to conduct experiments while this bug appeared, the bug was not experienced before the roller was powered on. Removing the power cable to the roller from the plug removed the error.

The voltage of the frame of the roller was measured against the ground of the PSU, the white painted frame did not exhibit any electrical properties, but measured on screws and the rotating cylinders, 108 V AC (RMS) was measured. Indicating that the paint acts as insulation.

The plug has been reversed whith no effect, and the ground on the F-type plug was measured in relation to PSU ground. 108 V was also detected on this wire. The control box for the roller is operational.

Conclusion

The roller has an internal transformer, which is noise reduced by capacitors to ground. When ground is not connected, the frame (connected to ground) is similar to a voltage division between a phase and neutral, hence the 108V. As this is a charged capacitor it is not lethal to personnel, but can be hazardous to AAU-BOT1. Personnel will often get a "mild" shock when working near the roller, which is annoying and can inderectly cause a dangerous situation.

J.7.14.2 Solution / Workaround

Cut off power from the roller, when personel is nearing it, and while operating the robot in motions that can cause it to touch metal on the roller.

The bug re-occurs when the roller is powered, the roller should be permanently removed from laboratory, or a converter should be attached to the plug such that ground can be connected to true ground.

J.7.15 Matlab Crashed, due to missing LibGL

When opening the AAU-BOT1 library, or opening a model containing simmechanics blocks, Matlab crashes.

J.7.15.1 Origin

The bug has not occured in Windows, In Linux this happens as Matlab is unable to locate the existing openGL libraries. It is not excactly known why GL is needed, however it causes a crash if it is not present.

Analysis

Google and mathworks.com presented a similar problem with other simulink blocks, and the workaround presented for solving that problem was succesfully adopted. Here the problem also concerned the libGL.so and libGLU.so

The search did not reveal why openGL is needed.

However the problem seems to be a hardcoded path in matlab installation.

Conclusion

Matlab needs help in locating libGL and libGLU

J.7.15.2 Solution / Workaround

a workaround has been found as:

```
/usr/lib$ sudo ln -s libGL.so.1 libGL.so
/usr/lib$ sudo ln -s libGLU.so.1 libGLU.so
```

J.7.16 OBC Chrashed when Giving "order to go!"

When giving order to go the OBC crashed, and the controller did not execute properly.

J.7.16.1 Origin

```
Message from syslogd@aaubot1 at Feb 17 10:24:10 ...  kernel:[ 8629.306099] Oops:
    0002 [#1]
Message from syslogd@aaubot1 at Feb 17 10:24:10 ...  kernel:[ 8629.306101] SMP
Message from syslogd@aaubot1 at Feb 17 10:24:10 ...  kernel:[ 8629.306171] CPU:
       0
Message from syslogd@aaubot1 at Feb 17 10:24:10 ...  kernel:[ 8629.306172] EIP:
       0060:[<f8cd2be1>]    Tainted: PF        VLI
...
..
Message from syslogd@aaubot1 at Feb 17 10:24:10 ...  kernel:[ 8629.306962]
    ======================
Message from syslogd@aaubot1 at Feb 17 10:24:10 ...  kernel:[ 8629.306965] Code:
    f0 0f ba 2d 1c c5 ca f8 1f 19 c0 85 c0 74 04 f3 90 eb ed c7 47 0c 00 00 00 00
    8b 97 10 03 00 00 bd c0 10 3b c0 8b 87 14 03 00 00 <89> 82 14 03 00 00 8b 87
    14 03 00 00 89 90 10 03 00 00 8b 47 14
Message from syslogd@aaubot1 at Feb 17 10:24:10 ...  kernel:[ 8629.307000] EIP: [<
    f8cd2be1>] give_back_to_linux+0x4c/0x204 [rtai_sched] SS:ESP 0068:f37ddd18
```

Analysis

It is a fault that usually occurs in a linux environment, when a variable is not protected by a semaphore, it is assumed that a driver fault or a kernel error causes it. It happens rarely, and has not been reproduced in testing.

Conclusion

The bug has not been isolated, and is not considered critical.

J.7.16.2 Solution / Workaround

No workaround has been found, but it is noticeable that the EPOS are not killed, hence a soft reboot of the OBC will not delete zero-positions, and can be re-established if the power has not bee cut from EPOS.

J.7.17 Left Ankle Pitch Zero Position Error

Left ankle rotates to maximum, or random location instead of zero position

J.7.17.1 Origin

The screw that establishes friction between drive shaft on the motor, and the small gear which interacts with the rubber belt is not tightened properly.

Analysis

Initially the problem occurred seldom, and was initially not solved, as potentiometers provided a better angle measurement, and could be used for zero positioning. After an unrelated re-assembly of the ankle, the problem was increasing in amplitude, and in frequency, and it became necessesary to investigate it.

The suspection was first given to the HDG, as it was detected close to the zero position, and as the belt was removed, the HDG was turned gently by hand to see if any errors could be detected, this was not the case, but during this test another gear was accidentically nudged, and was moveable.

The gear in question was attached to the DC motor by means of a small screw vertically on the axis, and should not be moveable. The screw was tightened and the joint reassembled, and the bug disappeared.

Conclusion

The screws connecting the gears with the DC motor shafts needs maintanence, the problem can be redetected if the zero position starts to drift.

J.7.17.2 Solution / Workaround

Disasemble joints, tighten screws in all gears, reassemble joints.

J.7.18 Zero Position Error after Emergency Stops

The robot does not return to zero position, after the emergency stops has been shut. Instead it remains in current position.

J.7.18.1 Origin

Unknown

Analysis

The emergency stops has been shut off during controllers, resulting in the LED's on the EPOS turning red. Sometimes the EPOS registers this as a new zero position when reset.

Conclusion

Unknown bug causes the EPOS to forget zero position after the 60V has been removed.

J.7.18.2 Solution / Workaround

None, but allways check if zero positioning works after emergency stops. The robot is equipped with marks allowing a manual reset to zero position, if the EPOS has forgotten it.

J.7.19 EPOS Sensor Breach

One or more EPOS reports sensor breach

J.7.19.1 Origin

Cable for tacho has snapped, or shorted.

Analysis

The left ankle was pitched to high, traping the cable between the aluminium frame and the gears. Afterwards the cable was fragile and nearly cut through. This was noted, and during the movement of the OBC to the AAU-BOT frame, the EPOS related to the ankle started reporting the error.

The cable was visually inspected and it was found that it was breached.

Later the cable was repaired and the EPOS did no longer report a sensor breach.

Conclusion

A cable for one of the DC-motors is broken. The EPOS number reveals which.

DO NOT attempt to operate the robot, the joint will not respond properly !

J.7.19.2 Solution / Workaround

Repaired broken cable.

J.7.20 Left Ankle Pitch Positioning Error

Left ankle rotates close to, but not past the zero position, direction is not important.

J.7.20.1 Origin

The ball bearings mounted in the plastic wheel which keeps the rubber belt tight has been locked by malfunctioning distance disk.

Analysis

The joint was reassembled to detect the reason for this error, and the HDG, and DC motors was turned to detect an increase in friction. This was not present, but when turning the plastic wheel (mounted on ball bearings) a high friction was found.

The wheel was loosened and the friction disappeared. This indictated that the ball bearings was either broken or locked. The wheel was disassembled, and it was found that the distancer disk between the inner axles of the two ball bearings had a deep groove. Further research showed that this groove was so deep that it could provide friction between the inner and outer axle of the ball bearings.

Conclusion

The distancer disks get worn, and should be replaced with some unknown time interval.

J.7.20.2 Solution / Workaround

Replace distance disk on a joint with increasing friction. Add some regular service routine to the robot maintanence plan, to avoid similar bugs.

J.7.21 Crane is locked and does not move.

When attempting to lift the robot the crane fails to move, but remains locked.

J.7.21.1 Origin

Shortcircuited capacitor causes lock to fail and the crane cannot release the lock.

Analysis

Fortunately the crane discontinued operation as a result of a maintenance of the cables. This meant that the robot was standing on it's feet while the crane broke down.

The robot was then mounted on experiment rails, to ensure that it would not fall while the crane was disassembled.

The operator handle was disassembled as it was warm, and when opened it smelled burnt, and multiple connections was melted a bit. The capacitor exhibited all known symptoms of a broken capacitor, (smell, burn marks) and it was not replaced as it was not easily obtainable.

Conclusion

Something caused the handle to overheat while the crane was rolled out and spun in again. The probable cause is a design flaw in the crane, as it gets hot after continous operation up to about 1 minute.

J.7.21.2 Solution / Workaround

Replaced crane. Do not run crane for a longer period of time.

J.7.22 IMU Quaternion readout error

When running a controller which utillizes the IMU, the IMU can sometimes deliver errornous quaternion data. Either zeros only, or it fails to update after the first sample. Hence the controller does not work as expected.

J.7.22.1 Origin

Unknown

Analysis

The IMU interface code was rewritten to output desired information.

After this was done, compiling the controller sometimes resulted in a bad configuration being sent to the IMU. All old files was deleted and this temporarily solved the problem.

Conclusion

Somewhere on the OBC some of the old code is available to the compilation process, and old code is used instead of the correct.

J.7.22.2 Solution / Workaround

no suitable solutions has been determined yet, but a workaround can be "make clean all"

K

Users Manual for AAU-BOT1 Laboratory

This appendice describes the hardware setup, as it appeared in the spring of 2010.

Safety Procedures

When operating AAU-BOT1 It is important to remember that the robot has no sensors to inform it that humans are in the vicinity of the robot, or any sensors relating to collision detection, neither with it-self or the environment. Hence operators should at all times consider some basic safety procedures.

On top of this AAU-BOT1 is expensive and budget limited, and no spareparts has been produced, so it will be both economically a challenge and time consuming to fix any damages caused by untimely risks taken by operators or observers.

Operators and Observers

- At any given time the robot should be operated by 1 person, and at least 1 observer. The observer must be in the laboratory and keep a watchfull eye on AAU-BOT1 while the operator issues commands or enables controllers. This is necessesary to avoid the robot damaging it self or the environment, if the controller has a malfunction.
  - The robot aquires initial angles on joints during the loading of a controller, this happens before the controller is enabled, and thus the observer should also be aware of the robot during initialisation. To warn the observer the build procedure turns on the yellow warning light, which indicates that the robot may become active.
- The AAU-BOT1 OBC can be controlled through a relatively simple SSH interface, and hence it is possible to actuate the robot without being physically in the room, this is not recommended, and should not occur even under debugging, as the observer is then left alone with the activated AAU-BOT1 and if any misfortune happens no one can help him! The purpose of this interface is that the OBC can be controlled, while mounted on AAU-BOT1 without requirering the operator to have a keyboard connected directly to the AAU-BOT1.
- The Observer should at all times have a hand placed on one of the emergency stops, as any malfunction can then be quickly shut off. Any distance to the emergency button will cause a slower response time, and as the system can move quite rapidly this is not desired, not even when testing a well known movement, as any a breakdown of the CAN communication can cause sudden and unforseen reactions from the robot.
- The treadmill is build with unfortunate electrical properties. It should not be connected to regular power. If simply connected to power, personel, AAU-BOT1, or other electrical

equipment must not have contact with the frame of the treadmill. Use the treadmill only if the roller and the equipment nearby has ground.

- ▷ Observers and Operators should at all times when the robot has powered actuators stay at least
  - $1\frac{1}{2}$ meter away from AAU-BOT1 in the robots sagital plane
  - $\frac{1}{2}$ meter away from AAU-BOT1 in the robots frontal plane
- ▷ Observers and Operators may interact with robot only when emergency stops are pushed, and power to the actuators are disabled. This is to avoid any unforeseen movements which may damage the operator and or observer.
  The EPOS amplifiers have LEDs indicating their state, and lights up blinking green when inactive, they will become red if the OBC has not configured them and / or if the emergency stop has been pushed, do not trust theese indicators blindly, allways push the emergency stop button before handling the robot.

Emergency Stops

The most important safety feature on AAU-BOT1 is the emergency stop buttons. In Figure K.1 is a schematic on the connections for AAU-BOT1, which labels the emergency stops (c), (b) and (a).

For regular use of AAU-BOT1 where the safety procedures are respected, the (c)-type emergency stops are sufficient as they kill power to the actuators. This preserves OBC settings and EPOS calibrations, however if anything goes wrong with the amplifiers, computers, or if anyone gets an electric shock while working in the laboratory, the (a)-type emergency stop will cut off all power to all the systems, and should be used.

The (c) type emergency stop will allow the EPOS to draw a huge amount of power from the 12V DC PSU (read: near 30 amps). Which after some time drops the voltage level to approximately 9V. It has been done frequently during developement, and does not seem to damage the equipment, so don't worry about using the stops.

The (b) type, disconnects the equipment supplyed through the table plugs, i.e. laptop computers, screens and desktop computers. It does NOT disable the 60V supply to the EPOS, nor the 12V supply to the OBC so this stop will NOT disable the motors on AAU-BOT.

The electrical crane motor, and the treadmill control has small emergency stops affixed to the control interfaces, theese are selfcontained circuits, and does not effect the rest of the laboratory, the (a) type will stop both the crane and the roller.

The physical locations of the emergency stop buttons is illustrated in Figure K.2.

Interfacing with OBC

The OBC is a standard computer and has an onboard ethernet controller, and an onboard graphics adapter. While the OBC is standing on the workstation table, (assuming it is unmounted from AAU-BOT1), a screen, keyboard, and mouse can be connected to it and it can be interfaced as a regular pc. It has an ssh server installed which allows interfacing to it using ethernet.

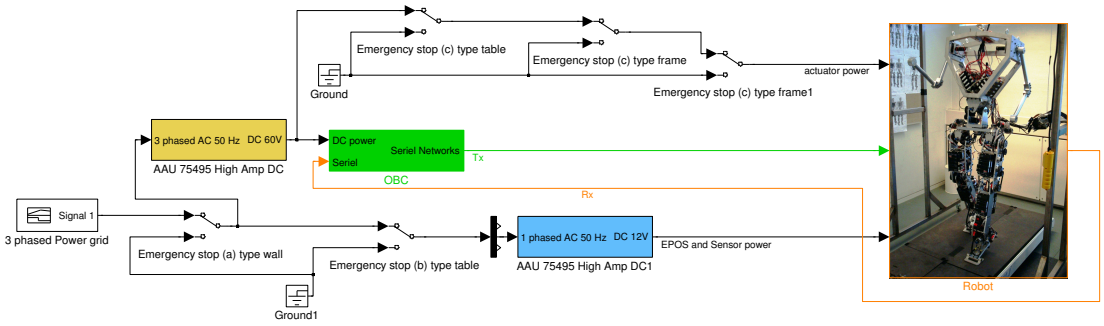


Figure K.1: Schematic containing the Emergency stops and the connections to the AAU-BOT1 hardware.

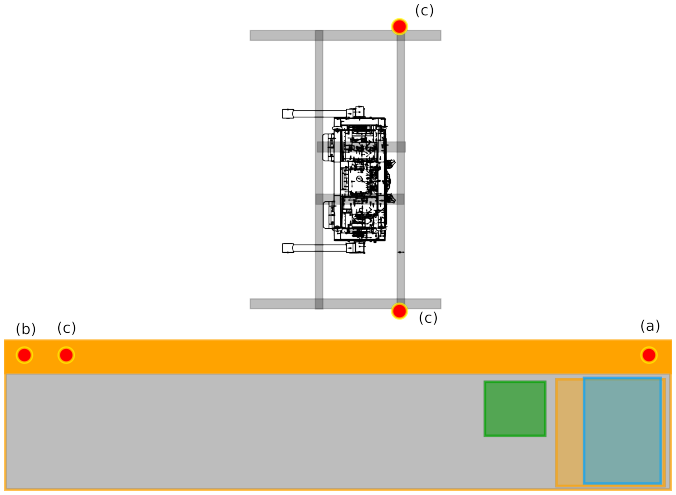


Figure K.2: Sketch (not to scale) of the laboratory, with location of the two types of emergency stops in relation to the robots position

Compiling, Building, and Running Controllers

Required Tools

- ▷ Matlab 2009b
- ▷ SimMechanics Toolbox
- ▷ Real Time Workshop Toolbox

Getting the code

The code is located at `/Matlab` folder on the DvD included with this project. This represents the code at delivery time. Alternatively the updated code can be fetched from the AAU-BOT1 homepage by running

```
user@localhost:~$ git clone http://www.aaubot.aau.dk/10gr1036/
```

```
user@localhost:~$ cd 10gr1036
```

If you have copied the code from the CD the latest changes can be fetched by running:

```
user@localhost:~/10gr1036$ git pull
```

inside the repository.

Repository structure

- Matlab – Source code of controllers and dependencies
  - aaubot_control – Control and supervisor program.
  - aaubot_simulink – Matlab blocks to communicate with the EPOS from the obc.
  - include – Header files for the aaubot system.
  - drivers – Linux drivers for the digital I/O.
  - libs – Dependencies for aaubot system.
  - SimpleTest – Small controller which moves the right arm back and forth.
  - SimLink - 2nd version drawings – Nonlinear Matlab simulation of the robot.
  - Untar-To-Matlab.tar.bz2 Installation.
  - More good stuff...
    - rtw/c/rtai – rtai target for real time workshop
- Documentation – source documents for the rapport
- More good stuff...

Generating the Code

To generate the code it is necessary to unpack a tarball from the repository to your local matlab installation. The file "Untar-To-Matlab-Installation" should be unpacked into "/usr/local/matlab/rtw/c/" or a similar place depending on where your local matlab is installed.

Now open your matlab and navigate to the Matlab folder.

Run the RunMe.m file which sets up the paths and necessary variables.

Automatic code execution

Open the Matlab/SimpleTest/test.mdl model and press save.

in a terminal navigate to the Matlab folder and run "./aau_engage". Everything should be done automatically[1]

Manual code execution

Enter Matlab/SimpleTest folder, which is important since matlab will output the generated files to the current directory, and the makefile expects the generated files to be placed inside this folder.

Open the test.mdl and click ctrl+b to build the model

[1] Initial positioning,FTS startup and IMU startup still has to be done manually after each reboot

Building the Code

Now copy the Matlab folder to the obc which have a compiler environment has been set up.

```
user@localhost:~/09gr936$ scp Matlab root@192.38.56.109:~
```

Now ssh to the server and run the makefile like this

```
user@localhost:~$ ssh root@192.38.56.109
root@obc:~$ cd Matlab/SimpleTest
root@obc:~/Matlab/SimpleTest$ make -C ../ clean all
```

Activating the EPOS and the Control program

It is recommended that open this in another terminal.

```
user@localhost:~$ ssh root@192.38.56.109
root@obc:~$ cd Matlab/aaubot_control
root@obc:~/Matlab/aaubot_control$ ./aaubot_control
```

This will bring out a terminal program that looks like this:

```
--------------------=00=-----------------
            AAUBOT-1 Main Menu
Please select an option that you need:
  1. Start FTS streaming
  2. Stop FTS streaming
  3. Request FTS streaming status
  4. Request current FTS queue status
  0. Get tx count

  5. Reset all EPOS
  6. Setup EPOS
  7. Start motor
  8. Request EPOS measurements
  9. Request EPOS error history

  f. Move joints to zero positions
  p. Change joint positions
  o. Print current positions
  r. Reset joint relative zero position
ESC. QUICK STOP all EPOS

  d. Disable CAN transmissions
  e. Enable CAN transmissions

  h. Reset CAN error counts
  i. Get all CAN error counts

  j. Enable initial zero positioning
  k. Disable initial zero positioning

  a. Setup IMU and start measurement
```

```
  b. Stop IMU
  c. Request IMU count
  g. Request current IMU queue status

  [space]. GIVE ORDER TO GO

  s. Reset supervisor
  q. exit
Your choice:
```

Here you can reset the EPOS by pressing "5" Which will output this

```
Resetting all EPOS.
Resetting EPOS no. 2.
Resetting EPOS no. 3.
Resetting EPOS no. 4.
Resetting EPOS no. 5.
Resetting EPOS no. 6.
Resetting EPOS no. 7.
Resetting EPOS no. 8.
Resetting EPOS no. 9.
Resetting EPOS no. 10.
Resetting EPOS no. 11.
Resetting EPOS no. 12.
Resetting EPOS no. 13.
Resetting EPOS no. 15.
Resetting EPOS no. 16.
Resetting EPOS no. 17.
Resetting EPOS no. 18.
Resetting EPOS no. 19.
Resetting EPOS no. 23.
Resetting EPOS no. 24.
Resetting EPOS no. 26.
Resetting EPOS no. 29.
Resetting EPOS no. 31.
Resetting EPOS no. 32.

All EPOS are reset and should be blinking green now.
```

If you get any errors about CAN channel not responding check for loose cables, but assuming everything went well all the EPOS should now be blinking green, which is a ready state where input is not read. Now press "f" follow by an "a" which should put the robot zero position using the potentiometers.

```
Select zero position mode: (r)elative, (a)bsolute, (c)ancel
All joints should now be in relative zero position...
```

This will activate the the actuators so be ready with the emergency stop if needed. The EPOS should also switch from blinking to a constant green.

Running the Code

Running the code is a two step process. First start the controller, which might actuate the robot so keep a hand on the emergency stop, but should not happen in this example since the robot should already be in zero position.

```
root@obc:~/Matlab/SimpleTest$ ./test -v -o -c 1 -w
Target settings
===============
  Real-time : HARD
  Timing    : internal / oneshot
  Priority  : 0
  Finaltime : RUN FOREVER
  CPU map   : 1
```

After 6 seconds the initial positioning should be done and the robot should not be able to move more, before you activate the real controller

```
Target info
===========
  Model name              : test
  Base sample time        : 0.004000 [s]
  Number of sample times : 1
  Sample Time 0           : 0.004000 [s]

Target is waiting to start.
```

and will stay here until you press space in the aaubot_control program. As a special notice here is that if you see any of the sample times that is 0 you model will not work as expected, and you should modify you matlab model not to have any continuous states. A hint here is to use the digital clock, instead of clock for signal generators.

When you feel ready to run the controller keep a hand over the emergency stop and press space in the terminal with the aaubot_control program, and in the terminal with the controller is should now write

```
Target is running.
```

To stop the program switch the the terminal in which the controller runs and press ctrl+c

```
Target is stopped.
AAUBOT EPOS write closed.
AAUBOT EPOS read closed.
Target is terminated.
```

Feature Notes on Handling AAU-BOT1

- ▷ When (c) type emergency stops have been shut off, the robot is unable to actuate, but sensor readings can be retrieved. This feature is usefull for calibration, or handling of the robot
- ▷ AAU-BOT1 is able to stand-up due to joint friction alone. Hence the crane is not necessesary to keep the robot upright, the crane is for easy handling, and safety gantry only.

- ▷ The previously undocumented FTS amplifiers, are located inside the shin in an aluminium box.
- ▷ To measure from EPOS sensors, it is necessesary to provide an input to EPOS, this may be a zero input.
- ▷ If the status LEDs on the EPOS are red, but dimmed, it indicates that the EPOS current is lower than usual, and this is usually because an emergency stop is probably shut off.