Dynamic Walk of a Biped Robot

Using model predictive control and a linear inverted pendulum model

Project Report

Group 1035

Aalborg University
Control and Automation

AALBORG UNIVERSITY
STUDENT REPORT

Department of Electronic Systems Control and Automation
Fredrik Bajers Vej 7
DK-9000 Aalborg
`www.es.aau.dk`

Title:
Dynamic Walk of a Biped Robot

Theme:
Master's Thesis

Semester:
4. Semester Control and Automation

Project Period:
1. February 2020 - 4. June 2020

Project Group:
1035

Group Members:
Frederik Arentz Sørensen
Søren Klintby Madsen
Thomas Juul Nielsen

Supervisor:
Henrik Schiøler

Report:
96 pages

Submitted:
4. June 2020

Abstract:

This report sets out to develop a MPC controlled biped robot with a simplified 6 DOF leg kinematic model, and a simple dynamic model. The biped robot aims at achieving dynamic walk with a walking velocity of 0.5 m/s, while being built with affordability in mind.
With this in mind the robot was developed with 3D printed links, joints actuated by NEMA 17 stepper motors, appropriate sensors and an ESP32 as the controller. The biped model was created in the simulation environment Simulink with a LIP model generating feet trajectories for the COM of the robot. From the LIPM, inverse kinematics was used to find the necessary joint angles for the dynamic walk.
The LIPM was controlled with a MPC controller and proved the stability of the biped robot if the COM trajectory could be achieved.
Simulation of the dynamic walk showed model errors were present and the errors were not found and solved before the project deadline.

Preface

This report was written by group 1035 on the 4rd semester of Control & Automation at Aalborg university. The project has been realised during a period starting at the beginning of February up to the beginning of June 2020.

Aalborg University, June 4, 2020

[illegible]	[illegible]	Thomas J.N.
Frederik Arentz Sørensen	Søren Klintby Madsen	Thomas Juul Nielsen
fasa15@student.aau.dk	smad15@student.aau.dk	tjni15@student.aau.dk

Reading Guide

It is assumed the reader has a basic understanding of robotic manipulators and how they can be modelled kinematically. The reader should also has an understanding of control and how systems are modelled/simulated. A very basic understanding of electronics such as motors and microcontrollers is advised.

Throughout this report references to figures will be shown as figure 1.0, tables as table I, equation as equation 1.0 and citations will be shown as [1]. Every figure has a description underneath and tables have their description above. Footnotes will be shown as footnote[1] at the bottom of the page.

Glossary

ABS	Akrylonitril-butadien-styren
CAD	Computer Aided Design
COM	Center of Mass
DH	Denavit-Hardenberg Parameters
DOF	Degrees Of Freedom
IMU	Internal Measurement Unit
I^2C	Inter-Integrated Circuit
LQR	Linear Quadratic Regulator
LIP	Linear Inverted Pendulum
LIPM	Linear Inverted Pendulum Model
MIPS	Million Instructions Per Second
MPC	Model Predictive Control
PID	Proportional Integral Derivative
PLA	Polylactic Acid
Pose	Position and Orientation
ROM	Range Of Motion
SSI	Synchronous Serial Interface
SPI	Serial Peripheral Interface-bus

Contents

CHAPTER 1
Introduction

The development of humanoid robots is motivated by the possibility of robots assisting humans in tedious, hazardous or physically demanding work that could result in injury or disability. One step towards realising this, is to develop a motion controller that makes the humanoid robot walk.

Development of humanoid robots is not a new concept as the first full scale humanoid robot was created in 1973 [1]. The robot was called WABOT and was developed at Waseda University. Since the creation of WABOT, universities and companies around the world have furthered the development with noteworthy mentions being Honda Motor CO. Ltd, with their creation of the P-series humanoid robots development program staring from the year 1986. Honda also created the state of the art Asimo robot series in 2005, being one of the most complex humanoid robots on the market [1].

The complexity comes at a price being that the Asimo robot costs an estimated 2,500,000 USD as of 2012 [2]. Asimo can be seen in figure 1.3. Other humanoid robots as the HUBO 2 created by KAIST cost an estimated 400,000 USD, being less complex than Asimo while still being a full body robot [1] [3]. The robot HRP-4 developed by Kawada Industries costs approximately 300,000 USD [2].

The price of such robots can be a restricting factor when studying the development of control theories for humanoid robots, as it makes the hardware less accessible. This report will focus on the development of a cheaper humanoid robot which can be used for teaching, research and pushing the boundaries in humanoid robot development. Developing a full body humanoid robot is not a trivial task and as this report was created in a single semester, the development was narrowed down. The development was reduced to just the lower body of a humanoid robot. This should be able to perform dynamic walk, inspired by how humans walk. Analyses of the current state of art regarding human gait and control theories for a biped robot is to be carried out, with the project solution based on the findings.

[1] full body robot: One that consist of head, torso, arms and legs

1.1 State of the Art

A wide range of innovative humanoid and biped robots have been developed throughout the years, from which this section will investigate a select few. The robot to be developed aims at the same functionality of dynamic walk as the state of art.

Atlas

Atlas, by Boston Dynamics, is a humanoid hydraulic actuated robot consisting of 28 joints. It has a height of 1.5 m and a 80 kg weight, and is capable of walking at a speed of 1.5 m/s [4]. Boston Dynamics often showcase the dynamic movements and agility of Atlas in promo videos, sporting almost human like agility and demonstrates the state of art humanoid robot control level. Atlas is based upon a COM trajectory planner, a full body inverse dynamic model and a MPC controller to make the robot perform dynamic walking [5]. The COM planner is based upon LIPM dynamics created for the robot thus resulting in an optimisation problem. The resulting control allows Atlas to walk on rubble, handle push's and walk dynamically.

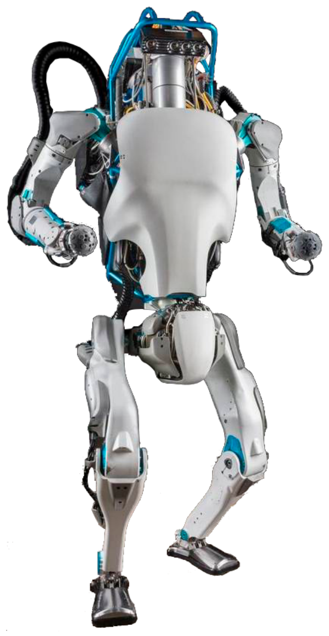


Figure 1.1 Atlas

Cassie and Digit

Cassie is a commercially available biped robot from Agility Robotics [6]. Cassie can be combined with an upper body, resulting in the robot Digit. The unactuated dynamics of Cassie include compliant materials, i.e. spring dynamics. The control of motion for Cassie has been posed as a nonlinear problem, and the developed controller has a sample rate of 2 *kHz* in order to handle the dynamics [7]. The results presented in [7] showcase a hybrid model to overcome the unique challenge that Cassie poses due to its compliant mechanism and highly underactuated nature of the dynamics. The hybrid model consist of a rigid model (simple model) and compliant model (full model), the simple model assumes all four leaf springs are rigid linkages, whereas the full model treats the rotational joints of the leaf spring linkage as a torsional joint, with stiffness and damping effects. These two models are compared in regards to computation performance and simulation, which suggested two directions. One being ignoring the compliance and designing controllers which are robust to the mismatch or using a more complex model which designs locomotive behaviours encoding the compliant behaviour. Contact detection is achieved from checking if the axial leg force is greater than 75 *N* and maintained for over 5 *ms* then contact is registered for a given leg. Additionally, if the force falls below 75 *N*, contact is considered broken. This estimation routine runs at 500 *Hz*, while the control thread runs at 2 *kHz*.

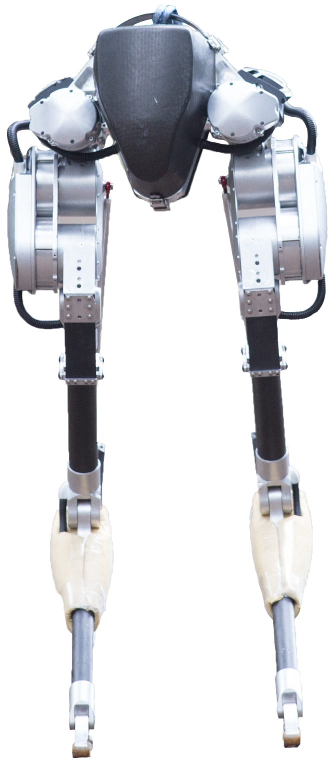


Figure 1.2 Cassie

Asimo

Asimo developed by Honda is a 34 DOF humanoid robot created to imitate human motions.
In contrast the human body has 57 DOF distributed as such: 3 DOF for the head, 18 DOF for the arms, 26 DOF for the hands, 2 DOF for the hip and 16 DOF for the legs [8].
The actuators of Asimo consist of servomotors, harmonic speed reducers and drive units, which together with a 6-axis area sensor in the foot, gyroscope and acceleration sensor in the torso enables Asimo to walk with a speed of 2.7 km/h. Asimo has a height of 1.3 m and a weight of 50 kg [8].

Different work has been made on Asimo by different researchers. The paper *The intelligent ASIMO: system overview and integration* has worked on advancing Asimo gesture recognition system, human interaction and task performance, with the primary focus on doing receptionist work [9]. While the paper *Footstep Planning for the Honda ASIMO Humanoid* has implemented a footstep planner for Asimo to advance its dynamic walk by being able to avoid obstacles [10].

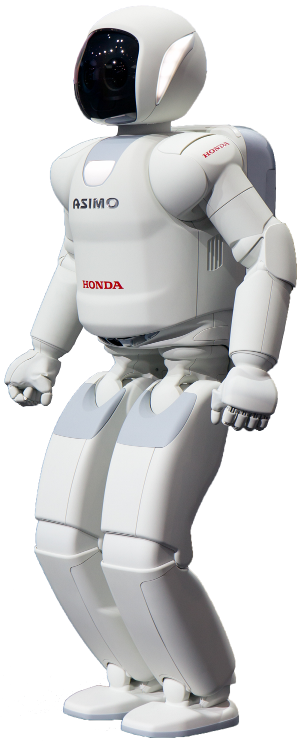


Figure 1.3 Asimo

Wabian-2R

Wabian-2R was developed by Kato/Takanishi Laboratory in Japan and is a 41 DOF humanoid robot created as a human motion simulator. It is 1.5 m high and weigh 64 kg. It is able to walk with a speed of 1.8 km/h with a step length of up to 0.5 m [11]. It is able to perform human like walking with its knee stretched, heel contact and toe offset. It is also able to walk with a horizontally oriented foot with sensors placed in each corner to measure ground contact and enables the robot to walk on uneven terrain and up/down slopes.

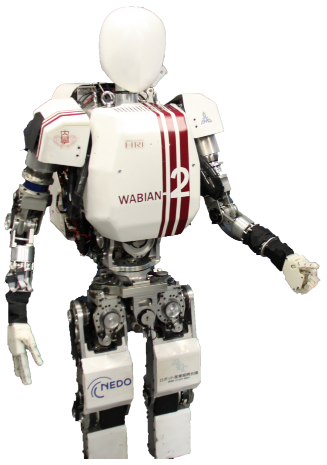


Figure 1.4 Wabian-2R

LOLA

LOLA is a humanoid robot with a total of 25 DOF. The legs each have 7 DOF, with the 7th joint being the toe joint. This makes the LOLA robot able to closely mimic a human gait, by adding a heel lift-off phase to the gait. While using single leg support during walk, LOLA has 9 DOF between the supporting foot and the upper body. This high amount of redundancy bring advantages such as increased agility, and reduces load on the individual joints [12]. The LOLA project aims for 5 km/h walking speeds. [12] mentions that for higher walking speeds, the time horizon of the motion planner must be extended beyond just a single step, to include multiple steps. An important part of stabilisation is real-time adjustment of gait parameters such as step length, width and frequency [12].

1.2 Gait Analysis

How humans walk has been a research subject for many decades with focus on how walking patterns are in normal gait and how the gait changes as age and injury affect the person.
This research is usually applied on biped robots in order to make them mimic human gait, as seen from the Asimo, Atlas, Wabian and LOLA.
The human body produces metabolic energy. Muscles use the energy when moving and energy usage increases as speed increases. Walking more efficient uses less energy and thus allows for greater speeds [13] To conserve energy humans developed dynamic walking. How long or short steps a human takes depends on the walking speed, where longer steps increases collision losses and shorter steps increases energy used when swinging the legs. Many factors affect the energy per-

formance for dynamic walk and the most optimal step length vary from human to human as leg properties change. The human gait will be analysed in the following section, such that it can form a basis for later decisions in the development of the biped gait.

Human Gait

The human gait can be described as a series of stances as the foot pushes off the ground, swings, and performs a ground collision which triggers the movement of the other leg [13]. In figure 1.5 the stances can be seen showcasing single foot support and double foot support during a gait cycle. Walking creates a sinus-like movement pattern for the COM mass as the support shifts from leg to leg when performing dynamic walking [14] [15].

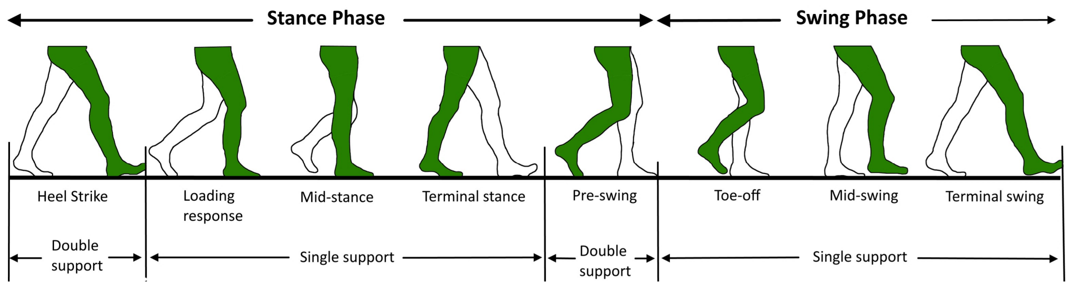


Figure 1.5 Human gait describes by the individual stances with single or double foot support [16]

Range of Motion

The normal human range of motion for lower body joints can be seen in table I. The human body is flexible and can hyperextend most joints, however this is not includes in table I. The notion of moving a limb, can be separated into the following categories of motions, called: flexion [2], extension [3], abduction [4], adduction [5], dorsiflexion [6], plantar flexion [7], eversion [8], inversion [9] All values of table I are based on the upright standing position as the zero point.

[2]Flexion refers to a movement that decreases the angle between two body parts.
[3]Extension refers to a movement that increases the angle between two body parts.
[4]Abduction is a movement away from the midline.
[5]Adduction is a movement towards the midline.
[6]Dorsiflexion refers to flexion at the ankle, so that the foot points more upwards
[7]Plantar flexion describes the extension of the ankle so that the foot points downwards.
[8]Inversion involves the movement of the sole towards the midline.
[9]Eversion involves the movement of the sole away from the midline.

TABLE I
RANGE OF MOTION OF HUMAN JOINTS [17] [18]. HIP FLEXION IS BASED UPON SINGLE LEG FLEXION WITH THE OTHER IN A NEUTRAL POSITION. THE NEUTRAL POSITION IS STANDING WITH STRAIGHT LEGS.

Range of motion (°):	**Max flexion**	**Max extension**
Hip - forward	110	30
Hip - sideways	*n/a*	40
Knee	150	0
	Max Dorsiflexion	**Max Plantar flexion**
Ankle - forward	20	50
	Max Eversion	**Max Inversion**
Ankle - sideways	12	23

In figure 1.6 the knee joint can be seen describing how the joint moves. With the ankle joint having 3 DOF the human foot can the tilted such full ground contact is obtained while walking up/down hills or uneven terrain. Walking in debris and with protruding objects in the ground is also possible.

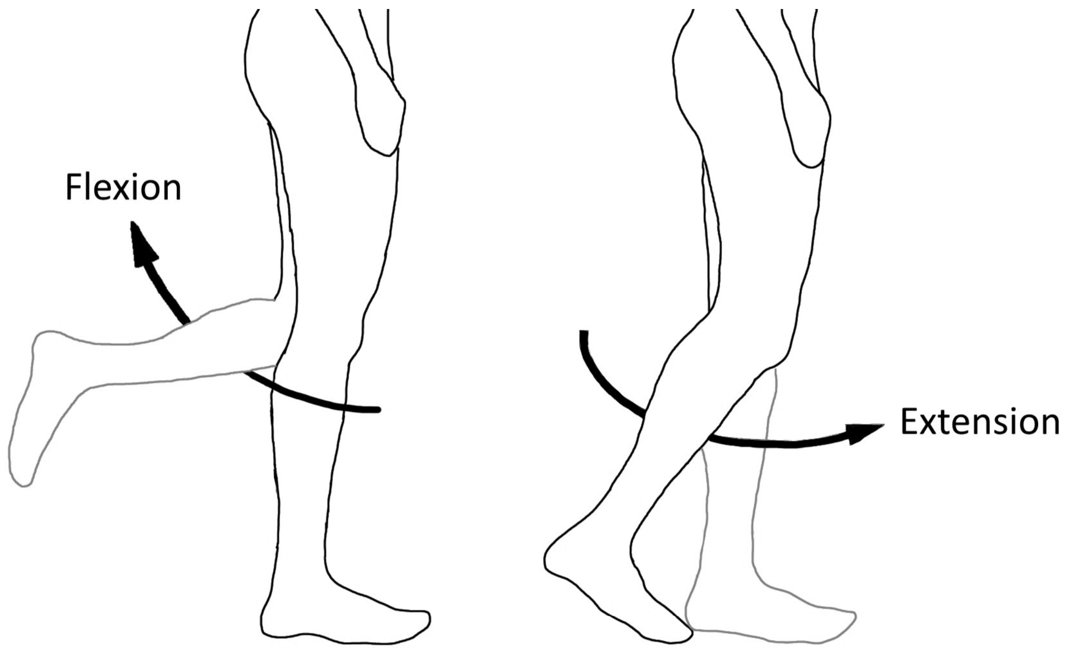


Figure 1.6 Human gait with arrows indicating the movement of the knee joint.

The hip, knee and ankle joint extend to the values seen in table II as dynamic walking is performed. Theoretically the human joints could be limited to said values if dynamic walk was all humans did.

TABLE II
Range of motion based upon normal gait at speeds between 0.4 to 1.6 m/s [14]. The knee joint can only do flexion.

Range of motion (°):	**Max flexion**	**Max extension**
Hip	34.33 ± 7.17	11.45 ± 5.11
Knee	62.42 ± 2.89	0
	Max Dorsiflexion	**Max Plantar flexion**
Ankle	15.23 ± 2.66	21.54 ± 3.71

Joint Actuation

As a human moves, the joints accelerates and decelerates resulting in dynamic walking being performed. Depending on the walking speed the joint velocities change and the most optimal joint actuation based upon energy conservation is used. For a normal human the peak joint angular velocities for the hip, knee and ankle can be seen in table III. As the energy usage is dependant on the weight of the limb the actual energy for each joint is different from human to human. The joint velocities are however similar and thus forms a basis from which specific joint torques can be found.

TABLE III
Peak joint angular velocity based upon normal gait at speeds between 0.4 to 1.6 m/s [14].

Peak angular velocity ($\frac{deg}{s}$):	**Max**
Hip	219.82 ± 27.87
Knee	386.39 ± 50.48
Ankle	331.16 ± 34.68

Robot Gait

To imitate the human lower body a biped robot can be used when dynamic walking is the focus. Due to the many degrees of freedom found in the human body, the biped robot representation is generally simplified with a less complex leg model,

in terms of the amount of DOFs. Some of the major functionality that is lost is the toe-joint of the foot. This joint is found in the forefoot of the human feet, and adds the toe-off phase to the human gait. The Lola robot [12] includes such a toe-joint, however it is seldom included in the design of a biped robot. Removing the toe-joint alters the pre-swing phase of the robot gait. An additional phase that is often removed to simplify the robot is the heel-strike phase. The heel-strike phase requires a foot capable of a rolling motion as the heel hits the ground. Such a robotic prosthetic is complex to recreate thus few robots implement a heel-strike phase.
The complexity of the human foot is hard to mimic as it has 30 DOF [8]. However simplified versions of the legs are often created. The robot Cassie has a 5 DOF leg while LOLA has 7 DOF. The 7 DOFs of the LOLA robot is distributed as: 3 in the hip, 1 in the knee, 2 in the ankle and 1 in the toe. Finally another simplified version of the leg, used in the Asimo robot, uses 6 DOFs where 3 are located in the hip, 1 in the knee and 2 in the ankle. The purpose of these simplified models is to approximate the function of human legs when creating humanoid robots, while reducing the complexity of the model.

As a result of the knowledge gathered up to this point, the robot used for this project will consist of a 12 DOF model, having 6 DOF in each leg, distributed as such: 3 DOF in the hip, 1 DOF in the knee and 2 DOF in the ankle. However walking on uneven terrain can be challenging with a regular robotic foot, therefore the biped robot designed for this project will be fitted with force sensors in each corner of the foot. This method is also used by the Wabian-2R robot, so as to provide feedback to the controller about how the foot is affected by the terrain.

The phases of the biped robot gait will mimic the human gait seen in figure 1.5 on page 6 to some degree. Due to the reduced amount of DOFs in the leg of the robot, certain phases cannot be replicated. These include the heel strike phase, the pre-swing phase and the terminal stance phase, which is illustrated in figure 1.5 on page 6. These phases have in common that only a part of the foot is in contact with the ground, meaning that the robot rests on either the heel or the toe. These phases has been altered such that the entire sole of the feet of the robot remains in contact with the ground. The phases of the robot gait can be seen in figures 1.7, 1.8 and 1.9. With figure 1.7 showing the phases of static robot gait, meaning that the COM, remains over the supporting foot, while going through the different phases. Figure 1.8 also show static gait, but instead of showing it from the side, this figure shows the phases from the back.
Figure 1.9 illustrates the concept of dynamic gait on the robot, by the COM is moved outside the supporting foot, which will lead to the robot falling over if not countered by placing the other foot at a stable position.
It should also be noted that the base of the robot is tilted forward, which is not

used in this project, as the base orientation and its height is considered fixed in order to simplify the model. However in order to include this, a feedback of the base orientation should be added to the model.

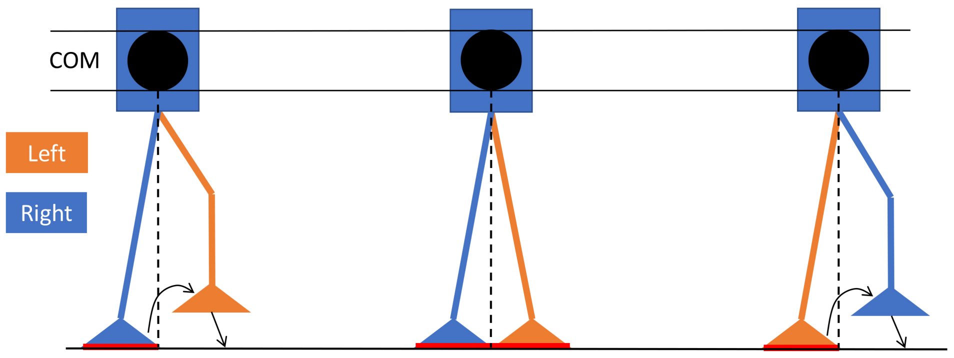


Figure 1.7 The phases during static gait seen from the side. Here the COM is visualised as a black dot, with it's ground projection shown with the black dotted line. The supporting foot is shown by the red line.

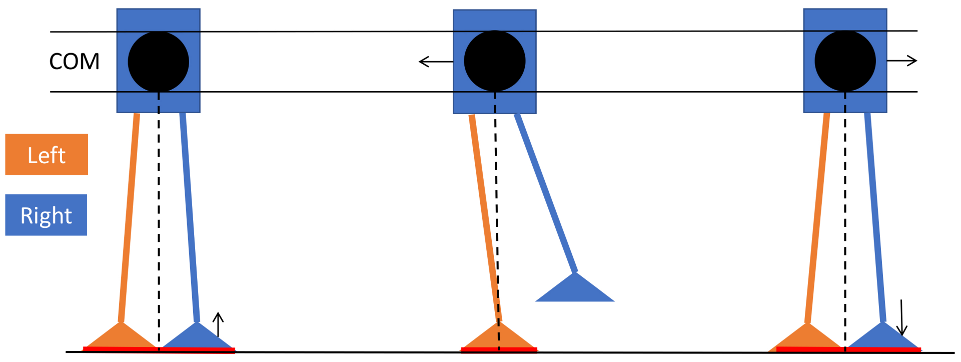


Figure 1.8 The phases during static gait seen from the front.

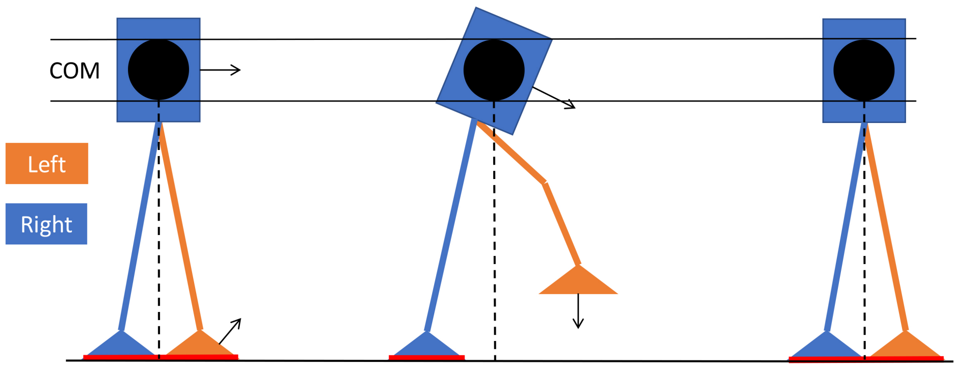


Figure 1.9 The phases during dynamic gait seen from the side.

Phases of the Biped Gait

This section will explain further in detail the mechanics of the different phases of the biped robot gait. The flowchart shown in figure 1.10 is a graphic representation of the phases explained within this section.

- Initiate walk:
  This is the first step of the walk, which will begin by a left foot step. Here the robot starts by standing still. Both feet remain in contact with the ground and stationary. The COM is then shifted above the right foot. This prevents the robot from falling when the left foot is lifted. Further explaining why this is the case, will be done in section 4.1 on page 41 and section 5.5 on page 49. This phase of the gait will only be repeated once, while the remaining phases of the gait will be repeated until the desired destination is reached.
- Single Foot Support:
  In this phase the robot is only resting on one foot while the other is in the swing phase. An illustration of this for static gait is seen in figure 1.7. During this phase the robot also moves it's COM towards the supporting foot as shown in figure 1.8.
- Double foot support:
  After the single foot support phase has ended, the robot will have placed the swinging foot back onto the ground. This then leads to the double foot support, which means that the robot is supported by both feet. This is visualised in figures 1.7, 1.8 and 1.9, as a long red line.
- Single foot support:
  This phase is similar to the first single foot support phase, although the feet have now switched roles. This results in the right foot

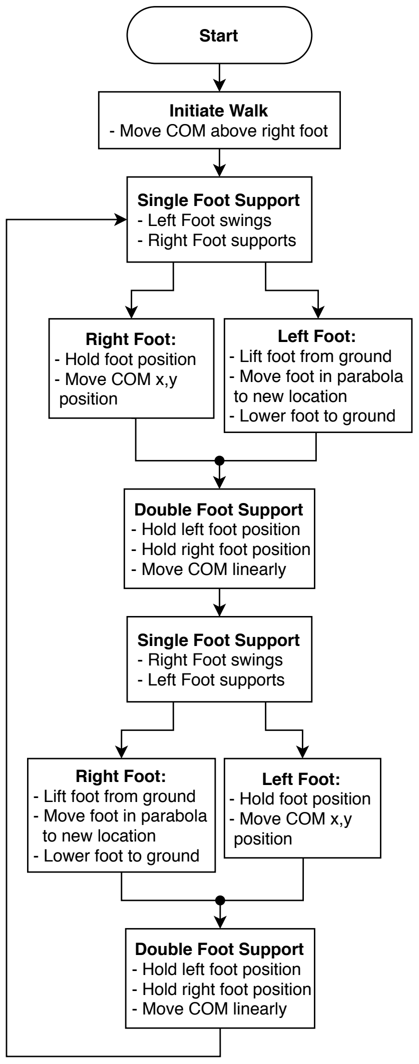


Figure 1.10 The phases of the robot gait

being the swing foot, while the left foot is the support foot.

- Double foot support:
  This phase is similar to the previous double foot support phase. The only difference between these is the direction that the COM is moved. The background for this is further explained in chapter 5 on page 46

Time-Based Gait

A time-based gait shifts between single and double foot support at fixed time intervals [19]. Say that the robot is walking, and is about to take a step. Using the time-based approach, the duration from the initiation of this new step until it's completion is predetermined. However, this duration is based on the assumption that the environment that the robot operates in is known. Minor disturbances can most likely be handled by the system, but is not always the case. Say a tile with a height of 10mm has been placed in front of the robot. If this obstacle had not been included when the trajectories were calculated, then the swing foot would come into contact with obstacle before it was anticipated. This will however not stop the robot from continuing to follow the flawed trajectory and will likely result in the robot falling, as the robot will keep pushing the foot downwards for the predetermined time. A method that seeks to improve upon this is the event-based method.

Event-Based Gait

Another gait method for biped robots is the event-based method, which is used by the LOLA robot [19]. As the name implies, the event-based method uses a gait that relies on the event that occurs when the foot strikes the ground. The event-based method draws inspiration from nature, where the strategy is found in legged animals ranging from insects, cats and etc, to humans. This method is suitable for unmodelled terrain, with debris and slopes. The event-based method uses measured system states to determine the reference trajectories for the gait, instead of the traditional time-based reference trajectories. This is typically done by using force transducers in the feet, which can be used to detect ground contact. This method makes the gait generation less time-dependant, thereby improving the systems robustness to disturbances [19]. During a gait cycle, should the foot come into contact with the ground before anticipated, this event will trigger the switching to the next phase of the gait.

Joint Actuation

Regardless of the weight of the robot the joints are to replicate the speed of human joints. As weight increases the energy required when moving increases as well. A lighter robot would reduce the energy need, thereby allowing the use of motors with less strength, thus smaller motors can be used to reduce weight. Materials such as plastic is cheaper than steel, aluminium and titanium. Plastic is also lighter at the price of less structural strength. A lighter robot is also safer to work with as the energy in the system is smaller making testing a simpler task. To imitate a human the robotic joint must be able to move at the velocities noted in table III. Being able to comply with those velocities allows the humanoid robot to perform dynamic walk with a gait speed up to 1.6 m/s [14].

Range of Motion

As human joints are flexible and have a high DOF, a humanoid robot aims at achieving same level of flexibility. Depending on the task the robot is built to perform, the degrees of freedom can differ from a human.
As this report focuses on dynamic walk, the robot aims at being able to simulate the walk of a human and therefore must adhere to the range of motion noted in table II. At best case the robot is fully capable of the range of motion noted in table I and thus being similar to a human while walking.

1.3 Problem Statement

How can a biped robot be designed to walk on a flat surface at a velocity of 0.5 *m/s* using a dynamic gait

In order to solve this problem several fields must be studied, including: mechanical design of the robot, design of control strategy and electronics and communication design. The problem statement can be broken up into a list of requirements that must be fulfilled if the problem statement is to be solved.

1.4 Requirements Specifications

The requirement specifications are based on the knowledge from the state of the art section 1.1. Here a range of different robot speeds are mentioned. Atlas with a walking speed of 1.5 m/s, Asimo with 0.75 m/s and Wabian-2R with 0.5 m/s. Using these requirements, along with the analysis of the existing biped robots, the list of requirements can be made.

Mechanical Design

- Each leg of the robot must have the following joints: 3$\times$ Hip, 1$\times$ Knee and 2$\times$ ankle.
- The joints must have the range of motion noted in table II.
- The soles of the feet must have sufficient friction to avoid slipping and thereby falling.
- The lengths of the robot's links must closely resemble a human.

Electronics and Communication Design

- A communication network must be designed to provide the required sample rates for sensors and actuators.
- The motors of the biped robot must, as a minimum, be able to achieve the angular velocities required to maintain the walking velocity of 0.5m/s.

Control Design

- Send motor velocity references at a sample rate of 2 kHz, similar to state of art [7].
- Strain gauges must be sampled at a rate of minimum 500 Hz [7].
- Sample the IMU at a rate of minimum 500 Hz [7].
- Generate trajectories that avoid tilting of the support foot during walking.

CHAPTER | 2
Hardware

This chapter will present the biped robot developed for this report. This will be divided into the mechanical and electrical design. The robot is developed with 6 DOF for each leg. These consists of 3 DOF for the hip, 1 for the knee and 2 for the ankle. The joints are actuated by stepper motors, where the 3 hip joints are rotational driven, and the knee and ankle joints are linear driven. The motors are powered and controlled by individual uStepper S motor drivers, all controlled by a single ESP32 where the the high level controller is implemented. The robot design, focuses on weight and affordability. These two factors makes the production and testing of the robot easier in the prototyping phase as alterations/errors in the design can easily be corrected.

Reducing the weight also allows for lighter motors, as less torque is needed to perform the movements. The savings on weight allows cheaper motors to be procured and components are diminished for less stress, also reducing cost.

The following section will describe the mechanical design of the robot including the choice of materials, dimensions, ROM (Range of Motion) and design of the feet.

2.1 Mechanical Design

The robot is primarily built by 3D printed components using PLA(Polylactic Acid) thermoplastic. The components are joined together using metal machine screws and ball bearings. Ball bearings and guide rails are created from aluminium and steel. The hip joints of the robot are rotationally driven, while the knee and ankle joints are linearly driven as the motors are connected with a belt to a linear push rod. The angular velocity of these joints should be as close as possible to the angular velocities from table III. This can be achieved through proper gearing of the motors. The ROM of the robot needs to be at least that of table II, in order to be able to achieve dynamic walk.

3D Printed Parts

Due to the accessibility of a 3D printer the robot was manufactured using 3D printed PLA. However for further development of the robot other alternatives should be considered. Several common materials were investigated with focus on weight, structural strength, price and ease of manufacturing. Making the parts out of steel would make the links heavy as it has a density of 7.82 g/cm^3 and metals such as aluminium, 2.70 g/cm^3, or titanium, 4.50 g/cm^3, are lighter and thus better alternatives [20]. Titanium is an expensive metal and therefore disregarded leaving aluminium as a possible material for the parts.
Aluminium is softer to process than steel and easier to manipulate, yet machining tools are still required.
Plastic is an attractive alternative, since it is often cheaper and lighter, compared to metal. ABS plastic has a density of 1.06 g/cm^3 and PLA a density of $1.15 - 1.25$ g/cm^3 [20]. These plastics can be used by a 3D printer and are often used for prototyping and lightweight part construction. ABS has a higher flexural strength, improved ductility, lighter and is cheaper, while PLA is easier to print and the most common printer material [21].
While ABS would be better for the parts, since it has higher strength, PLA was chosen as the material for the biped body parts as it was easier to print with and was available for the production of the robot.

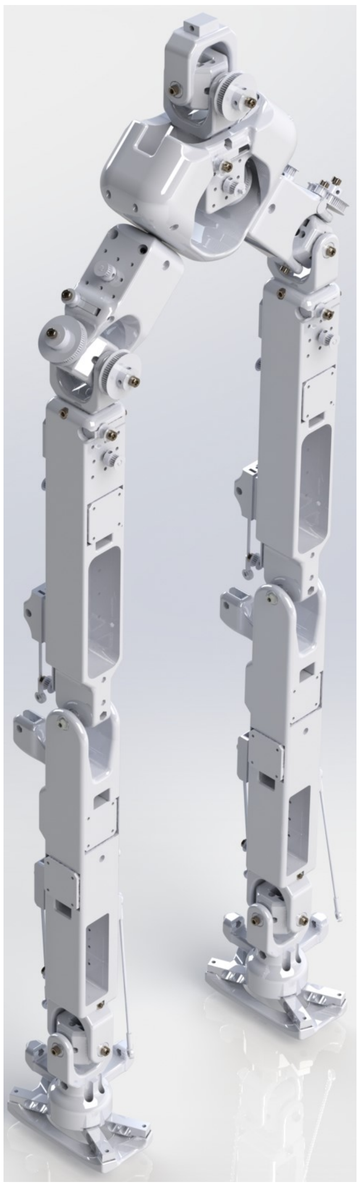


Figure 2.1 The initial configuration of the robot, where all joint angles are at 0 radians. Here the robot is shown in white for better detail. Also this model of the robot includes a spine, which could be implemented onto the robot, to test other control strategies

The rigid links and joints were printed with an infill density of 20 %, using a hexagonal pattern. The design of the parts include cutouts for the motors, batteries, slide rails and ball bearings for the joints. These cutouts can be seen in figure 2.1, which shows the full robot, consisting from the top and downwards of a spine, a hip, cardan joint, upper leg, lower leg, ankle and foot. Note the spine shown in figure 2.1 was added to the design, as it could allow for other control strategies. However this is not touched upon in this report and serves only as a design idea, meaning the rest of the report will focus on a robot without a spine. Another design choice was to have a 45° hip joint, as this will result in all 3 hip joints being used for forward motion, which will distribute the load among the joints.

The dimensions of the robot are as follows: The robot has a total height of 101.1 cm, a depth of 8 cm at the hip (excluding the feet) and a width of 32.1 cm from leg to leg.
The length of the parts were designed so as to duplicate the leg dimensions of one of the authors in a 1:1 scale making the mechanical design as humanoid as possible. The exceptions from this are the weights and the reduced amount of DOF. The total mass of the robot is $\approx$ 8 kg.

Designing the Feet and Soles of the Robot

The friction between the soles of the feet and the ground play a vital role in walking, exactly why will be analysed in section 4. This is further complicated by the fact that the robot might operate in environments with different types of friction, as the floor of one environment could be concrete while in another environment it could be dirt. In short, it can be said that the greater the friction is, the larger the set of solutions is to where the feet of the robot can be positioned during walking without slipping and falling.

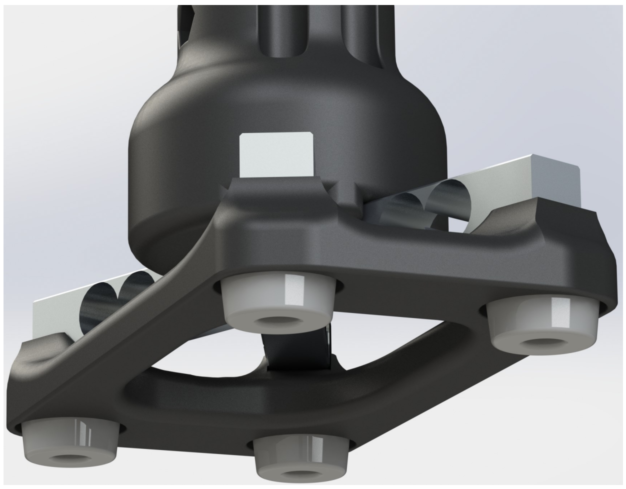


Figure 2.2 Soles of the feet of the robot, showing the rubber cylinders ensuring increased friction between the feet and the ground, here the strain gauges can also be seen as metal bars with holes in them

Therefore it is desirable to design the soles of the feet of the robot to have as great friction as possible. Another aspect is the area of the sole of the feet. As explained in section 4.1, the larger the sole of the feet is, the greater is the support polygon they form. Ultimately this again reduces the risk of falling. However, as tempting as it might be to create feet with large soles, it also comes at a cost. As the size of the feet goes up, so does the risk of collision, and thereby falling. Even when walking in an obstacle free environment, the robot risks that the feet may collide during walking.

A strategy should be devised such that the robot can recover from the feet colliding with each other or another object. However, since it is possible to reduce the problem at it's source, this is of cause preferable.
Finding the optimal size of the sole of the feet is not a trivial task, and it's determination is therefore best suited through empirical trials.
For this robot a foot area of 13.75 $cm/6.85\ cm\ (L/D)$ is chosen. The reason behind this decision is explained in section 4.1 on page 41 This foot area gives a length of 23.6 cm between the feet to minimise the chance of collision between them.
Increasing the friction between the soles of the feet and the ground can be done, as long as the chosen material does not also add stiction, or becomes too soft Additionally, adding extrusions at the corners of the feet will ensure that the contact points are known.

In summary, this elaborate the mechanical requirements in section 1.4 to include the following:

- Friction of the soles must be that between concrete and rubber, which is $\approx\ 0.6 - 0.85\ \mu_{kinetic}$, as this is the material found at the test site.
- The area of the soles must be 13.75 cm by 6.85 cm in order for the robot COM being inside the support polygon for the robots initial position, as explained in section 4.1.
- Extrusions at the corners of the feet, in order to easier model the contact points of the feet.

Degrees of Freedom

The developed robot has, without the spine, a total of 12 DOF, with 6 DOF in each leg. For each leg 3 DOFs are located at the hip for yaw, roll and pitch rotation, 1 at the knee for pitch and 2 at the ankle for pitch and roll. The location of the DOF can be seen in figure 2.3. The 6 DOF leg model as is mentioned in section 1.2 on page 5, was chosen as it is simpler to model, compared to the numerous DOF of a human leg as noted in section 1.1 on page 2. The 6 DOF model also allows the robot to make the fundamental movements of a human leg, such as yaw, roll and pitch rotation of the hip, pitch rotation of the knee and ankle and roll for the ankle. This allows the robot to place the feet at stabilising positions. The 6 DOF leg model reduces the complexity of the inverse kinematics, reduce number of motors, reduce cost and save weight. Based on the analysis in section 1.2, it is estimated that the robot designed for this project would emulate the human gait sufficiently enough to be able to perform dynamic walk.

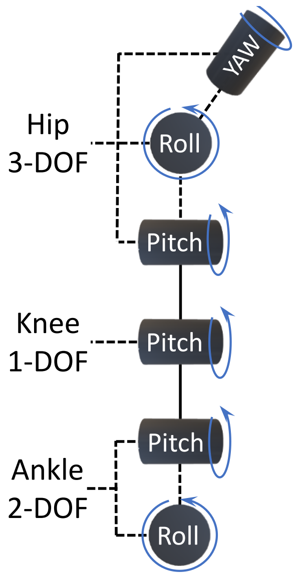


Figure 2.3 Figure showing the DOF for the robot legs seen from the anterior side. The dotted lines between joints indicates a distance of zero.

Range of Motion

As is shown in table II on page 8, the ROM of the each joint of the robot encompasses the joint ROM for human gait. The robot mostly outperforms the ROM of a human, except for knee rotation where a human is able to do 150° compared to the 90° of the robot and ankle pitch which is −50° compared to −30° of the robot. The angles can be visually seen in figure 2.4.

TABLE IV
Table listing the joints ROM from tablesI,II and the ROM for the robot, all data has been rounded to nearest whole number.

Joint	ROM Human	ROM Human gait	ROM Robot
Hip Roll	40°/ -*n*/*a*°	*n*/*a*°/ -*n*/*a*°	135°/ -90°
Hip Pitch	110°/ -30°	34°± 7°/ -11°± 5°	110°/ -110°
Hip Yaw	*n*/*a*°/ -*n*/*a*°	*n*/*a*°/ -*n*/*a*°	180°/ -180°
Knee	150°/ -*n*/*a*°	62°± 3°/ 0°	90°/ 0°
Ankle Roll	12°/ -23°	*n*/*a*°/ -*n*/*a*°	30°/ -30°
Ankle Pitch	20°/ -50°	15°± 3°/ -22°± 4°	30°/ -30°

2.2 Electrical Design

The electrical system consists of 12 NEMA 17 stepper motors, one actuating each joint of the robot. Each of the stepper motors have an uStepper S motor driver, which provides position feedback at a 0.0055° resolution [22]. The 12 motor drivers are connected by I^{2}C to an ESP32 microcontroller. In each of the two feet of the robot are 4 strain gauges. Each of the strain gauges are connected to an AD7705 ADC. The output of these ADCs are connected to the ESP32 using I^{2}C. Additionally the system is equipped with a 10 DOF GY-91 Internal Measurement Unit (IMU) which provides a 3-axis gyroscope, 3-axis accelerometer, 3-axis compass and a barometric pressure sensor. An overview of the electronic parts can be seen in table V.

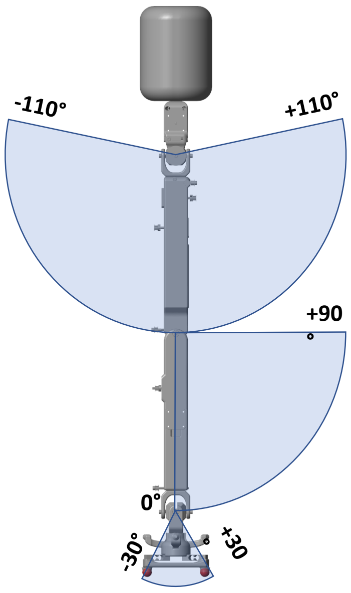


Figure 2.4 Figure showing the ROM of the robots legs.

Actuators

The requirements for the joint velocity trajectories of the robot are described in section 1.4 on page 14. In order for the joints to follow this desired joint velocity, the motors and associated gearing must be chosen accordingly. The torque produced by the actuators must be sufficient to accelerate the associated mass, such that the joint follows the velocity trajectories. For this project, stepper motors are used, as they have better positional control, compared to servo motors, which is essential when controlling a robot [24]. Additionally stepper motors offer a promising amount of torque, considering the relatively low price [25].

TABLE V
Table listing the electronic components of the robot

Hardware Part:	**Amount:**	**Description:**
Motor	12	TRINAMIC QMOT QSH4218
Motor Controller	12	Ustepper S
Gait Controller	1	ESP32 Devkit V1
Strain gauge	8	10kg Straight Bar TAL220 [23]
ADC	8	AD7705 strain gauge ADC
IMU	1	GY-91: 10DOF MPU-9250, BMP280
Battery	3	Turnigy 2200 mAh 3S 25C Lipo Pack

Stepper motors comes with a wide range of performance, depending on the size of the motor. One of the most widely used standards of stepper motors is the NEMA 17, which has a height and width of 42 *mm* [26]. The NEMA 17 motor comes with various lengths of the stator and rotor, allowing for a greater amount of motor windings which in turn gives a higher torque output. Generally it can be said that the larger the motor is, the higher torque it produces. However the cost of adding a larger motor is both increased power demand, and increased weight. Finding the correct motor is based upon the power to weight ratio and the energy supply available. The torque capabilities of the NEMA 17 stepper motors can be derived by analysing the torque/velocity characteristics of the motor model. These characteristics are determined by the combination of a given motor and associated motor driver. An additional parameter that determines the performance of the stepper motor is which stepping strategy is chosen for the motor driver. For this project uStepper S stepper motor drivers will be used [22]. The TMC5130A motor driver used on the uStepper S board comes with various modes of stepping strategies. These primarily differ in the amount of micro-stepping being used. On average, the "256uS-SpreadCycle" micro stepping strategy offers the highest amount of torque. The stepping strategy produces a maximum torque of approximately 0.33 *Nm* from 100 to 500 *rpm* [25], before the torque output begins to significantly decay due to increased velocity of the motor.

Torque Demands

The required torque needed to follow the velocity trajectories of 0.5 m/s as described in section 1.4 on page 14 must be investigated. By knowing the maximum torque requirements, it can be made sure that the actuator is capable of delivering the required torque at all times. The torque requirements are joint specific, since each joint is exposed to a different loads due to the masses. The load is also dependant on the phase of the gait cycle. There are two approaches to determine the torque requirements prior to implementation. Either it is derived by analysing the dynamic equations of the robot during walking, or it is found experimentally in a simulated environment. The CAD model of the robot can be directly imported into Simscape Multibody, which is an extension of Matlab/Simulink that provides a simulation environment for mechanical systems. Within the simulation, the actuators can be given infinite torque capabilities and during a simulated walk, the torque produced by the motors can be monitored, and the maximum torque requirements can be found.

When choosing the motor/gearing combination for implementation, it is recommended to add some overhead to the motor torque, such that the motor can handle any minor model inaccuracies.

Gearing

Gearing offers the ability to trade velocity for torque, or opposite. However, gearing also comes at additional cost such as backlash and added friction, which must be taken into account of the mechanical design. Each joint has a specific requirement for torque and velocity, therefore each joint must be individually fitted with the appropriate gearing. This can be included as part of the simulation described in chapter 7 on page 58 to verify that the appropriate gearing has been chosen.

Sensors

An important design choice is to choose the appropriate sensors for the robot. Robots such as Asimo, Cassie and Wabian-2R, which was mentioned in section 1.1 on page 2, uses an IMU sensor, which gives information about the robots torso orientation and acceleration. Additionally, they use force/torque sensors for detecting ground contact. The sensors used in this project is an IMU, angular encoders for the joints and strain gauges for the feet.

IMU

The IMU has been chosen to give a measurement of the pose and velocities of the robot in relation to the world. The position is necessary if the robot has to navigate in the world. The velocities can be used as feedback for the control strategy described in chapter 6 on page 50. The IMU used for the project is the 10 DOF Gy-91. This sensor can be sampled at 1 kHz, and has a 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer and 1-axis barometer [27]. The sample rate of the sensor is promising, as other biped robots [7] successfully use an IMU with a sample rate of 800 Hz. However, sample rate alone does not ensure satisfactory feedback. The quality of the measured IMU data, including accuracy, noise and drift is a cornerstone of successfully implementing a walking controller on a biped robot [12]. Should the system performance be unacceptable the IMU performance should be evaluated.

Angular Encoders

The angular encoders for the joints are necessary to create closed-loop control of the joint positions, described in section 6.3 on page 56. The angular encoders used for this project are the ones provided by the uStepper S motor drivers. These encoders can be sampled at 1 kHz [28]. A similar biped robot such as Cassie uses 2 kHz angular encoders. However by maintaining relatively low velocities and accelerations of the joints it is expected that 1 kHz will suffice.

Strain Gauges
The strain gauges provide a measurement of the contact force when the feet collide with the ground. This can be used for ground detection, and can be used to get a measurement of the ZMP, which is described in section 4.2 on page 43. The strain gauges are sampled using an ADC with a sample rate of 500 Hz [29], inspired by the sampling frequency of a similar system from [7]. The required sample rate of the strain gauges depends on the chosen control strategy, as described in section 1.2 on page 6.

For event-based walking control it is important that ground contact is correctly detected, which increases the demands on both the sampling rate of the strain gauges, as well as precise classification of ground contact [19]. Improved contact detection promises increased walking performance [19]. The task of correctly detecting ground contact relies on the following: First, reducing the time between ground impact happening and the event being detected. Secondly, ensuring no false positives/negatives due to noisy signals. Using a moving average filter could lower this probability. However, this would increase detection time, which means the sampling rate should be increased. If possible, a simple threshold will give the fastest detection. However some measures, such as hysteresis, must be used to counteract any ringing that might occur from the foot striking the ground.

The strain gauges used for this project can be seen on figure 2.2 on page 17. While the robot is static, the strain gauges can be used to measure the total mass of the robot. This can be useful if the robot is to carry items, since the mass of the system model used for control in chapter 6 on page 50 can then be updated on the fly. This does however entail that the location of the added mass is known.

Microcontroller board

The high-level control unit for the robot consists of an ESP32 microcontroller board [30]. The board offers up to 600 MIPS (Million Instructions Per Second), as well as the necessary I^{2}C peripherals to communicate with both the motor drivers and the sensors. The computational power of the high-level control unit has an impact on the performance of the control strategy discussed in chapter 6 on page 50. Should the ESP32 prove to be a performance bottleneck, a more powerful microcontroller should be found. Additionally it comes with built-in Bluetooth and Wi-Fi, allowing for high-level control commands such as waypoints to be send remotely. Lastly the low power consumption of the ESP32 also makes it ideal for a system relying on battery power, consuming only 250 mA at 3.3 V continuously. The ESP32 can be connected to a MATLAB GUI from which high-level control commands can be sent and data logging performed. The communication done using the Bluetooth connection to ease the setup.

Power Consumption & Supply

For early implementation a laboratory power supply and a power cable is preferable. However battery supply should be used as soon as possible since the weight of the batteries contribute significantly to the mass of the robot, and thereby the dynamics. The amount of batteries should be chosen so as to provide sufficient power and have an energy deposit at a minimum large enough to enable the robot to perform a full gait cycle [1] test on a single charge. An estimation for the battery supply was calculated based on the maximal power consumption of the motors. The supply needed for the sensors, IMU and ESP32 is disregarded as these components have a low power consumption compared to the motors. The voltage chosen for the system is 33.3 V based on the motor driver torque specifications [25].

The motor drivers have a peak current draw of 2.5 A for each of the 12 motors. Thus the robots total peak current usage can be calculated as in equation 2.1. The batteries must be able to provide a discharge rate greater than the the peak current draw of the combined 12 motors. LiPo Batteries often used for drones, have a high discharge rate, making it ideal for this application. The battery used for the following calculations is a Turnigy 2200 mAh 3S 25C Lipo battery [31]. Having 3 batteries gives a voltage of 33.3 V and a total capacity of 6600 mAh. The LiPo batteries are able to discharge with a rate of 2200 $mAh \cdot 25C = 55,000\ mA = 55\ A$ and therefore suitable for powering the robot as shown by calculating the power consumption in equation 2.1.

$$12\ motors \cdot 2.5\ A = 30\ A \tag{2.1}$$

The power consumption is based on all motors drawing full power simultaneously, which would result in a total power consumption of $30\ A \cdot 33.3\ V \approx 990\ W$. This should be considered a worst case scenario. It is unlikely that nominal performance during a gait cycle at a walking speed of 0.5 m/s will require this amount of power. While walking at a speed of 0.5 m/s, an adult human male of average weight and height consumes $\approx$ 200 W [32]. The human body should be considered far more energy efficient at walking than the biped robot shown in figure 2.1 on page 16. However, this biped robot has a significantly lower total mass of just 8 kg. The battery life of the robot using maximum power is calculated in equation 2.2. This yields the worst case performance, which can easily be scaled to the appropriate power consumption based on the simulation of the system in section 7 on page 58 and calculated in equation 2.2.

[1] a full gait cycle for this project consist of a support phase, swing phase for first leg, support phase, swing phase for second leg and then end on support phase

$$\frac{6600\ mAh}{12 \cdot 2500\ mah} = 0.22\ h \approx 13min \tag{2.2}$$

2.3 Communication

Several communication protocols were investigated for this project, among which were I^{2}C and SPI. The ESP32 supports both I^{2}C and SPI, as do the uStepper S motor drivers. The IMU and strain gauge ADCs use I^{2}C. The communication cables run along the legs to the motors and are with such length suitable to noise from external factors.

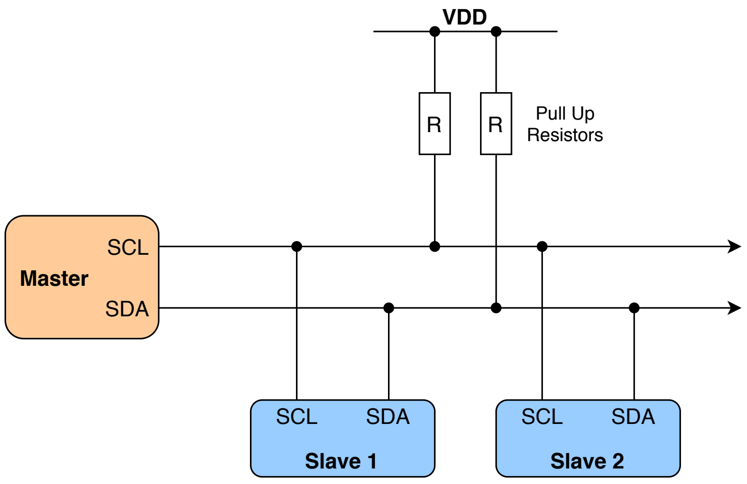


Figure 2.5 An I^{2}C network with a single master and two slaves. Pull up resistors ensure the correct line voltage

The SPI protocol is potentially faster than I^{2}C and can with SS (Slave Select) chose which unit a given message is for, and run full duplex communication. I^{2}C is slower than SPI and supports multiple devices on one bus instead of having a separate SS line, reducing the required amount of cables. An investigation into SPI, for the hardware in this project, proved it could not be used. The findings of this investigation can be read in appendix A.1.

I^{2}C was chosen as the communication protocol between the ESP32 and the uSteppers and the protocol can be seen in figure 2.5. I^{2}C is based on a master/slave principle with a two line bus connecting all units on the network. The communications lines are used with a pull up supply to VDD to ensure square signal waves as the units pulls the line low when communicating. As for the hardware used for this project, the uSteppers S has an internal pull-up power supply on the I^{2}C lines and therefore an external supply is not needed. For large scale I^{2}C networks a common ground between the connected units is necessary as I^2C is developed for embedded systems and has high impedance. By using a common ground the signals are tethered to the same zero point. An explanation of I^2C can be seen in figure 2.5. The remaining hardware is connected to the ESP32 with communication protocols as seen in figure 2.6.

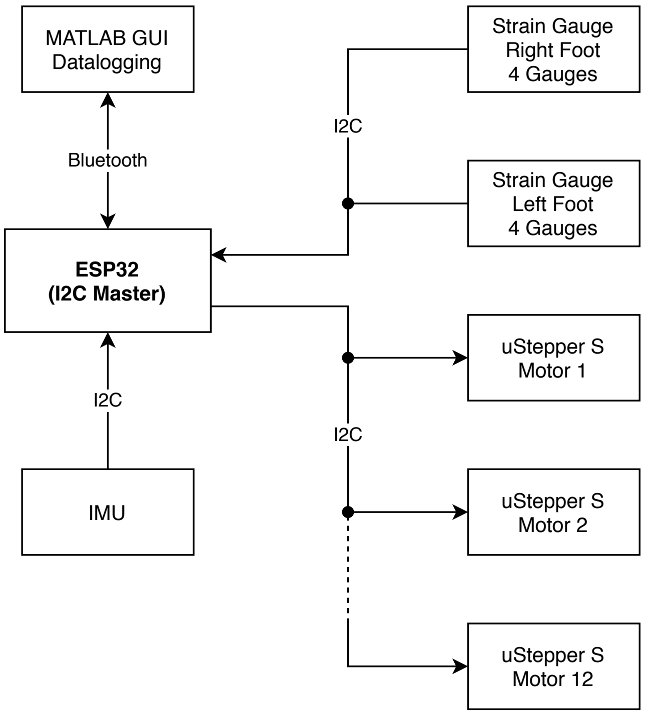


Figure 2.6 Overview of system communication

Implementation of Controller on ESP32

The robot requires a controller and for this purpose the ESP32 microcontroller board was mounted on the robot. The computational demand of the different controller types varies, where PID, specifically P, requiring very little computational power. Adding the ID terms increase the computational demand and switching to a predictive controller such as MPC greatly increases demand. Especially MPC is a heavy controller as it recalculates the control signal for the entire prediction horizon for each iteration [33]. The controller is also affected by the amount of states, with each added state increasing the computational demand [33]. As the controller is to be created in Matlab and Simulink it can be converted to C code with the built in Simulink Coder^{TM}. Simulations of the controller would result in the optimal controller parameters, however reality is often hard to simulate precisely, therefore it is expected the settings would be tuned further on the actual robot.

CHAPTER | 3
Kinematics

When modelling a mechanical system, the first step is to model the kinematics of the system. For a mobile robot such as a biped robot there must be derived both local frames for the robot and a world frame to describe the motion of the robot related to the world around it. One approach to model the kinematics is to derive the Denavit-Hartenberg (DH) parameters, which section 3.1 will cover. Once the DH parameters are found, transformation matrices that relate each of the local frames to each-other can be derived. The transformation matrices are often used to describe either the forward kinematics or the inverse kinematics. The forward kinematics describe the pose of the end-effector, given a set of joint angles. Inverse kinematics describe the possible joint angles, given a pose of the end-effector.

Inverse kinematics has been derived for both a biped robot using 45° hip yaw joints, and one using 90° hip yaw joints. This chapter will describe the derivation of the 90° inverse kinematics. The 45° inverse kinematics are located in appendix A.3 on page 86.

3.1 Denavit-Hartenberg Parameters

In order to describe the link and joint configuration of a robotic system, Denavit-Hartenberg(DH) parameters are often used. These parameters describes rotation and translation between the frames of the system. There are two different forms of DH-parameters, classic and modified DH-parameters. The difference between classic-DH and modified-DH parameters are the locations of the coordinates system attachment to the links and the order of the performed transformations. This difference is shown in equation 3.1, where:

$$\theta_i : z\ rotation \qquad \alpha_i : x\ rotation \qquad d : z\ translation \qquad r : x\ translation$$

The classic DH-parameters are described in [34] and are visualised in figure 3.1 since this approach will be used for the robot.

$$\text{Modified: } T_i^{i-1} = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & r_{i-1} \\ sin\theta_i\, cos\alpha_{i-1} & cos\theta_i\, cos\alpha_{i-1} & -sin\alpha_{i-1} & -d_i\, sin\alpha_{i-1} \\ sin\theta_i\, sin\alpha_{i-1} & cos\theta_i\, sin\alpha_{i-1} & -cos\alpha_{i-1} & d_i\, cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$\text{Classic: } T_i^{i-1} = \begin{bmatrix} cos\theta_i & -sin\theta_i\, cos\alpha_i & sin\theta_i\, sin\alpha_i & r_i\, cos\theta_i \\ sin\theta_i & cos\theta_i\, cos\alpha_i & -cos\theta_i\, sin\alpha_i & r_i\, sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1)$$

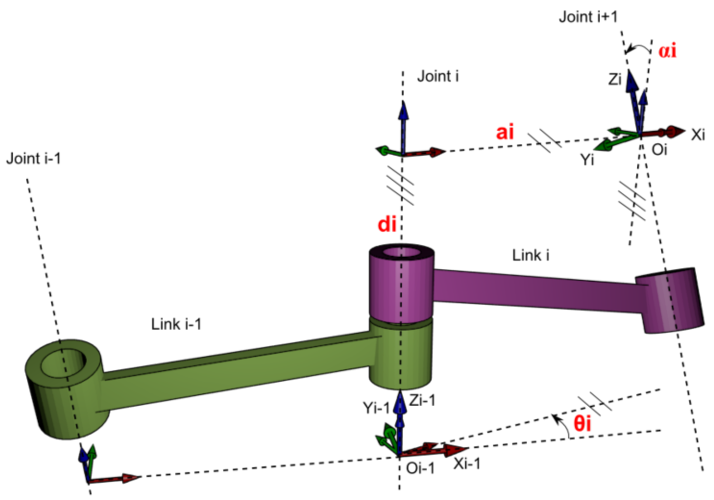


Figure 3.1 The Classic DH-parameter description [35]

Using the classic DH-parameter approach on this robotic system results in table VI. Here two DH-parameters are shown, one for each leg. The only difference between each leg is the sign change of joint 0. This joint is also fixed, meaning that they are not controlled, and are only used for achieving the correct orientation of the controlled joints.

TABLE VI
CLASSIC DH PARAMETERS FOR THE LEGS OF THE ROBOT

(A) LEFT LEG

Joint i	$r_{i-1}[m]$	$\alpha_{i-1}[Rad]$	$d_i[m]$	$\theta_i[Rad]$
0	L_1	0	$-L_2$	π
1	0	$-\pi/2$	0	$-\pi$
2	0	$-\pi/2$	0	$\pi/2$
3	L_3	0	0	θ_3
4	L_{L4}	0	0	θ_4
5	0	$\pi/2$	0	θ_5
6	L_{L5}	0	0	θ_6

(B) RIGHT LEG

Link i	$r_{i-1}[m]$	$\alpha_{i-i}[Rad]$	$d_i[m]$	$\theta_i[Rad]$
0	$-L_1$	0	$-L_2$	π
1	0	$-\pi/2$	0	$-\pi$
2	0	$-\pi/2$	0	$\pi/2$
3	L_3	0	0	θ_3
4	L_{L4}	0	0	θ_4
5	0	$\pi/2$	0	θ_5
6	L_{L5}	0	0	θ_6

Lengths L_1 and L_2 are the offset from the torso, down to either leg, followed by L_3 for upper leg, L_4 for lower leg and L_5 for foot offset from the ankle joint. The DH-parameters are illustrated in figure 3.2 on the following page, where the resulting joints and links are shown for the robot. The robot has 3 joints in the hip located at the same position, one joint in the knee and two joints in the ankle. The robot is oriented to walk in the positive y-direction when walking forward, and sideways stepping in the x-direction.

3.2 Inverse Kinematics

The inverse kinematics approach consists of the following: Given desired end-effector (which in this case are the feet) position and orientation, calculate the required angle of each joint.

The inverse kinematics problem is solved using the approach by [36], where the goal is to find the "closed-form" joint solution. A joint solution is said to be "closed-form" if the unknown joint angles can be obtained symbolically in terms of the arc-tangent function.

A closed form joint solution exists if a robot manipulator's three adjacent joint axes are parallel to one another or if they intersect at a single point [36]. The robot designed for this project has three intersecting joint angles in the hip joint.

The approach suggested by [36] handles the inverse kinematics problem of the legs of a biped robot by separating each leg into a kinematic chain of it's own. The approach then reverses the kinematic chain, such that the three intersecting joints at the hip of the leg becomes the end-effector frame of the kinematic chain. The foot then becomes the base frame. This effectively makes the last three joints of the leg intersect at the same point. According to [36] this makes it significantly easier to find a closed-form solution.

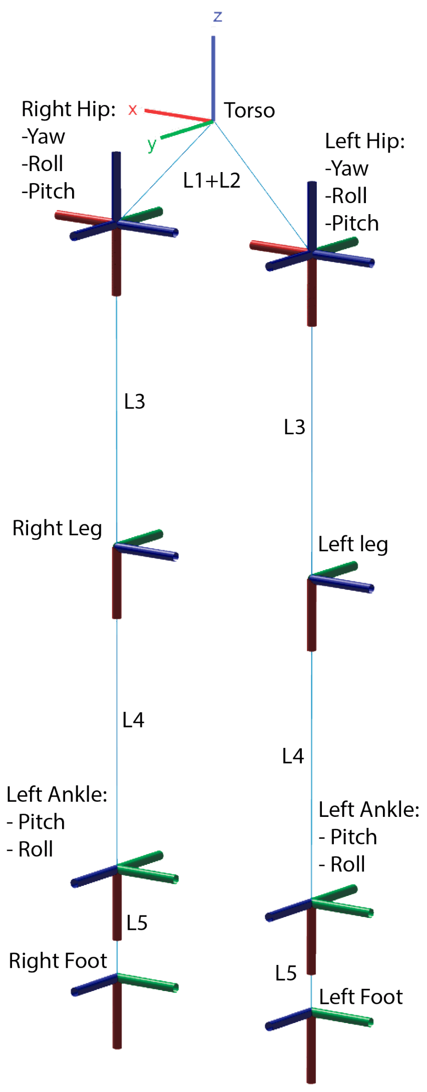


Figure 3.2 Matlab figure with links and joints, from the DH-parameters shown in table VI.

A common solution to the inverse kinematics problem for humanoid robots is the Jacobian method [36]. However that method comes with drawbacks such as: Singularities (e.g. with fully stretched limbs), computational complexity and accumulation of position error (Jacobian method is velocity based and iterative).

Using classic DH-Parameters for a robot such as the one shown in figure 3.2 on the previous page, a total of seven transformation matrices describe the transform from base to end-effector, denoted as equation 3.2

$$T_6^0 = \prod_{i=1}^{6} T_i^{i-1} = T_1^0\, T_2^1\, T_3^2\, T_4^3\, T_5^4\, T_6^5 = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

However, for the robot seen in Fig. 3.2, transformation matrix T_0^B is fixed and contains the transformation seen in equation 3.3. These are the transformations from the base(located at the centre of the hip) down to the hip-joint intersection point for each leg. The sign of $L1$ depends on which leg the transformation is used for. The right leg has negative sign.

$$T_0^B = \begin{pmatrix} -1 & 0 & 0 & \pm L_1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & -L_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

The remaining transformation matrices from the point of the three intersecting hip joints, to the end-effector, i.e. the foot, can be found as seen in equations 3.4. In these equations, L_i is the corresponding length of the robot links, as described in the DH parameters. θ_1 through θ_6 is the joint angles of the robot. A note should be made that angle θ_1 is fixed, as this is part of the fixed transform from the base to the point where the three hip joints intersect, shown in figure 3.2.
Within this section, the equations follow the notation that $cos(\theta_i) = C_i$ and $sin(\theta_i) = S_i$

$$
T_1^0 = \begin{pmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T_4^3 = \begin{pmatrix} C_4 & -S_4 & 0 & L_4\,C_4 \\ S_4 & C_4 & 0 & L_4\,S_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$
$$
T_2^1 = \begin{pmatrix} C_2 & 0 & -S_2 & 0 \\ S_2 & 0 & C_2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T_5^4 = \begin{pmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.4)
$$
$$
T_3^2 = \begin{pmatrix} C_3 & -S_3 & 0 & L_3\,C_3 \\ S_3 & C_3 & 0 & L_3\,S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T_6^5 = \begin{pmatrix} C_6 & -S_6 & 0 & L_5\,C_6 \\ S_6 & C_6 & 0 & L_5\,S_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

Now that the transformation from the base frame to the end-effector is known, it is time to reverse the kinematic chain. Once the kinematic chain has been reversed, the frame located at the feet becomes the base frame, while the three intersecting joints at the hip becomes the end-effector.
For the following equations of this section it is important to emphasize the following:
The new base frame is the frame located at the feet.
The new end-effector frame is at the point where the three hip joints intersect.

$$
T' = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} n'_x & s'_x & a'_x & p'_x \\ n'_y & s'_y & a'_y & p'_y \\ n'_z & s'_z & a'_z & p'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = T_5^6\,T_4^5\,T_3^4\,T_2^3\,T_1^2\,T_0^1 = T_0^6 \qquad (3.5)
$$

With the reversed kinematic chain, it is now possible to separate the leg into two parts, namely a part that controls the position and a part that controls the orientation of the end-effector. The position of the point where the three hip joints intersect is denoted p'. In order to obtain an expression for p', the transformation matrix T_6^1 must be inverted. This is then denoted as T_1^6 and contains the column

p'. The contents of p' is shown in equation 3.6:

$$p' = \begin{pmatrix} -C_6\,(L_3\,C_{45} + L_4\,C_5) - L_5 \\ S_6(\,L_3\,C_{45} + L_4\,C_5) \\ -L_3\,S_{45} - L_4\,S_5 \\ 1 \end{pmatrix} \tag{3.6}$$

Following the approach of [36] which incorporates Pieper's approach. Three equations are made by setting the first, second and thirds row of the p' matrix equal to p'_x, p'_y and p'_z respectively, which denotes the position for each axis.

$$p'_x + L_5 = \,-\,C_6\,(L_3\,C_{45} + L_4\,C_5) \tag{3.7}$$

$$p'_y = S_6\,(L_3\,C_{45} + L_4\,C_5) \tag{3.8}$$

$$p'_z = \,-\,L_3\,S_{45} - L_4\,S_5 \tag{3.9}$$

Squaring and adding equations 3.7, 3.8 and 3.9, and then isolating for C_4 results in equation 3.10.

$$C_4 = \frac{-L_3^2 - L_4^2 + (L_5 + p'_x)^2 + {p'_y}^2 + {p'_z}^2}{2\,L_3\,L_4} \tag{3.10}$$

Then by applying rule A.3 on page 85, S_4 can be found, as shown in equation 3.11

$$S_4 = \pm\sqrt{1 - \frac{(-L_3^2 - L_4^2 + (L_5 + p'_x)^2 + {p'_y}^2 + {p'_z}^2)^2}{4\,L_3^2\,L_4^2}} \tag{3.11}$$

Now θ_4 can be calculated by using $atan2(S_4, C_4)$, which is an arc-tangent function that returns $tan^{-1}(\frac{S_4}{C_4})$ adjusted to the proper quadrant.

$$\theta_4 = atan2\,(S_4,\, C_4) \tag{3.12}$$

By squaring equations 3.7 and 3.8 and then adding and expanding them, equation 3.13 is formed.

$$C_5\,(C_4\,L_3 + L_4) - S_5\,(S_4\,L_3) = \pm\sqrt{(p'_x + L_5)^2 + {p'_y}^2} \tag{3.13}$$

Then by expanding equation 3.9 using rule A.1 on page 85, equation 3.14 can be obtained.

$$S_5\,(C_4\,L_3 + L_4) + C_5\,(S_4\,L_3) = -p'_z \tag{3.14}$$

Note that the terms $(C_4\,L_3 + L_4)$ and $(S_4\,L_3)$ are apparent in both equation 3.13 and 3.14, and on the same form. Since both equations are on the same form, they can be rewritten by creating variables k_1 and k_2 as seen in equation 3.15.

$$\begin{aligned} k_1 &= C_4\,L_3 + L_4 \\ k_2 &= S_4\,L_3 \end{aligned} \tag{3.15}$$

Here k_1 is the first term of equation 3.10 and 3.14 and k_2 is the second term of the two equations. Another new variable r is defined in equation 3.16, which consists of the right hand side of equation 3.13 and 3.14.

$$r = \sqrt{(L_5 + p'_x)^2 + {p'_y}^2 + {p'_z}^2} \tag{3.16}$$

Another variable γ can be defined from k_1 and k_2 as:

$$\gamma = atan2(k_2, k_1) \tag{3.17}$$

Now let $(C_4\, L_3 + L_4) = rC_\gamma$ and $(S_4\, L_3) = rS_\gamma$, then by substituting them into equation 3.13 and 3.14 gives equation 3.18.

$$rC_{7\gamma} = \pm\sqrt{(p'_x + L_5)^2 + {p'_y}^2} \tag{3.18}$$

$$rS_{7\gamma} = -p'_z \tag{3.19}$$

Then by dividing equation 3.19 by 3.18, results in $tan(\theta_5 + \gamma)$, which by using rule A.4 on page 85 gives equation 3.20.

$$\frac{S_5 + atan2((L_3\, S_4), (L_4 + L_3\, C_4))}{C_5 + atan2((L_3\, S_4), (L_4 + L_3\, C_4))} = -\frac{p'_z}{\sqrt{(L_5 + p'_x)^2 + {p'_y}^2}} \tag{3.20}$$

Which can be rewritten by using equation 3.17, as equation 3.21.

$$\frac{S_5}{C_5} = -\frac{p'_z}{\sqrt{(L_5 + p'_x)^2 + {p'_y}^2}} - \gamma \tag{3.21}$$

Then from equation 3.21, θ_5 can be defined as equation 3.22.

$$\theta_5 = atan2(-p'_z, \pm\sqrt{(p'_x + L_5)^2 + {p'_y}^2}) - \gamma \tag{3.22}$$

In order to find θ_6 equation 3.8 are divided by 3.7 resulting in equation 3.23.

$$-\frac{S_6}{C_6} = \frac{p'_y}{L_5 + p'_x} \tag{3.23}$$

Then using rule A.4 on page 85, θ_6 can be derived in equation 3.24.

$$\theta_6 = atan2((p'_y), (-L_5 - p'_x)) \tag{3.24}$$

In order to find θ_1, θ_2 and θ_3 the inverse transform method, used in [37] and also applied in [36], will be applied. This approach differs from the standard approach which works by simultaneously solving all entries of equation 3.5, which is 12 equations with 7 unknowns. This approach is non-trivial since the equations are

transcendental, meaning both the sine and cosine are required to determine the angles uniquely and accurately, the robot also exhibits more than one solution for a given position.
There are, however, seven other matrix equations obtained by successively pre-multiplying equation 3.5 with the T matrix.

This method works by first creating a variable, in this case G_n seen in equation 3.25, which has as left side matrix G_n^{LHS}, and a right side matrix G_n^{RHS}. Hence G_n can be written as $G_n^{LHS} == G_n^{RHS}$, by comparing these two matrices, $\theta_1, \theta_2, \theta_3$ can be found.

$$G_{6\to1} = \begin{cases} G_6^{LHS} = G_6^{RHS} \\ G_5^{LHS} = G_5^{RHS} \\ G_4^{LHS} = G_4^{RHS} \\ G_3^{LHS} = G_3^{RHS} \\ G_2^{LHS} = G_2^{RHS} \\ G_1^{LHS} = G_1^{RHS} \end{cases} \tag{3.25}$$

The right side of G_n, called G_n^{RHS} is defined as in equation 3.26, where the red transforms are being moved over on the other side, by inverting them. This action is shown in equation 3.28.

$$\begin{aligned} G_6^{RHS} &= T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2\, T_0^1 \\ G_5^{RHS} &= T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2\, T_0^1 \\ G_4^{RHS} &= T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2\, T_0^1 \\ G_3^{RHS} &= T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2\, T_0^1 \\ G_2^{RHS} &= T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2\, T_0^1 \\ G_1^{RHS} &= T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2\, T_0^1 \end{aligned} \tag{3.26}$$

The matrix elements of the right hand sides are either zero, constants, or functions of the nth to 2th joint variables. As matrix equality implies element by element equality 12 equations from each matrix equation are obtained, that is, one equation for each of the components of the four vectors n', s', a' and p' combined in equation 3.27.

$$G_6^{RHS} = T_0^6 = \begin{bmatrix} n'_x & s'_x & a'_x & p'_x \\ n'_y & s'_y & a'_y & p'_y \\ n'_z & s'_z & a'_z & p'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.27}$$

The left side of G_n, called G_n^{LHS} is defined as in equation 3.28, these equations are functions of the elements of the inverse composite link transformation matrix T_0^6, which was shown in equation 3.5, multiplied by the inverse of the first -n joint variables of G_n^{RHS}:

$$G_n^{LHS} = G_n^{RHS\prime}\, T_0^6 \tag{3.28}$$

This is visualised as:

$$\begin{aligned}
G_6^{LHS} &= T_0^6 \\
G_5^{LHS} &= T_6^5\, T_0^6 \\
G_4^{LHS} &= T_6^5\, T_5^4\, T_0^6 \\
G_3^{LHS} &= T_6^5\, T_5^4\, T_4^3\, T_0^6 \\
G_2^{LHS} &= T_6^5\, T_5^4\, T_4^3\, T_3^2\, T_0^6 \\
G_1^{LHS} &= T_6^5\, T_5^4\, T_4^3\, T_3^2\, T_2^1\, T_0^6
\end{aligned} \tag{3.29}$$

Only G_5 is used as the remaining thetas can be found using this equation. The left side of G_5 is seen in equation 3.30.

$$G_5^{LHS} = \begin{bmatrix} n'_x\, C_6 - n'_y\, S_6 & s'_x\, C_6 - s'_y\, S_6 & a'_x\, C_6 - a'_y\, S_6 & C_6\,(L_5 + p'_x) - p'_y\, S_6 \\ n'_y\, C_6 + n'_x\, S_6 & s'_y\, C_6 + s'_x\, S_6 & a'_y\, C_6 + a'_x\, S_6 & S_6\,(L_5 + p'_x) + p'_y\, C_6 \\ n'_z & s'_z & a'_z & p'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.30}$$

The right side of G_5, called G_5^{RHS} can be seen in equation 3.31, and consists of the transformation T_0^5:

$$G_5^{RHS} = \begin{bmatrix} C_1\, C_2\, C_{345} - S_1\, S_{345} & S_1\, C_2\, C_{345} + C_1\, S_{345} & S_2\, C_{345} & -L_3\, C_{45} - L_4\, C_5 \\ -C_1\, S_1 & -S_1\, S_1 & C_2 & 0 \\ C_1\, C_2\, S_345 + S_1\, C_{345} & S_1\, C_2\, S_{345} - C_1\, C_{345} & S_2\, S_{345} & -L_3\, S_{45} - L_4\, S_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.31}$$

Then by comparing elements (2,3) of the left and right side of G_5 gives equation 3.32.

$$C_2 = a'_y\, C_6 + a'_x\, S_6 \tag{3.32}$$

Which by using rule A.3 on page 85, gives S_2 from equation 3.33.

$$S_2 = \sqrt{1 - (a'_y\, C_6 + a'_x\, S_6)^2} \tag{3.33}$$

In order to find θ_2 equation 3.33 are divided by 3.32 resulting in equation 3.34.

$$-\frac{S_2}{C_2} = \frac{\sqrt{1 - (a'_y\, C_6 + a'_x\, S_6)^2}}{a'_y\, C_6 + a'_x\, S_6} \tag{3.34}$$

Then using rule A.4 on page 85, θ_2 can be derived in equation 3.35.

$$\theta_2 = atan2(S_2, C_2) = atan2(\sqrt{1 - (a'_y\, C_6 + a'_x\, S_6^2}, a'_y\, C_6 + a'_x\, S_6) \tag{3.35}$$

By comparing the elements of $(2,1)$ and $(2,2)$ of G_5^{LHS} and G_5^{RHS}, two equations are given as:

$$S_1 S_2 = -S_6\, s'_x - C_6\, s'_y \tag{3.36}$$

$$C_1 S_2 = -S_6\, n'_x - C_6\, n'_y \tag{3.37}$$

By dividing these two equations results in $tan(\frac{S_1}{C_1})$, from which the joint solution for θ_1 can be obtained as:

$$\theta_1 = atan2(S_1, C_1) \tag{3.38}$$

Then by comparing the elements of $(1,3)$ and $(3,3)$ of G_5^{LHS} and G_5^{RHS}, result in two equations, which if divided by each other gives:

$$\frac{S_{345}}{C_{345}} \tag{3.39}$$

by which the joint angle θ_{345} is given as:

$$\theta_{345} = atan2(S_{345}, C_{345}) \tag{3.40}$$

From this the joint solution θ_5 is given as:

$$\theta_3 = \theta_{345} - \theta_4 - \theta_5 \tag{3.41}$$

Now that the robots joints can be determined given a desired position and orientation of the COM, the next step is to figure out where to place the feet, in order for the robot to walk without falling. One approach is by using Zero Moment Point, which the following chapter will look at.

CHAPTER | 4
Zero Moment Point

The Zero Moment Point (ZMP), is the point in space where all moments sum to zero. This is relevant for biped robot walk, as a crucial part of walking is to place the feet at positions that prevents falling. The strategy for placement of the feet can have an effect on many aspects of walking. These include energy efficiency, velocity, robustness etc. One method for determining where to place the feet is the ZMP approach [38]. During any motion of the robot, the feet should be positioned at points where the contact between the soles of the feet and the ground is maintained. If not, the robot is likely to fall.

For the robot depicted in figure 4.1, the sole of a foot is formed by the 4 extrusions under the foot. There are two ways that the contact between the sole and the ground can be broken, either the foot slips[1] or it tilts[2].

It is important to keep in mind that the ZMP is the sum of all moments acting upon the foot. Say you have a foot, with a single 3D inverted pendulum on top, as seen in figure 5.1. When the pendulum is in an upright position, all the forces acting on the foot are perpendicular to the ground. At this particular point, where the pendulum is truly perpendicular to the ground, the foot will not move no matter how small the friction is between the foot and the ground. However, once the pendulum moves away from the upright position, the

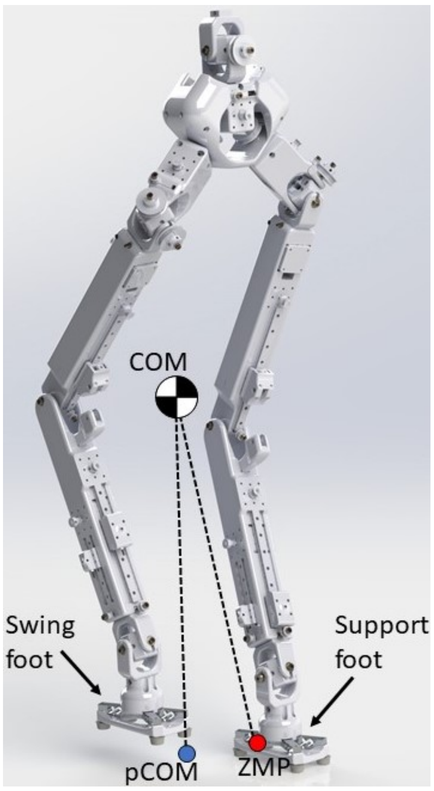


Figure 4.1 The support foot and the swing foot of the robot during walk. pCOM is the COM projected unto the ground plane

[1]Slipping of the foot is the state where the supporting foot of the robot no longer remains static

[2]Tilting of the foot is the state where at least 1 of the contact points of the supporting foot (or feet) are not in contact with the ground

combined forces acting on the foot will no longer be perpendicular to the ground. The forces will be a product of the vertical and horizontal forces. This is illustrated in figure 4.2, showing a block lying on a inclined plane. The forces acting on this block is the gravitational force F_g, the normal force F_n and the friction force F_f. The gravitational force is defined as mass times gravity $F_g = m\,g$, the normal force definition depends on the inclination of the plane, if horizontal $F_n = m\,g$ and if inclined it is determined using vector analysis, as less of the force of gravity is perpendicular to the face of the plane. Friction force F_f is parallel to the surface, and in a direction opposite to the net of applied forces [39].

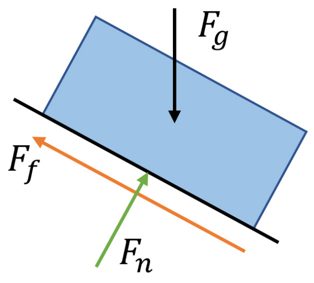


Figure 4.2 Forces included in friction. F_f = force of friction, F_h = forces of horizontal, F_g = force of gravity and F_n = normal force

The point at which the moments no longer sum to zero must be taken into account when planning the walking pattern of the robot. The further the projected COM (pCOM)[3], seen in figure 4.1, moves away from the centroid of the support polygon, the higher the horizontal forces acting on the supporting foot will be. Therefore, the step length and width, as seen in figure 4.3, must be limited to ensure that it does not create horizontal forces large enough to overcome the friction between the supporting foot and the ground.
However, determining this exact point is no trivial task, as it depends on: The combined forces that the robot affects the supporting foot with, as well as the friction between the support foot and the ground.

[3]The Projected COM is known as the point where the gravity force vector of the COM intersects with the ground

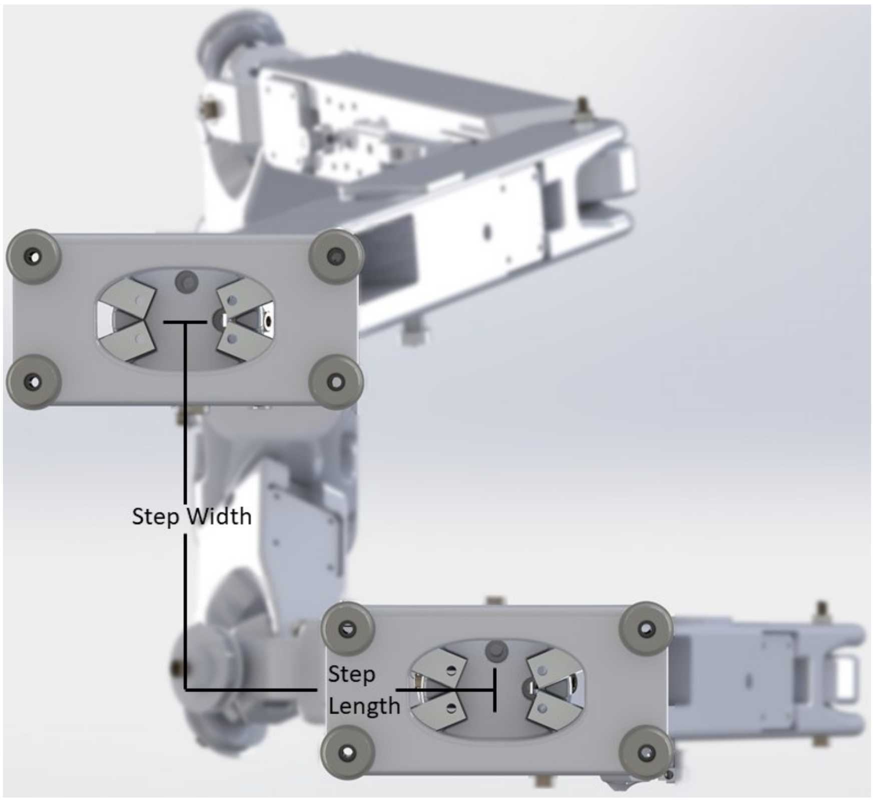


Figure 4.3 Step length and width during walking. Robot seen from bottom

4.1 The Support Polygon

A concept closely related to the ZMP is the support polygon. This is defined as the convex hull of all contact points between the sole of the feet and the ground [38]. figure 4.4 shows the support polygon during the double foot support phase of walking. The support polygon is found by taking the convex hull of ground contact points. This requires detection of the ground contact points, which is done by using the strain gauges which is described in chapter 2.

The design of the sole, described in section 2.1, ensures that when the feet is in contact with the ground and the position of the contact points are known. The convex hull is then found by checking which strain gauges reads a value above a given ground contact threshold. By using the geometry of the feet and using the kinematic configuration of the robot, the x and y coordinates of each contact point can be found.

One method for finding the convex hull of a set of points is the Jarvis march algorithm, also known as the Gift Wrapping algorithm [40]. When compared to other convex hull algorithms it is described as being simple to implement, and having favourable performance compared to other convex hull algorithms, when used for a small number of points.

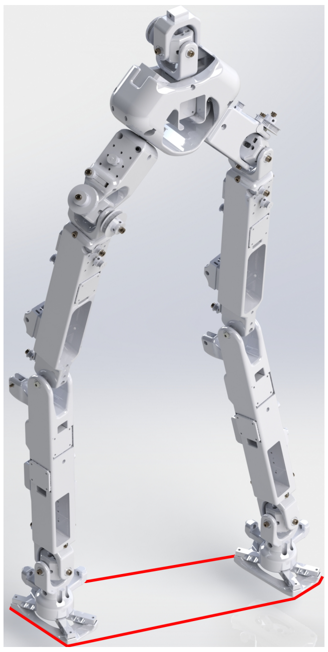
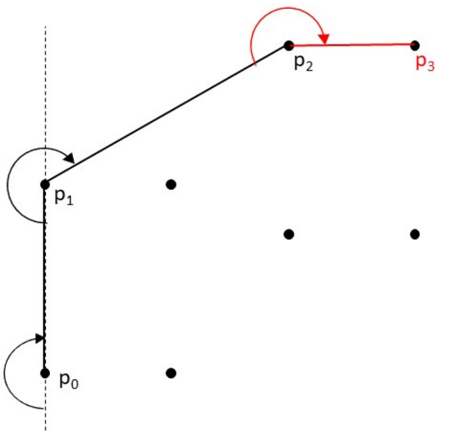
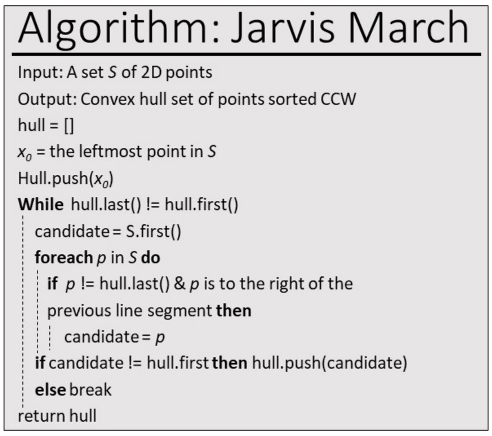


Figure 4.4 The support polygon of the robot in the double foot support phase, marked with a red polygon





```
Algorithm: Jarvis March
Input: A set S of 2D points
Output: Convex hull set of points sorted CCW
hull = []
x_0 = the leftmost point in S
Hull.push(x_0)
While hull.last() != hull.first()
    candidate = S.first()
    foreach p in S do
        if p != hull.last() & p is to the right of the
        previous line segment then
            candidate = p
    if candidate != hull.first then hull.push(candidate)
    else break
return hull
```

(a) Illustration of the Jarvis' March Algorithm processing convex hull point candidates

(b) Pseudo code describing the Jarvis' March Algorithm

Figure 4.5

An illustration of how the algorithm evaluates candidate points for the convex hull, can be seen in figure 4.5a. Pseudo code describing the overall functionality of the algorithm can be seen in figure 4.5b. The robot described in chapter 2 can be simplified by describing the contact between the feet and the ground with a maximum of 8 contact points, 4 for each foot. This simplification reduces the size of the support polygon slightly, as the contact between the sole of the feet and the ground is assumed as single points. Each of the points are located at the centre of the rubber cylinders, seen in the corners of the foot in figure 2.2. The contact points of a foot can be seen in figure 4.6 and lie at the centre-axis of the mounting holes of the rubber pads underneath the feet. The local position for the contact points of the individual feet are constant, but the position in relation to the base frame of the robot changes as the kinematic configuration of the robot changes.
As the robot moves the feet, the shape and size of the support polygon changes, but the global position of the feet change as well. The robot moves in relation to a world coordinate frame, which means that the position of the contact points, as well as the support polygon, must also be transformed to global coordinates.

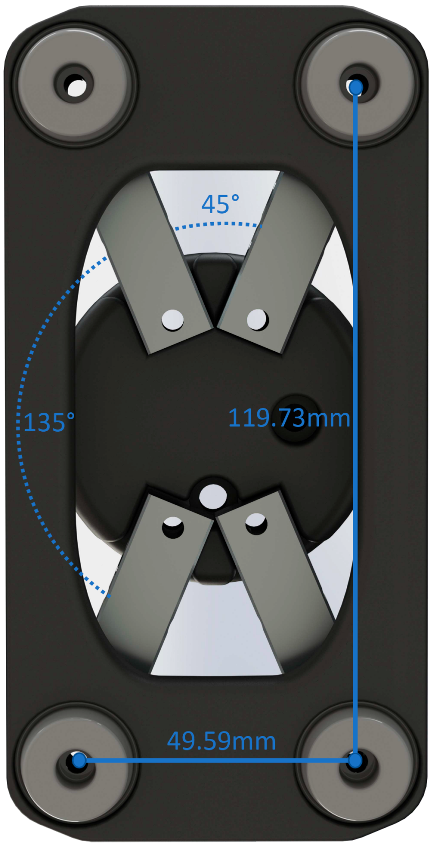


Figure 4.6 Contact points of the sole of one of the feet. Distances are marked with red.

Static vs. Dynamic Walk

Two claims can be made regarding the balance of the robot [38]:

- The ZMP must at all times lie within the boundary of the support polygon of the robot.
- The projected COM may exist outside the support polygon.

The ZMP can never exist outside the support polygon, since this would cause a non-zero sum of all forces acting on the foot, disqualifying the point from being a ZMP. If the robot is static, then a projected COM outside the boundary of the support polygon will cause the robot to fall. However, while moving dynamically, the COM may exist outside the support polygon, as long as other forces cause the sum of forces acting on the foot to become zero, i.e. as long as a ZMP exists [38]. One example of such a dynamic activity is running, where the COM is moved outside the support polygon in the running direction, in order to move the robot at a higher velocity. Here a ZMP still has to lie withing the support polygon, while the COM does not. Within this report, the concept of "Static Walk" will be defined as walking while at all times keeping the projected COM within the boundary of the support polygon. "Dynamic Walk", or just "walk", will include trajectories where the projected COM exists outside the boundary of the support polygon, while a ZMP exists. "Falling" will therefore be defined as the case where a ZMP does not exist.

4.2 Measuring the ZMP

For the 2D case, the vertical and horizontal forces acting on the distribution of the foot, can all be represented by an equivalent force and moment, placed at a point p_x in the sole of the foot. For the case, where the moment $\tau(p_x)$ about the point p_x sums to zero, the ZMP can be determined as equation 4.1, which is also described by [38] as the centre of pressure. p_x is the ZMP. x_1 and x_2 are the boundary points of the 2D sole, within which the ZMP may exist. $\rho(\xi))$ is the vertical force component which is the ground reaction force.

$$p_x = \frac{\int_{x1}^{x2} \xi \rho(\xi) d\xi}{\int_{x1}^{x2} \rho(\xi) d\xi} \tag{4.1}$$

The ZMP and pressure distribution across the sole of the foot is closely related. When the pressure is equally distributed across the foot, the ZMP is found at the centre of the foot. If the robot leans forward, the pressure distribution shifts towards the front of the foot, and the ZMP moves forward as well. By the notion that a ZMP is the point where all moments acting on the foot must sum to zero, it can be assumed that numerous ZMPs exist within the boundary of the sole. These points together form a region of ZMPs. Using ZMPs close to the boundary of the support polygon should be done with caution. The closer the ZMP lies to the boundary of the support polygon, the less robust the robot will be to external disturbances. However, increasing the velocity of the COM in the horizontal plane will move the ZMP closer to the boundary, so robustness comes at a cost.
For the 3D case, the ZMP is called p, and consists of the two coordinates (p_x, p_y). These can be found using equation 4.2, assuming that the moments about p are zero [38], i.e. $\tau_n(p) = [\tau_{nx} \ \ \tau_{ny} \ \ \tau_{nz}]^T = [0 \ \ 0 \ \ 0]^T$.

$$\begin{aligned} p_x &= \frac{\int_S \xi \rho(\xi, \eta) dS}{\int_S \rho(\xi, \eta) dS} \\ p_y &= \frac{\int_S \eta \rho(\xi, \eta) dS}{\int_S \rho(\xi, \eta) dS} \end{aligned} \tag{4.2}$$

Here the position of the ZMP is defined by a position vector $r = [\xi \;\; \eta \;\; 0]^T$ which is a translation from the world frame to the ZMP of the foot. With no moments about the ZMP p, it effectively becomes the centre of pressure [38]. This leads to an elaboration on the definition of the ZMP, which states that for 3D: **The ZMP is the point where the moment produced by the horizontal ground reaction forces is zero.** [38]

ZMP for Single Foot Support

Each foot of the robot is equipped with 4, 1 - dimensional force sensors as described in section 2, and shown in figure 4.7. The force acting on the strain gauges depends on the configuration of the robot. If the robot stands in the standard configuration, as seen in figure 2.1, then the signal values of all strain gauges are approximately equal. If the robot begins to lean forward, the force acting on the forward facing strain gauges will increase, while the force acting on the backward facing strain gauges will decrease.
Calculating the ZMP based on measurements is done for either the single foot support phase, or double foot support phase. For the single foot support phase, 4 of the force sensors are in contact with the ground. The location of the ZMP can then be calculated using equations 4.3, where the x and y components of forces acting on the foot has been set to zero, leaving only the forces in the z axis. In this equation N is the number of force sensors in the foot, which in this case is 4. p_x, p_y are the coordinates of the ZMP. Force sensor j is affected by the force f_{jz}.
p_{jx} and p_{jy} are the coordinates at which each of the forces are measured. Therefore, by measurement of the forces, and knowledge of the location of each force sensor, the ZMP can be found.

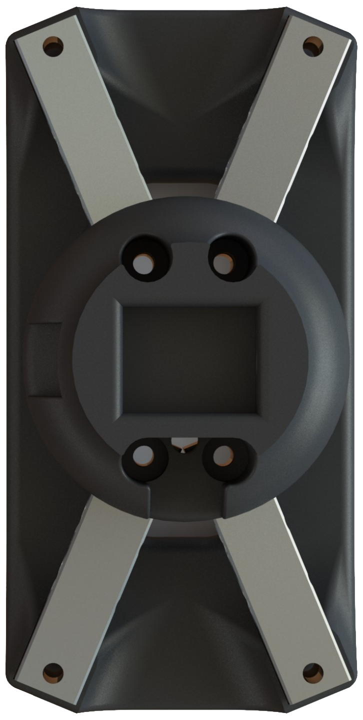


Figure 4.7 One foot of the robot showing the four strain gauges. The strain gauges are placed on a common circle with offsets of 45°in the top and bottom, and 135°at each side

$$p_x = \frac{\sum_{j=1}^{N} p_{jx} f_{jz}}{\sum_{j=1}^{N} f_{jz}} \qquad p_y = \frac{\sum_{j=1}^{N} p_{jy} f_{jz}}{\sum_{j=1}^{N} f_{jz}} \tag{4.3}$$

ZMP for Double Foot Support

The double foot support phase is an intermediate phase between taking a new step, where both feet are in contact with the ground. Contact is determined by measurements of the strain gauges. The procedure for calculating the ZMP for double foot support requires that the ZMP is calculated for each individual foot first, following the procedure described in section 4.2 on the preceding page. Once a ZMP has been found for both feet, the combined ZMP can be found. Equations 4.4 have been derived by [38], and describes how to calculate the x and y coordinate (p_x, p_y) of the combined ZMP. Equations 4.4 are based on the calculated ZMP for the right foot p_{Rx}, p_{Ry}, the left foot p_{Lx}, p_{Ly} as well as the force measurements f_{Rz}, f_{Lz} from each of the feet.

$$\begin{aligned} p_x &= \frac{p_{Rx} f_{Rz} + p_{Lx} f_{Lz}}{f_{Rz} + f_{Lz}} \\ p_y &= \frac{p_{Ry} f_{Rz} + p_{Ly} f_{Lz}}{f_{Rz} + f_{Lz}} \end{aligned} \tag{4.4}$$

Tilt Detection

A critical state, that should be avoided at all cost, is when the robot balances on the edge of the feet. This can be detected when the strain gauges in the supporting foot (or feet for double foot support) has signal values of zero. Although it is recommended to avoid this state, a control strategy should be developed to recover from this state, since a disturbance could push the robot into this state.

The analysis of the ZMP shows why it is an ideal method for choosing references for the position of the feet of the robot. In practice, positions for the feet are chosen, and then the rest of the body must be moved such that these points become and remain ZMPs during movement. Moving the rest of the robot body is no trivial task. While taking a step, such as figure 4.1, a chain of 12 joints and 8 bodies are attached at the supporting foot. Should an upper body be added to the robot in the future, the task becomes even more challenging. A complex problem such as this calls for simplification. For a majority of biped robots, the model used for control design is simplified using methods such as the LIPM (Linear Inverted Pendulum Model), which is a model of the behaviour of the linear inverted pendulum(LIP). A method for doing so, will be introduced in chapter 5.

CHAPTER 5
Linear Inverted Pendulum Model

This chapter will dive into the LIPM simplification method, and explain how it can be used for motion control of a biped robot. Similarly to humans, the biped robot walk consists of double legged and single legged periods. One approach to model a biped robot during a single legged period, is to follow the LIPM approach [41]. Here the biped walking robot is modelled as a 3D LIP with a telescopic arm, as illustrated in figure 5.1. The COM of the LIP corresponds to the combined COM of the whole robot. The telescopic arm of the pendulum is attached to the floor at the point of the supporting foot. The other foot of the robot is in a swing phase, as shown in figure 4.1. It is assumed that the supporting foot does not slip. The arm that connects the COM to the floor has no mass. The LIP is constrained to movements in the xy-plane, which is why the telescopic joint must be added to the arm of the LIP. If not, then the LIP would be restricted to movements in a circle on the xy-plane. Since the COM is only allowed to move in the xy-plane, the z coordinate i.e. the height of the COM is fixed.

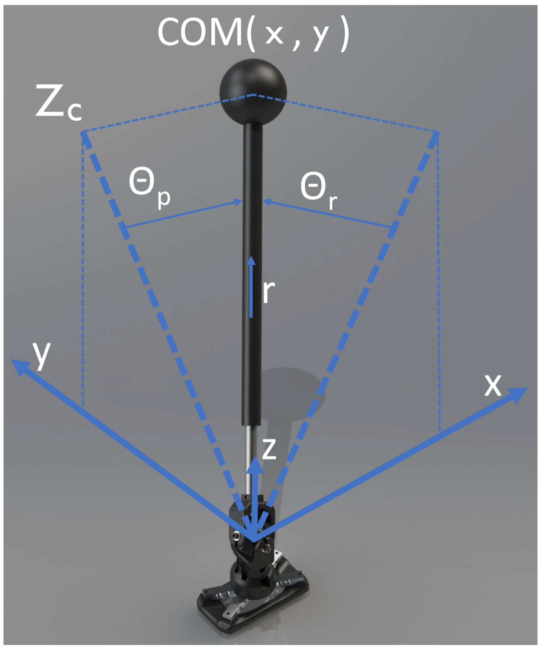


Figure 5.1 The biped robot simplified as an inverted pendulum with a telescopic joint, attached on top of single foot. The entire mass of the robot is represented by a point mass located at the centre of the COM sphere

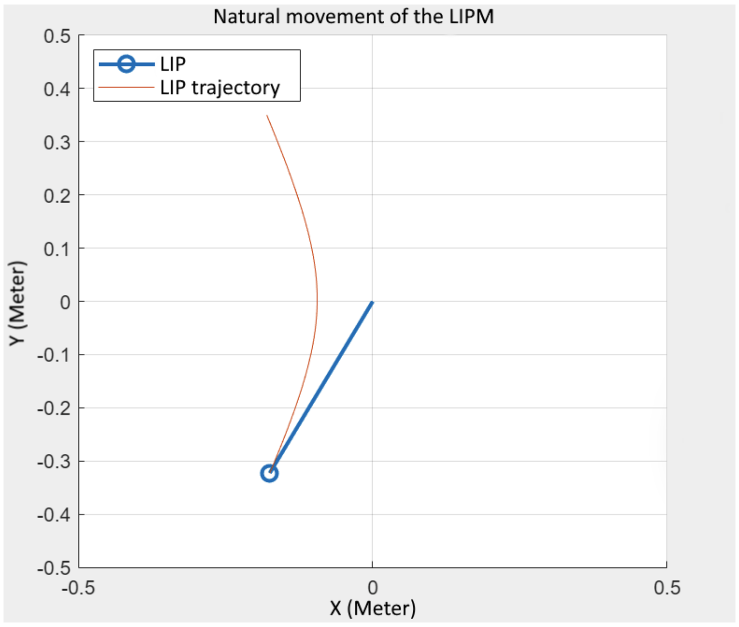


Figure 5.2 Top view of the LIP, and it's recorded natural trajectory. Initial conditions are: $x_0 = -0.08$, $y_0 = 0.35$, $\dot{x}_0 = 0.48$, $\dot{y}_0 = -1.28$

In figure 5.1 the variables x, y and Z_c describe the position of the COM of the LIP, and can be found using equation 5.1. This position can be found using the two angles θ_r, θ_p of the pendulum, as well as the length L of the telescopic arm.

$$\begin{aligned} x &= L\, sin\, \theta_p \\ y &= L\, sin\, \theta_r \\ z_c &= L\sqrt{1 - sin\, \theta_p{}^2 - sin\, \theta_r{}^2} \end{aligned} \tag{5.1}$$

The unactuated, or natural, movement of the LIPM is given as equation 5.2. An example of this movement can be seen in figure 5.2. Here the initial velocity of the LIP is nonzero. Since the LIP is constrained to movements in the XY plane, the acceleration in the z direction becomes zero. Once the swing foot connects with the ground, the base of the LIP instantaneously changes position to that of the new support foot.

$$\begin{aligned} \ddot{x} &= \frac{g}{z_c}x \\ \ddot{y} &= \frac{g}{z_c}y \end{aligned} \tag{5.2}$$

The actuated movements for the LIP are given as seen in equation 5.3. Here m is the total mass of the system, g is gravity and u_x, u_y are virtual inputs designed to linearize the LIPM [41]. The virtual inputs are given in equation 5.4, where τ_r and

τ_p are the torque about the x and y axis respectively.

$$\begin{aligned}\ddot{x} &= \frac{g}{z_c}x + \frac{u_x}{mz_c} \\ \ddot{y} &= \frac{g}{z_c}y - \frac{u_y}{mz_c}\end{aligned} \tag{5.3}$$

$$\begin{aligned}u_x &= \frac{\tau_r\sqrt{1 - sin\theta_r{}^2 - sin\,\theta_p{}^2}}{cos\,\theta_r} \\ u_y &= \frac{\tau_p\sqrt{1 - sin\,\theta_r{}^2 - sin\,\theta_p{}^2}}{cos\,\theta_p}\end{aligned} \tag{5.4}$$

Figure 5.3 shows the trajectory of the LIP during walking. The robot is travelling in the positive y direction, with a symmetric trajectory about the 0 value of the x axis. The robot starts the walk on the left foot. The initial conditions of the walk are found through study of the orbital energy of the LIPM.

5.1 Orbital Energy of the LIPM

When standing at rest, in the standard configuration as shown in Figure 2.1 on page 16, the robot has the COM between the two feet. Say we desire for the robot to move forward. When taking a step only one of the legs support the weight. This means, that when pushing forward, the COM is pushed both along the y-axis direction, i.e. forward, but also along the x-axis direction, i.e. sideways. Each new step contributes with force to the COM in both the y and x direction. Therefore, for each new step, the COM is pushed across the sagittal plane, which in this case is found at the 0 value of the x-axis in figure 5.3 . The energy of the COM, described by [41] as the orbital energy, can be found using equation 5.5. These equations are obtained by integration of equation 5.2.

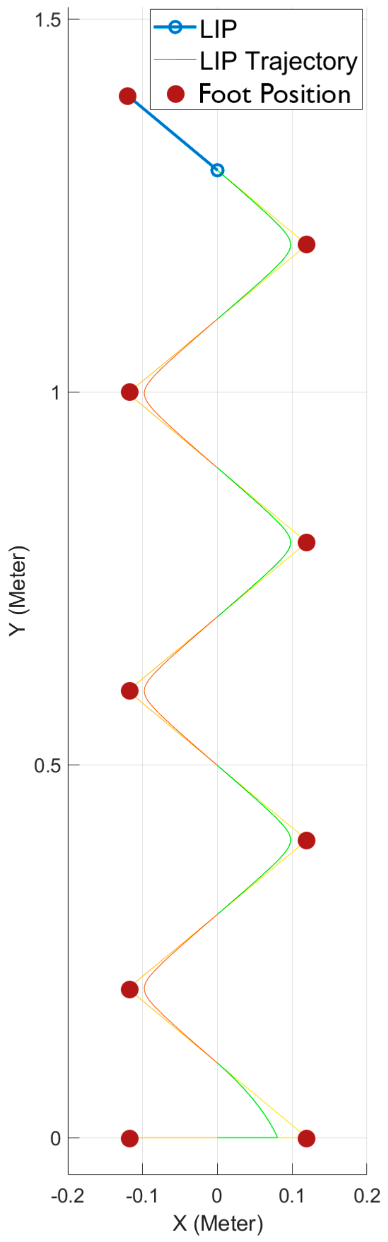


Figure 5.3 Topview of the trajectory of the LIP movement during walking. The green line marks the trajectory during the right-foot support phase, the red line marks the left-foot support phase. The red dots mark the positions of the support foot

$$\begin{aligned} E_x &= -\frac{g}{2\,z_c}x^2 + \frac{1}{2}\dot{x}^2 \\ E_y &= -\frac{g}{2\,z_c}y^2 + \frac{1}{2}\dot{y}^2 \end{aligned} \tag{5.5}$$

Looking at the bottom of figure 5.3 the first two red dots account for the initial steps, while the following steps are the consecutive steps of the walk. The initial steps should be regarded as a special case, designed to bring the LIP up to the initial velocities required for the consecutive steps. In figure 5.3 during the initial steps the LIP is moved towards the right to bring the projected COM within the support polygon of the right foot, allowing the robot to take the first step with the left foot without falling. This first step is half the length of the desired step length, in order to get the initial velocities, which in the ideal case results in symmetric trajectory for the consecutive steps. For asymmetric trajectories, more robust techniques should be investigated. However this will not be touched upon in this report, but will remain a subject worthy of further investigation.

Finding the initial conditions for the symmetric consecutive steps can be done by using the orbital energy of the LIP. The equations for the orbital energy of the LIP can be seen in equation 5.5. Here the energy of the LIP is described in the x- and y-axis, and is denoted E_x, E_y. Gravity acceleration is g, z_c is the height of the LIP. x, y is the position of the LIP, while $\dot{x}$, $\dot{y}$ is the velocity of the LIP. As figure 5.3 shows, halfway through a step the x-velocity of the LIP $\dot{x}$ is equal to zero as the LIP movement changes x-direction.

CHAPTER | 6

Predictive Control

Different types of control strategies are used in order to move, in this case, legs. For humans the brain calculates a trajectory for the legs and sends signals to the muscles in order to contract the muscles, thereby making the leg follow the trajectory and resulting in dynamic walk. For robots, which are controlled by a series of motors acting as the robot muscles, a reference is required. This is usually either velocity, position or torque and a controller to make the motors follow the reference. Among commonly used controllers are PID, LQ and MPC. Each controller has its own set of advantages and disadvantages and depending on the system the most suitable controller is chosen. The humanoid robot controller should have a sufficient controller sampling rate to follow a gait reference. In other words, it should command the motors at a sufficient enough rate to prevent the robot from falling. Plainly the controller sampling rate affects how good the controller is at following the reference. However while high frequencies allow for better reference tracking, the cost is increased computational power and could amplify controller gain errors. If the controller is too aggressive each control iteration would fight with the next to be as close to the reference as possible, resulting in excessive motor actuation and wasting energy.

State of art humanoid robots such as Atlas use predictive control to manipulate the system, specifically Atlas uses MPC. The use of predictive control has the advantage of finding the optimal inputs for a finite horizon, based upon a system model of the robot. It has the ability of anticipating future events and act accordingly, to make a more smooth walk unlike PID which can only control the current instant. As the report focuses on creating a biped robot capable of competing with the state of art the most obvious controller would be a predictive controller.

A promising method of biped walking control consists of the ZMP of chapter 4 and LIPM of chapter 5 combined with preview control [42]. This method of the combination of the ZMP and LIPM approaches aim to remove some of the drawbacks of both methods. Using the ZMP approach puts high torque demands on the actuators of the ankle, in order to keep the feet at exact positions. The LIPM does not suffer from the same drawback, however when using the LIPM the position of the feet is not directly controlled [42]. This makes the LIPM ill-suited for applications where the exact position of the feet is important, such as walking in terrain with obstacles where the feet must be placed at specific points. By using the method that mixes the ZMP and LIPM approach, arbitrary foot positioning becomes possible [42].

The LIPM presented in chapter 5, although useful, remain an approximation of a dynamic model of the robot during walk. One of the causes of model inaccuracy is the approximation of the COM position. A simple method used for positioning the COM for a full body humanoid robot is to assume it is located at the centre of the pelvis link [42]. However the COM position depends on the configuration of the limbs of the robot. If significant enough, the COM position error can cause the robot to fall, as the ZMP calculated by the simple LIPM model may not be an actual ZMP. A method of handling this model inaccuracy is by preview control [42]. In order to do so, knowledge is required of the expected ZMP error between the LIPM and the multibody model, this error is in [42] found by comparing the ZMP reference against the real ZMP measured from the dynamical system. The expected future ZMP error is then stored in a memory buffer. This knowledge can then be used by a preview controller to compensate for the errors caused by the model inaccuracies of the LIPM [42]. For implementation of the preview controller, it should be noted that the necessary amount of compensation and size of preview horizon correlate. A practical example of this is when the robot walks on what is assumed to be a flat ground, but it actually has a slope. This will cause an error in the estimated ZMP when compared to the measured ZMP which the predictive controller must compensate for.

The first step to apply the preview control method proposed by [42], is to rewrite the LIPM equation 5.3 on page 48 such that the ZMP becomes the output, this is done by first merging it with 5.4. This yields equation 6.2 which, given position of the COM x, y, z_c, gravity g and accelerations of the COM $\ddot{x}, \ddot{y}$ results in the ZMP.

$$\begin{aligned} \ddot{x} &= \frac{g}{z_c}\left(x - u_x\right) \\ \ddot{y} &= \frac{g}{z_c}\left(y - u_y\right) \end{aligned} \tag{6.1}$$

Then isolating for u_r and u_p by dividing with $\frac{g}{p_z}$ and moving p_x and p_z to the opposite side of the equal sign, and finally multiplying both sides with -1:

$$\begin{aligned} u_x &= x - \ddot{x}\frac{z_c}{g} \\ u_y &= y - \ddot{y}\frac{z_c}{g} \end{aligned} \tag{6.2}$$

The next step is then to reformulate equation 6.2 as a servo control problem on state space form [42].

First, two new variables named $\dddot{x}, \dddot{y}$ are created, which are the time derivative of $\ddot{x}, \ddot{y}$ respectively. The third derivative of position is often referred to as jerk and is the rate of change of acceleration [43]. The system dynamics are formulated on state space form, as shown in equation 6.3. Here u_x, u_y is the x and y coordinate of the ZMP. Z_c is the height of the COM of the robot and g is gravity acceleration. The system dynamics are based on equation 6.2 and shown in equation 6.3.

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dddot{x}, \quad & \frac{d}{dt}\begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dddot{y} \\ u_x &= \begin{bmatrix} 1 & 0 & \frac{-z_c}{g} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}, & u_y &= \begin{bmatrix} 1 & 0 & \frac{-z_c}{g} \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} \end{aligned} \tag{6.3}$$

A walking pattern generator can be made using the ZMP system dynamics in equations 6.3 [42]. Essentially the walking pattern generator will generate trajectories for the COM of the robot, such that the ZMP references are realised. An important observation made by [42] is that the COM of the robot must start to move before taking a new step. The same behaviour can be experienced in human gait. Here a new step is initiated by first leaning the torso slightly forward, before the swing foot is lifted from the ground. However, since the COM trajectory is based on ZMP references, and since the COM must move before a new ZMP reference is received, then the output is calculated not from the current input but from the future input. Calculating current output based on future input calls for application of preview control. Numerous control strategies exists that apply preview control, notable mentions among these are the Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC).

6.1 LQR

LQR is an optimal control strategy that seeks to minimise the cost of operating a dynamic system. LQR has been successfully implemented to generate trajectories for the COM of a biped robot [42]. LQR makes use of a quadratic cost function as the one seen in equation 6.4. Here $x(k)$ is the system state, Q_1, Q_2 are weighting matrices, and u_k is the input signal. The LQR problem is solved either with an infinite or finite horizon. For the infinite horizon LQR, it can be said that the solution is globally optimal.

$$\mathcal{I} = \sum_{k=0}^{\infty} x(k)^T Q_1 x(k) + u(k)^T Q_2 u(k) \tag{6.4}$$

The optimal controller is described using equation 6.5 where the gain $L(k)$ is found by solving the discrete time algebraic Ricatti equations.

$$u(k) = -L(k)x(k) \tag{6.5}$$

However, a drawback of the LQR control strategy is that although an optimal controller can be found, the gain $L(k)$ is calculated prior to execution, were it is run once and then calculates the desired control for the plant for all future iterations. This means that should model errors exist, or external disturbances affect the system, the gain will no longer result in optimal control. Since it only runs once there is no error correction on the plant thus robustness is low.

6.2 MPC

Unlike LQR, MPC recalculates the gains for each new sample of execution. The downside to this, is that it significantly increases the computational requirements to execute the algorithm. For embedded systems, where computational power is often scarce, MPC should be considered with caution. The computational requirements increase with the number of system states, as well as the size of the horizon.

The MPC uses a fixed horizon, the required length of which is determined by system dynamics and expected disturbances. Since the MPC algorithm is executed for each new time step, the horizon is said to be receding. Since the horizon is finite, it can only be guaranteed that the MPC provides a locally-optimal controller, in the sense that it is only guaranteed optimal for the given time horizon.

As is also the case for LQR, MPC is optimised with respect to a quadratic cost function, such as in equation 6.6. [44] Here the tune-able parameters are $Q(i)$ and $R(i)$ which are weighting matrices that specify the cost of particular states and inputs. Additional tune-able parameters are H_p, H_u and H_w which are the prediction horizon, control horizon and window parameter respectively. Notice that the mentioned parameters do not depend on time. $\hat{z}(k+i|k)$ is the measured system output, $r(k+i|k)$ is the reference signal and $\Delta\hat{u}(k+i|k)$ is the input. The Δ of the input implies the use of integral action to combat steady state error.

$$V(k) = \sum_{i=H_w}^{H_p} ||\hat{z}(k+i|k) - r(k+i|k)||^2_{Q(i)} + \sum_{i=0}^{H_u-1} ||\Delta\hat{u}(k+i|k)||^2_{R(i)} \tag{6.6}$$

When choosing the prediction horizon H_p, the size of it should be larger than the slowest dynamics of the system. Looking past the start-up phase, and walking at a constant pace, the biped gait repeats the same trajectories over and over. This may draw to the conclusion that a prediction horizon corresponding to a single gait cycle could be sufficient. However, should the robot diverge from nominal performance by encountering a disturbance, it may take numerous steps to recover, if possible at all. This is similar to human walk, where an obstacle can cause a person to stumble, which often requires several steps to recover from. Changing the length of the prediction horizon would create a increasingly stable system with only downside being the computation time, the horizon should be as large as the computational system can handle. At some point increasing the horizon would no longer affect the optimality of the system and only increase computational power, then the horizon should be shortened. Tests in the Simulink environment would indicate the turning point for performance/computational power use.

MPC can also take into account the constraints of the system states and input [44]. The constraints are linear inequalities as seen in equation 6.7. These represent the constraints on the actuator slew rate[1], actuator range and constraints on the manipulated variable.

$$E \begin{bmatrix} \Delta\mathcal{U}(k) \\ 1 \end{bmatrix} \leq 0, \quad F \begin{bmatrix} \mathcal{U}(k) \\ 1 \end{bmatrix} \leq 0, \quad G \begin{bmatrix} \mathcal{Z}(k) \\ 1 \end{bmatrix} \leq 0 \tag{6.7}$$

System Constraints

The system can be constrained with joint limits like the values from table IV on page 19. The maximum joint velocity and acceleration could be inserted as constraints in the controller with weights indicating the amount of softening for each individual constraint. Hard constraints can also be used, for example limiting a joint angle.

Constraint Softening

The robot has multiple physical constraints, such as joint limits and slew rate of the actuators. These are considered hard constraints, as the robot cannot physically violate them. Setting a constraint as a hard constraint, means that the solution must at all times satisfy the constraint. If this is not possible, the solution becomes infeasible [44]. Even if a solution is found that is feasible, disturbances could push the system past the hard constraint, which could result in a loss of system control [45].

[1]Slew rate is rate of change. Max slew rate is max rate of change

For those constraints where it is possible, this leads to the desire for soft constraints. Setting a constraint as soft, allows the constraint to be violated, without causing a solution to become infeasible [44]. Not all constraints are suited for softening. However, constraints such as velocity of the COM is suitable for softening. Say it is desirable for the COM of the robot to move at a specific velocity. Should a situation arise, where a feasible solution that fulfil the velocity demand is not found, then it is unlikely that a catastrophic failure occurs. On the other hand, should a situation arise where the MPC produces a trajectory that violates a joint limit of the robot, then it is crucial that the solution is deemed infeasible.

Soft constraints should be used with caution, as some system dynamics may be hard constrained in practice. Examples of such could be saturation of motor torque, or limited range of motion of the joints. On the other hand, hard constraints should also be placed with caution, as they decrease the size of the solution space. Placing unnecessary hard constraints can result in discarding solutions that could otherwise have saved the robot from a fall. Having several hard constraints may cause conflicts and result in reduced working space or complete failure.

The constraints for the proposed robot are based upon the states from the LIPM state space model and are; COM position, COM velocity and feet positions. The states are further explained in section 7.2 on page 69. The COM velocity can be soft constrained with the peak gait velocity of 1.6 m/s The positions can also be constrained if deemed necessary. The position constraints can set limits for how long steps the biped robot is allowed to take, and where the COM position is allowed to swing. These limits could be useful if feet slippage proves to be a problem when taking steps above a certain length.

6.3 Position Control of Joints

The simple system model used for MPC does not handle control of the individual joints of the robot, but solely produces a trajectory that the joints are expected to follow. In order to make the joints follow the given trajectory, a control loop must be devised for each joint. In order to linearise the system the computed torque [34] method will be used. The essence of this method is to use knowledge of the system dynamics, to calculate a torque that compensates for said dynamics. Effectively, this makes the model linear, as it is reduced to an integrator, as seen in figure 6.1This does however imply that the actuator can produce enough torque to move the joint at the desired acceleration. The computed torque method requires a model of the system dynamics, which can be described by the Euler-Lagrange equation of motion [34]. The first step of applying this method, is to find the kinetic energies k_i of the system as shown in equation 6.8. Here m_i is the mass of link i of the robot, v_i is the linear velocity, I_i is the inertia matrix, and ω_i is the angular velocity of joint i.

$$k_i = \frac{1}{2} m_i v_i^2 + \frac{1}{2} I_i \omega_i^2 \tag{6.8}$$

The second step to obtaining the Lagrangian, is to compute the potential energies p_i of the system. These are found as seen in equation 6.9. Here h_i is how high body i is above the ground plane. m_i is the mass of body i and g is gravity.

$$p_i = m_i g h_i \tag{6.9}$$

The expressions for kinetic and potential energy can be used to form the Lagrangian L, as shown in equation 6.10

$$L = \sum_{i=1}^{\infty} k_i - p_i \tag{6.10}$$

Knowing the Lagrangian and the vector of generalised coordinates q, the Euler-Lagrange equation can be formulated as shown in equation 6.11. The generalised coordinates is a vector of the positions of the robot joints and body frame. The Γ is the vector of joint torques.

$$\Gamma = \frac{d}{dt}\frac{\delta L}{\delta \dot{q}} - \frac{\delta L}{\delta q} \tag{6.11}$$

Reformulation equation 6.11, the dynamics can be put on the form of equation 6.12. Here M is the nxn mass matrix, V is an $nx1$ vector including the Coriolis and centrifugal force terms and G is the $nx1$ gravity vector [34]. Adding this to the control loop, yields a control structure as seen in figure 6.1.

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \Gamma \tag{6.12}$$

An interesting observation is the case where the position error e becomes zero. In this case the controller becomes solely a gravity compensator, calculating the necessary joint torque to cancel out the gravity force. This model does however not include all dynamics. A major part that has been left out is the contribution of contact forces when the feet collide with the ground. To keep the effect of the contact dynamics as low as possible, the kinetic energy of the foot should be low when coming into contact with the ground. This should be taken into account when planning the trajectories for the feet.

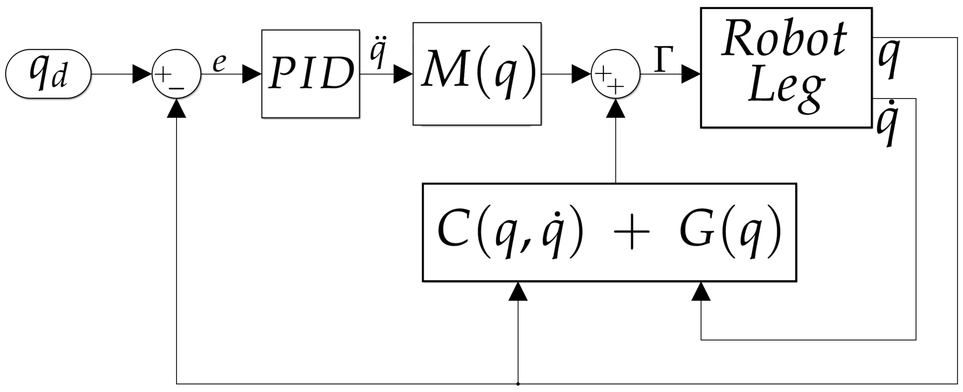


Figure 6.1 Position controller with computed torque system linearization. The linearization consists of a model of the mass matrix, Coriolis vector and gravity vector. The generalised coordinate q is the joint angle reference computed by inverse kinematics.

CHAPTER | 7
Simulation

Before implementation of the control strategy it is tested through simulation. The results of the simulation can be used to verify that the chosen strategy provides satisfactory system performance. Additionally the simulation will give a measure of the system performance given the chosen stepper motors, and whether the motor size should be re-evaluated. The robot has been modelled in Matlab/Simulink using Simscape Multibody and its associated libraries. The simulation is separated into two parts: The walking controller and the dynamic model of the robot and the overview can be seen in figure 7.1.

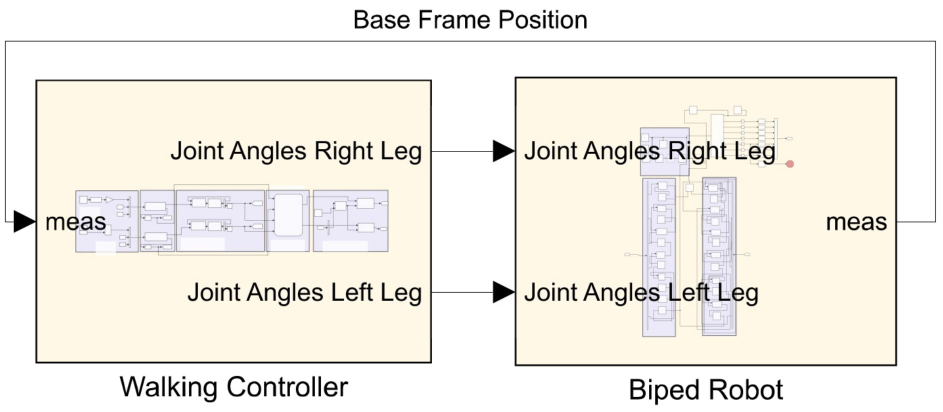


Figure 7.1 The Simscape model of the biped robot and the walking controller used to control it

7.1 Simscape Multibody Model of the Robot

The robot is modelled using the Simscape Multibody library of Simulink. The model is built from 12 revolute joints, connected by 13 rigid links. This structure can be seen in figure 7.2 where the left and right leg are shown, connecting to the the common hip link. Each link of the robot has an associated mass, moment of inertia and product of inertia. These are generated from the CAD file of the robot, which can be exported to Simscape Multibody. Here each part of the robot has been assigned the appropriate density to approach the same mass distribution as the physical robot. This means that each individual bolt, nut, motor etc. of the robot has been assigned a density matching the material of which it is made. This is an important aspect of the simulation, as the dynamics associated with the masses of the system significantly impacts the systems behaviour.

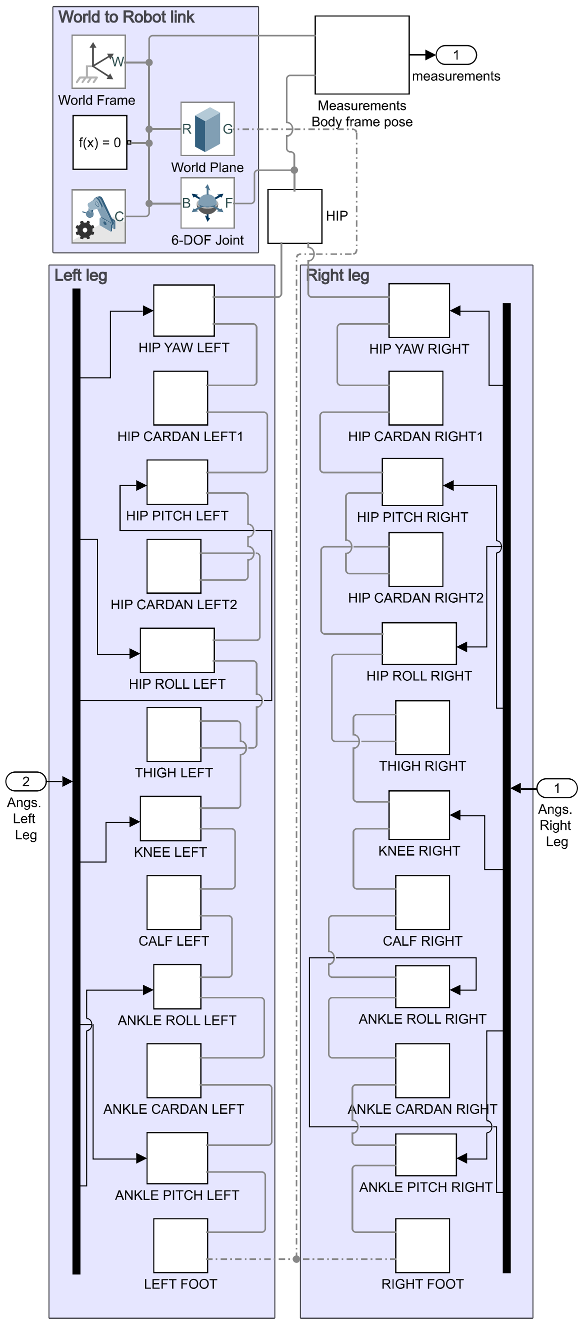


Figure 7.2 The Simscape model of the robot, separated in right leg, left leg and world, all connected at the hip link of the robot

Another important aspect of the model is the associated coordinate frames and length of links. The DH-parameters used to derive the inverse kinematics described in chapter 3 must correspond to the ones of the model. This has proven to be a non-trivial task, as the coordinate frames generated by Simscape does not follow the DH-convention. Each coordinate frame of the kinematic chain must be transformed to follow the DH-convention, described in section 3.1. Here the coordinate frame of each joint must rotate about the z-axis, and the x-axis must point along the next link in the kinematic chain.

Modified Robot Design

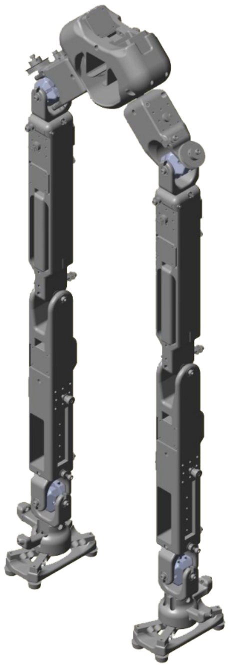


Figure 7.3 The robot design where the yaw joint of the hip is mounted at a 45 degree in relation to the hip link

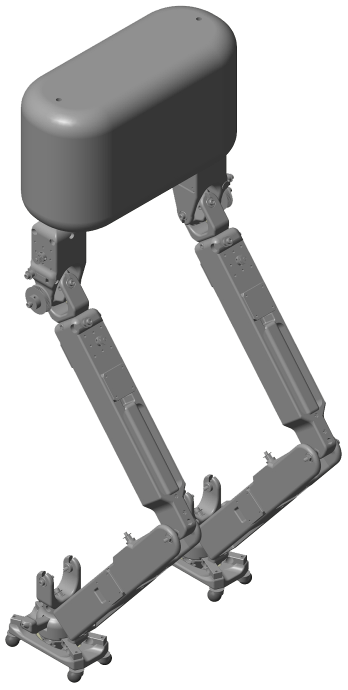


Figure 7.4 The robot design where the yaw joint of the hip is perpendicular in relation to the hip link

The original mechanical design of the robot discussed in chapter 2 and seen in figure 7.3 has been changed to the one seen in figure 7.4. Within the scope of this project it was not possible to control the robot shown in figure 7.3. This can be accounted to either an error in the derived inverse kinematics or an error in the Simscape model of the robot. Changing the robot design to the one seen in figure 7.4, and directly applying the inverse kinematics method described chapter 3 on page 27 proved solvable. The main difference of the two robot designs lies in the hip yaw joint. In figure 7.3 the hip yaw joint is mounted with a 45 °angle, while in figure 7.4 it is mounted at an 90 °angle. The design using 45 °angle should however be investigated further. Experiments shows that the design using 90 °does not utilise the yaw joint while walking straight, which puts a high load on the remaining hip motors. The 45 °design utilises all motors for walking straight to reach the desired angles.

Another modification of the robot design is the location of the two ankle joints. In the original design they are located at a distance of 116.7 mm above the sole of the foot. In the new design the location of the ankle joints have been moved closer to the sole of the feet, such that they are now located at a distance of 36.7 mm above the sole. This change was made based on experimentation which showed that ankle joints located further from the sole made the robot more susceptible to loss of foothold, as the forces affecting the ankle at the initial height would exceed the foot support frame and tilt the ankle.

Friction

The walk of the robot relies heavily on the interaction between the foot of the robot and the ground. A crucial part of this interaction is the dry friction between the ground and the feet. As explained in chapter 4, for a ZMP point all forces acting on it must sum to zero. This is visualised in figure 4.2 on page 39. If the supporting foot slips, collapse of the robot is likely.
Friction can be separated into two categories; static friction and dynamic friction. As the names of these implies that static friction is the friction force that exists while the foot remains stationary, while the dynamic friction is when the foot slides. For this simulation friction coefficients for static and dynamic friction has been chosen to simulate the interaction between dry rubber and concrete [46]. To prove that the robot can operate in a variety of terrains, performance should be tested for a range of coefficients of friction.

Contact Forces

Another interaction that heavily impacts the operation of the robot is when the feet and the ground plane collide. In this simulation the Simscape Multibody Contact Forces Library has been used to simulate contact between the two bodies, such as seen in figure 7.5.
Figure 7.6 shows the 4 contact points of the supporting foot, marked with red spheres. Simscape Multibody Contact Forces Library defines contact through 3 parameters: contact damping, contact stiffness and transition region width. The damping is given in the unit $N/(m/s)$ and is the dissipation of energy, based on velocity. Having too little damping results in an underdamped system, which is experienced as the foot bouncing on the surface after the initial contact. On the other hand, having too

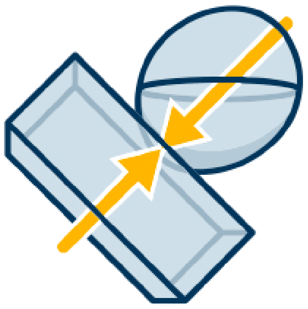


Figure 7.5 The forces of colliding bodies

much damping causes the system to become overdamped, resulting in a long settling time upon contact. The ideal damping is found by means of experimentation. However, for implementation purposes the damping is not as easily controlled as in simulation. In the practical cases, the damping is, like friction, determined by the properties of the colliding bodies, namely the foot and the ground.

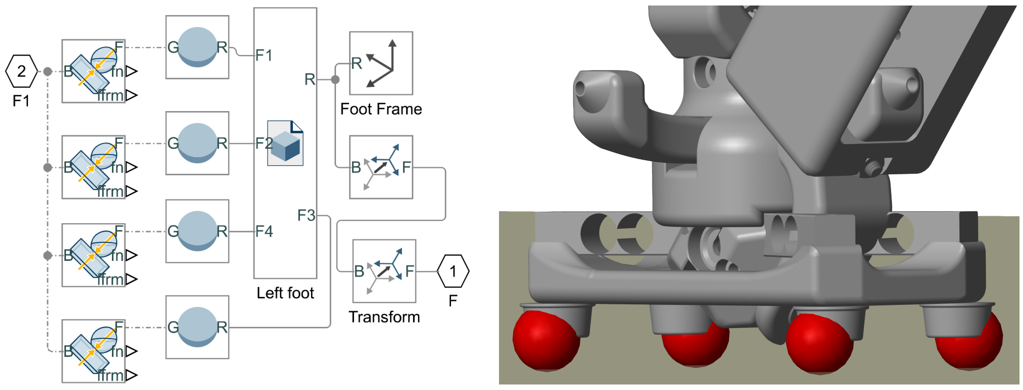


Figure 7.6 The supporting foot of the robot, shown schematically and animated. The contact points are shown using red spheres. The four contact points of the robot are rigidly connected to the foot

In addition to damping, the stiffness must also be determined. Stiffness is a measure of how much an object resists deformation, and has the unit N/m.

It is important that the stiffness should be chosen such that the feet of the robot does not break through the ground plane. Another argument for increasing the stiffness of the contact points is to avoid the robot tilting. Too low stiffness makes the robot sink into the ground plane, which becomes apparent when the projected COM of the robot moves close to the edge of the support polygon. This causes the unwanted behaviour of the edge of the foot "sinking" into the ground plane. These arguments motivates the choice of a high amount of stiffness to be applied. The downside however, is that experiments shows that an increase in stiffness negatively impacts simulation speed.

Connecting the Robot and the World

In figure 7.7 the robot is connected to the world frame (top-left block) using a 6-DOF joint. This allows for the robot to move in any direction and orientation in the simulation world. The centre-left block denoted $f(x) = 0$ is used to configure the specific ODE solver. For this simulation the backward Euler method has been chosen. The bottom-left block defines external forces, applied to the bodies within the simulation. In this simulation the only external force is the gravity force. The gravity force vector has been defined as perpendicular to the ground.

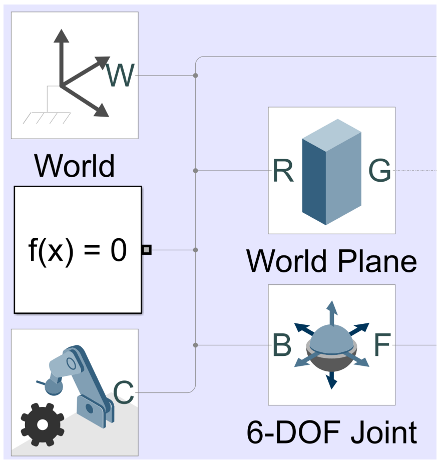


Figure 7.7 The simscape model of the robot showing the solver block, world block, gravity block, world plane block and the 6-DOF joint block connecting the robot to the world.

Joint Position Controller

In order for the joint to follow a given position reference, each joint of the robot model is equipped with a separate joint controller. To best replicate the physical robot, this can be done using a control loop as shown in figure 7.8. This requires that every PI controller is individually tuned to obtain a desired system response for each joint. A properly tuned joint controlled should make the joint track the given reference signal without significant overshoot and with only a slight delay due to the response time. To improve the performance of the joint controllers, an inverse dynamics model should be used to linearize the system, as described in chapter 6 on page 50. This model must take into account all system dynamics, ideally including the dynamics caused by ground contact, and calculate the torque required to compensate for the system dynamics.

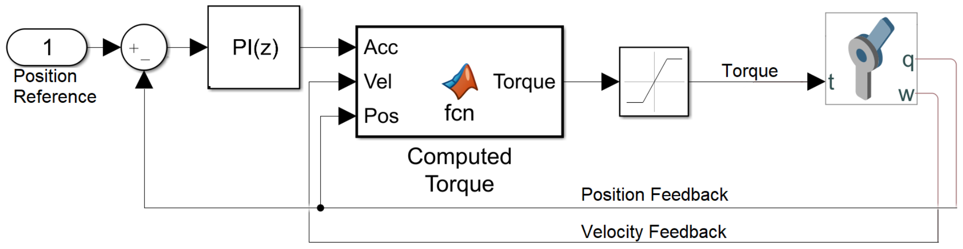


Figure 7.8 PI Controller with computed torque system linearization

Instead of linearization and PI control, Simscape offers an alternative method of joint control. Using the direct motion control of the revolute joint seen in figure 7.9, Simscape automatically computes the necessary torque to reach the desired position. Since the motion controller of Simscape has no torque limits it will follow the discrete signal perfectly. However this causes issues as it produces a high amount of accelerations in the system, which experimentation has shown breaks the friction between the foot and the ground. To improve upon this, a second order filter has been added to the reference, before it enters the joint. Essentially, the motion control along with a second order filter, produces results as could be expected by linearization and PI control of the joint. The direct motion control of the joint can be used as a means of comparison, when designing the actual controller for implementation.

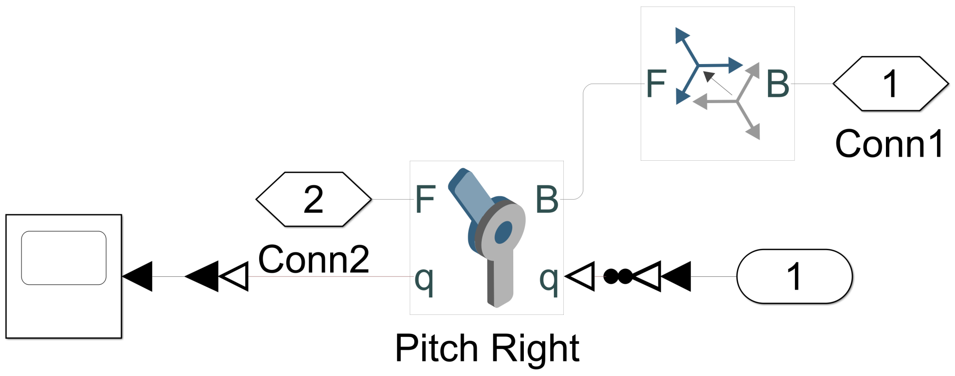


Figure 7.9 The simscape model of a joints layout, consisting of a transformation box, linking the joint together with the other parts of the robot. The joint is directly controlled by the provided position reference, where Simulink computes the necessary joint torque to track the position reference exactly

Measuring Joint Torque Requirements

Simscape offers the ability to measure the joint torque during simulation. This will give an estimate of the torque requirements of the physical robot motors. This can be done by first giving the motors in the simulation unlimited torque capabilities. The joints are then set to follow a specified motion profile that includes the requirements described in section 1.2 on page 5. While the joint follows this motion profile, the torque can be measured. By inspection of the measurements the maximum torque requirements for the joints is then obtained. Comparing this to the torque capabilities of the motors will show if the chosen motors can manage to move the joints as desired.

7.2 Walking Controller

The second part of the simulation is the walking controller. This part encompasses the ZMP generator, the MPC, LIPM, Stepping Logic and Inverse kinematics. The structure of the simulation is inspired by the work of [47].

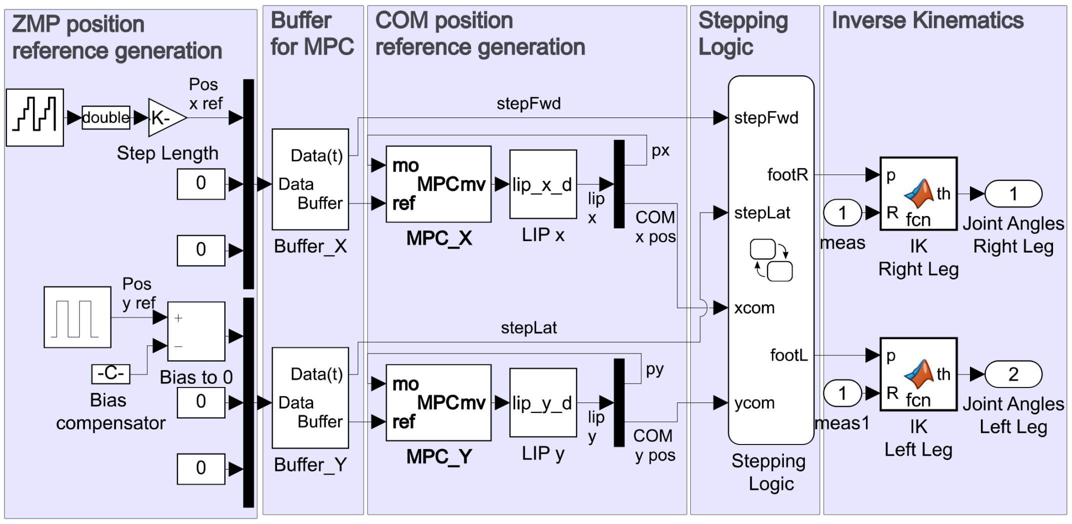


Figure 7.10 Overview of the walking controller in the simscape model, consisting of ZMP generator, MPC, LIPM, stepping logic and inverse kinematics.

ZMP Generator

The first part of figure 7.10 is the ZMP generator. This generator is made for time-based gait, which was described in section 1.2 on page 5, and outputs the x and y position of the ZMP. These positions are derived using the parameters; step length, step time and swing height. The x position is created using a counter which holds the step time parameter and is then multiplied with the step length. Hence if the step time is 1 second and the step length is 20 cm then if the robot is told to walk for 5 seconds the combined step length will be 100 cm. Put differently, there is a new ZMP every 20 cm out on the x-axis. This is also shown in figure 7.11, where the line "stepFwd" is the x position

For the y position a pulse is given instead of a counter, since the feet are positioned opposite of each other on the y-axis. Put differently, when walking outwards the x-axis, one foot will have a negative y position while the other will be positive. This pulse is determined from an amplitude set to the step width, and a period set to the step time. This is shown in figure 7.11, where the line "stepLat" is the y position.

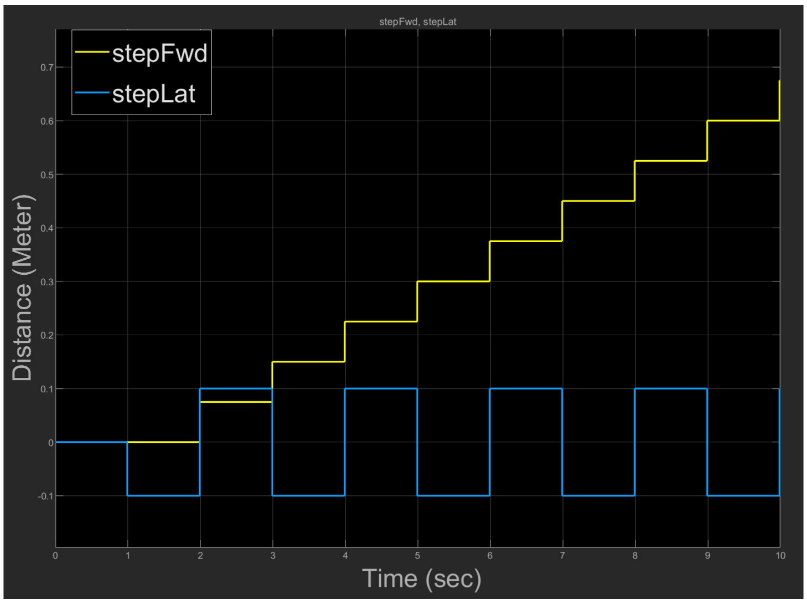


Figure 7.11 Graph showing the ZMP reference generation, showing the x position of the ZMP as "stepFwd" and the y position as "stepLat". Here the parameters are as: "Step time" = 1 sec, "Step length" = 0.075 meter.

The first step in making a walking controller is to define the parameters of the walk. These parameters include the step length, step time and swing height of the foot, and is defined as:

Parameters of the Walk

The characteristics of the swinging foot is determined by the following parameters:

- **Step length**. The length between between the start and end position of the swinging foot.
- **Step width** The width from the centre of the robot out to the foot.
- **Step time** The duration of time for a step, including lift, swing and contact.
- **Swing height** The Z plane height of the foot in the swinging phase.

These parameters are based upon the requirements for the biped robot. If the system is required to move at a high velocity, then the step time and length must be chosen accordingly. Increasing the velocity of the robot can be achieved by faster and longer steps. The stepping speed can be increased until the motors reaches their maximum torque limit. The stepping length can be increased until the joint limits are reached. However, the stepping length is often preceded by

another limit, namely that the foot in contact with the ground must be at a ZMP as described in section 4, which summarises to the following: In order for the foot to be at a ZMP, all moments must sum to zero. A state at which this is not true, is when the horizontal forces of the body acting on the foot, exceeds the friction between the foot and the ground. Should this happen, the foot will slip and the robot will fall. When generating trajectories for the feet of the robot, it must be made sure that these are at all times at a ZMP. Therefore, choosing a too large step length will cause the foot in contact with the ground to slip. Thus, the step length is limited by the amount of friction between the foot and the ground.
For this system the step length, step time and swing time should be chosen as to fulfil the requirement stated in 1.4 on page 14, which states that the robot should aim for a walking speed of 0.5 m/s. The optimal stepping length and speed is to be found through experiments in order to find an optimal balance.

Interpolation of Trajectory from ZMPs

When the robot is tasked with moving a foot to a new point, a suitable trajectory that connects the current position of the foot, with the desired position, must be computed. Since it is assumed that the robot walks on a flat surface, the shortest distance between the two points is a straight line. Before the foot can be moved it must break contact with the ground so that it is not affected by friction. It is achieved by raising the height of the foot during the transition between points, starting and ending with a height of zero. There exists numerous solutions for the shape of the trajectory. One solution, is to use a cubic polynomial to interpolate a trajectory between the current foot position and the goal foot position [47]. The swinging foot will reach the top of the swing at the midpoint in time, as can be seen in figure 7.13. Likewise, the movement in the Y direction also uses a cubic polynomial as shown in figure 7.12. For the case where the foot is only moved in the YZ plane, the foot will follow the trajectory shown in figure 7.14, if cubic polynomial interpolation is used.

Depending on the situation, other trajectories might be more suitable than the cubic polynomial interpolation approach. In a situation where there are obstacles on the ground, it might be desirable to elevate the swinging foot further, and at an earlier point during the step. A simple approach could be to elevate the foot to a certain height, before moving the foot in the X or Y direction. This would decrease the likelihood of tripping over an object in front of the foot.

Now that the ZMP references are derived, the next step is to contruct the LIPM and design the MPC.

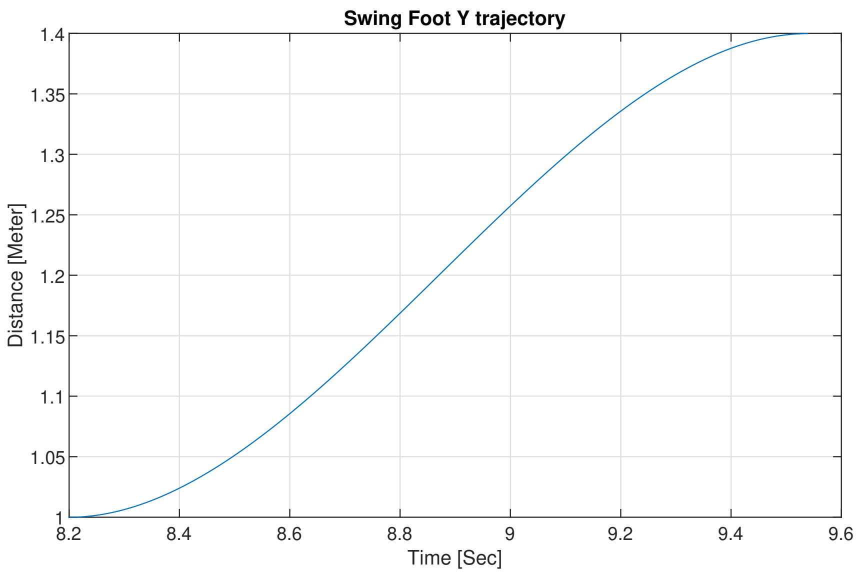


Figure 7.12 Swing Foot trajectory in the Y axis

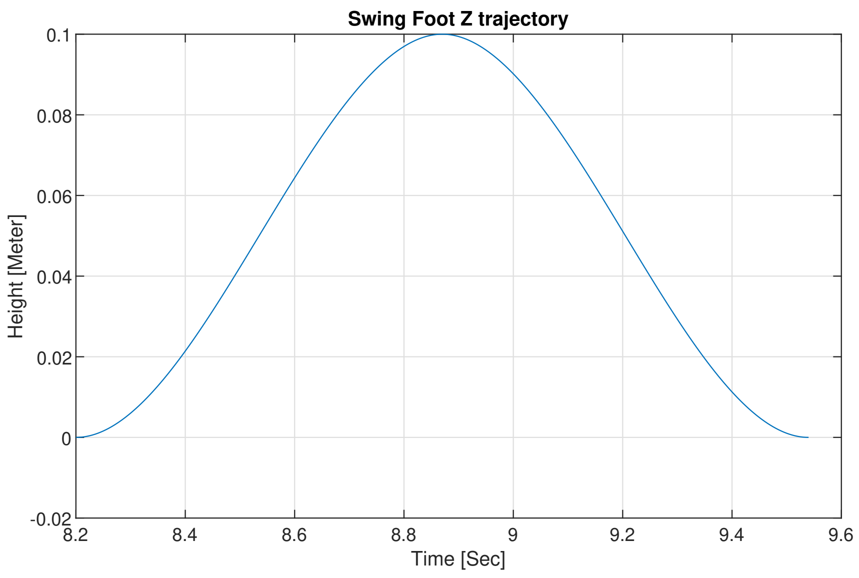


Figure 7.13 Swing Foot trajectory in the Z axis. The peak of height is reached at the midpoint in time

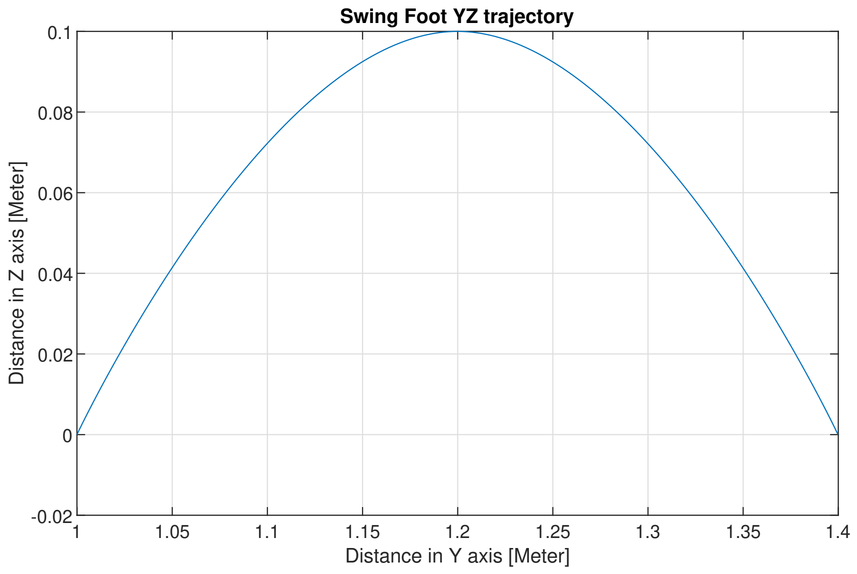


Figure 7.14 Swing Foot trajectory in the YZ axis

MPC Block

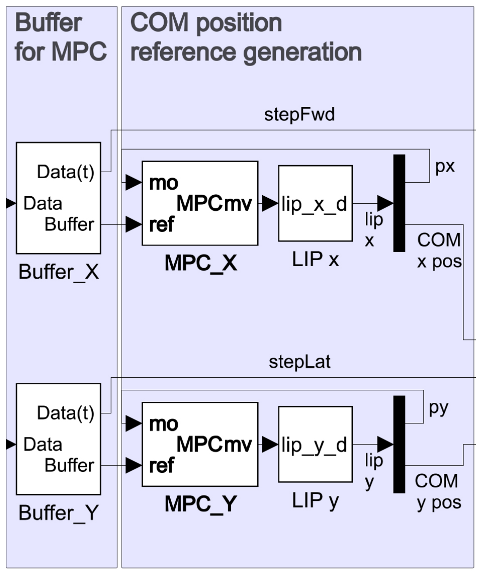


Figure 7.15 MPC and LIPM part of the walking controller simulation

In figure 7.15, two block named MPC_X and MPC_Y can be seen. These blocks uses the reference "ref", together with with the measured output "mo", in order to calculate the manipulated variable "mv". The reference consists of a series of future ZMP references, spanning for the length of the prediction horizon. This entire series of ZMP references is passed to the MPC. This gives the MPC knowledge of future inputs, as explained in chapter 6 on page 50. The measured output is the feedback used for the MPC. For the MPC shown in figure 7.15, the feedback consists of px, py, which is the measured ZMP. This means that it can be considered an open-loop trajectory generator, as it has no feedback from the model of the actual robot. Ideally, the references from this open loop trajectory generator should be sufficient to keep the robot walking. However, since the MPC is decoupled from the actual robot, the trajectory will not be altered to correct for disturbances.

The solution is closing the control loop and can be done by replacing the feedback from the LIPM with feedback from the actual robot. This is done by providing the MPC with measurements of the ZMP. The ZMP is found using the calculations described in section 4.2 on page 43, and measurement of the contact force. The contact force is measured directly from the contact points shown in figure 7.6, using the "fn" output. Due to time limits the controller did not receive feedback from the actual biped robot.

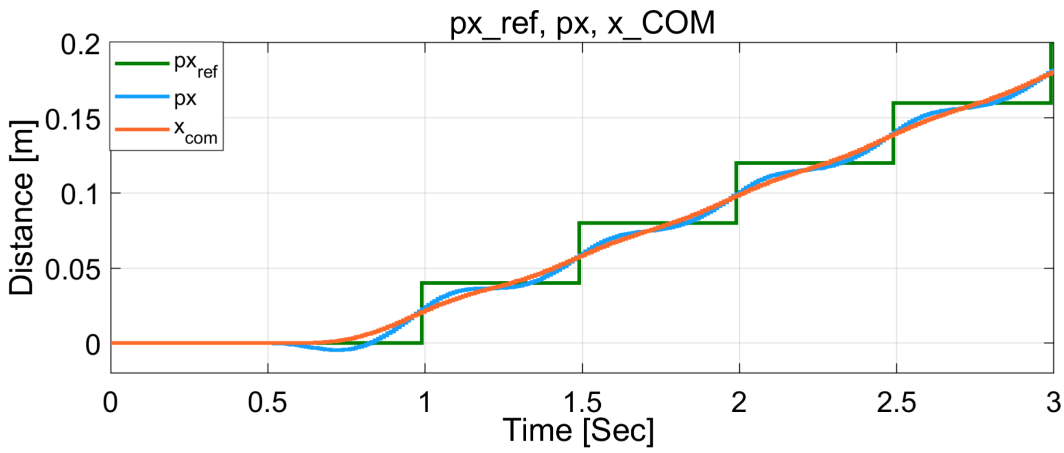


Figure 7.16 Reference tracking of the MPC. Blue is ZMP x reference. Yellow is COM trajectory. Notice how the MPC begins tracking the reference ahead of time

The performance of the MPC can be seen in figures 7.16 and 7.17. These figures show the ZMP reference and the resulting COM trajectory produced by the MPC, with the output states being: COM position, COM velocity and foot position. An important detail in these figures is that the MPC begins tracking the reference ahead of time. This is the result of supplying the MPC with the series of future references and using its advantage: predictive control. The controller is focused on energy conservation and as can be sen is behaving in a smooth manner despite the jagged reference.

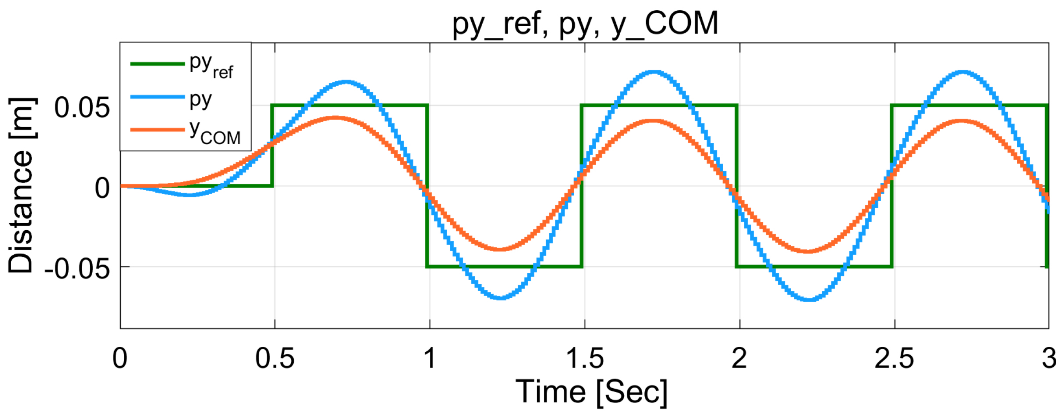


Figure 7.17 Reference tracking of the MPC. Green is the controller reference. Blue is ZMP y reference. Orange is COM trajectory. Notice how the MPC begins tracking the reference ahead of time

Controller Constraints

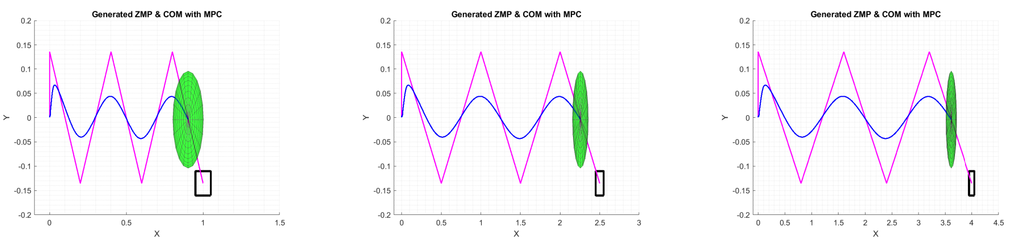
The controller outputs two position states and one velocity state. The velocity constraint should be set to the requirement noted in section 1.4 with a soft constraint as the gait velocity is not of critical nature. The position constraints should be set such that the COM and foot is not allowed excessive swings in Y plane while a slope constraint for the rising X coordinated should be used.
Input constraints can also be set to limit the input to the controller. This reduces the impact on the controller if it receives values that are not reliable or are noisy. For this simulation the controller receives feedback from the ideal LIP model. Therefore no constraints are necessary.

LIPM Block

The LIP model is designed using dimensions resembling the physical robot. The simulations are based upon the desired walking velocities, noted in the requirement specification 1.4 on page 14. The simulation has been carried out for three COM velocities of 0.4, 1.0 and 1.6 m/s. The LIPM model is affected by the step length, step width and step time. The step time was chosen as 0.5 second and step lengths are 0.2, 0.5 and 0.8 m. The step width was set equal to the hip width of the robot, which is 0.135 m.

$$
\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_x , \quad \frac{d}{dt}\begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_y
$$
$$
p_x = \begin{bmatrix} 1 & 0 & \frac{-z_c}{g} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} , \qquad p_y = \begin{bmatrix} 1 & 0 & \frac{-z_c}{g} \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} \tag{7.1}
$$

The MPC controller is connected to a state space model of the LIPM, seen in equation 7.1. Tests showed the controller required a prediction horizon of 3 steps when using a sample time of 100 Hz, thus a prediction horizon of 150 time stamps to make the model stable. The controller outputs 1 measured state, feet position, and 2 unmeasured states, the COM velocity and COM position. The simulation results can be seen in figure 7.18. The results show the LIPM model is stable and is able to be used for the simulation of the biped robot.



(a) LIPM simulation walking speed of 0.4 m/s with a step length of 0.2 m

(b) LIPM simulation walking speed of 1.0 m/s with a step length of 0.5 m

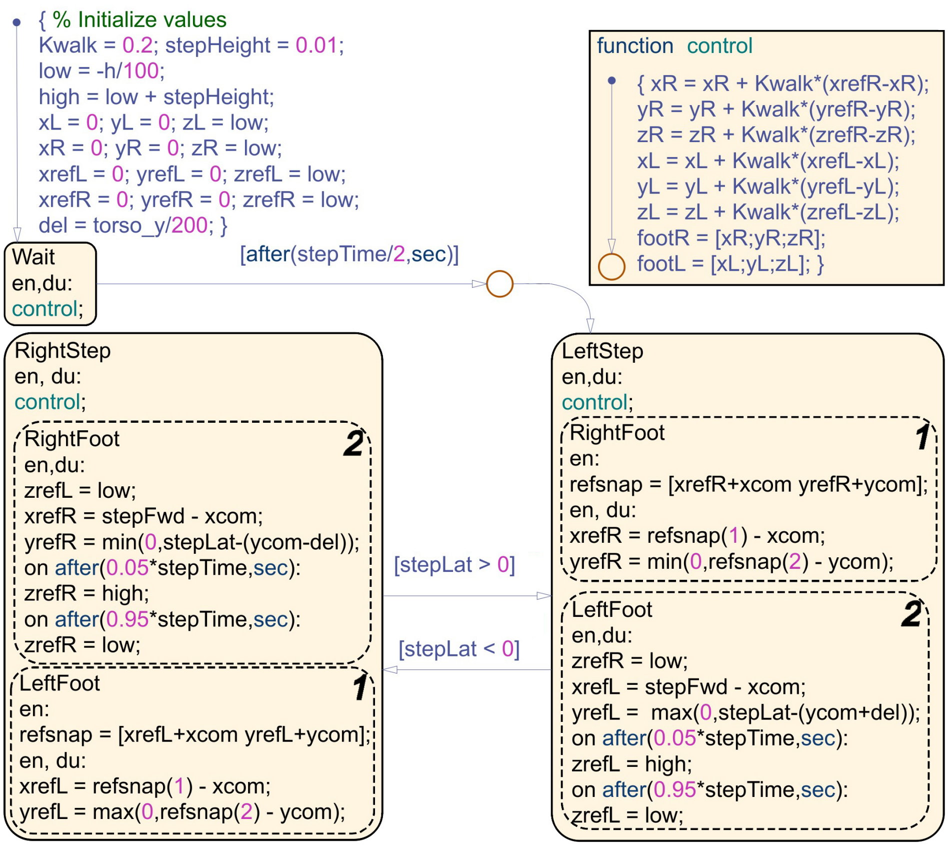
(c) LIPM simulation walking speed of 1.6 m/s with a step length of 0.8 m

Figure 7.18 Graphs displaying the X,Y position of the COM of the LIPM simulated model with a COM height set to the value of 0.7 m. The blue line indicates the COM position and the purple line indicates the LIPM foot position reference. The figures can be seen in full detail in appendix A.4 on page 91

Stepping Logic

The contents of the stepping logic block seen in figure 7.10, is shown in figure 7.19. The design of the stepping logic is based on the work of [47]. From this figure it can be seen that the stepping logic consist of two phases for both legs. The figure shows that the stepping logic block contains some initial values, a "Wait" block, a control function on the upper right corner and a "RightStep" and "LeftStep" block. The "Wait" block is defined to wait for the amount of *stepTime*/2. After this time has passed the "LeftStep" block is executed. This block starts by setting the right

foot as support foot, and sets the left foot as swing foot. Then the value of the "stepLat" signal is checked, this signal defines the lateral movement of the foot. In this case, where the robot is walking along the x-axis, the lateral step reference will change between negative and positive as shown in figure 7.11 depending on which foot is actuated.
This procedure is then copied for the "RightStep" block, but now with the left foot as support foot and right foot as swing foot instead. This results in the phases of a robot gait, as was shown in figure 1.10 on page 11. The output of the stepping logic block are the x, y, z position references for the feet throughout the simulation. These trajectories are then send to the inverse kinematic, to compute the necessary joint angles.



```
{ % Initialize values
Kwalk = 0.2; stepHeight = 0.01;
low = -h/100;
high = low + stepHeight;
xL = 0; yL = 0; zL = low;
xR = 0; yR = 0; zR = low;
xrefL = 0; yrefL = 0; zrefL = low;
xrefR = 0; yrefR = 0; zrefR = low;
del = torso_y/200; }
```

```
function control
{ xR = xR + Kwalk*(xrefR-xR);
yR = yR + Kwalk*(yrefR-yR);
zR = zR + Kwalk*(zrefR-zR);
xL = xL + Kwalk*(xrefL-xL);
yL = yL + Kwalk*(yrefL-yL);
zL = zL + Kwalk*(zrefL-zL);
footR = [xR;yR;zR];
footL = [xL;yL;zL]; }
```

```
Wait
en,du:
control;
```

```
[after(stepTime/2,sec)]
```

```
RightStep
en, du:
control;
RightFoot 2
en,du:
zrefL = low;
xrefR = stepFwd - xcom;
yrefR = min(0,stepLat-(ycom-del));
on after(0.05*stepTime,sec):
zrefR = high;
on after(0.95*stepTime,sec):
zrefR = low;
LeftFoot 1
en:
refsnap = [xrefL+xcom yrefL+ycom];
en, du:
xrefL = refsnap(1) - xcom;
yrefL = max(0,refsnap(2) - ycom);
```

```
[stepLat > 0]
[stepLat < 0]
```

```
LeftStep
en,du:
control;
RightFoot 1
en:
refsnap = [xrefR+xcom yrefR+ycom];
en, du:
xrefR = refsnap(1) - xcom;
yrefR = min(0,refsnap(2) - ycom);
LeftFoot 2
en,du:
zrefR = low;
xrefL = stepFwd - xcom;
yrefL =  max(0,stepLat-(ycom+del));
on after(0.05*stepTime,sec):
zrefL = high;
on after(0.95*stepTime,sec):
zrefL = low;
```

Figure 7.19 Stepping logic block [47]. Per top left corner the algorithm starts and sets initial system values. After a wait time, to allow for filling the MPC controller reference buffer, the algorithm enters LeftStep and switches between it and RightStep after each step as stepLat changes

Inverse Kinematics

In order for the simulated robot to move the robot feet to their desired position, the inverse kinematic has to be implemented. This is done by using the expressions for θ derived in section 3.2 on page 30 in a function in the simulation. The function blocks is seen in figure 7.10 on page 65, at the right side on the figure, and is shown in more details in figure 7.20. Note that this figure only show the inverse kinematic function for the right leg but there are two inverse kinematic functions, one for each leg. This was done to avoid having to use true/false statements to define which leg the function should solve for. The reason for this, as was explained in section 3.2 on page 30, is that there is a difference in the sign of $L1$ between the two legs.

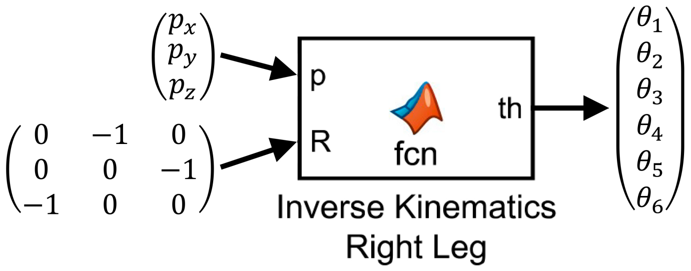

$\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$
$\begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{pmatrix}$
p
R
th
fcn
Inverse Kinematics
Right Leg
$\begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{pmatrix}$

Figure 7.20 Inverse kinematics function block, which takes in the a position vector p of the foot, and a rotation matrix R Which defines the orientation difference between the base and the world. From these inputs the function then outputs a vector of theta's in this case six, as each leg has 6 motors.

These functions take in the desired position of the feet and the rotation matrix between the base frame and the world frame, which for this project is fixed, as the orientation of the base is fixed as was mentioned in section 1.2 on page 5.
From the feet position references, the inverse kinematic function calculates and output the necessary joint angles θ_i. These joint angles are send to the joint blocks as shown in figure 7.9 on page 64 which will then realise the desired feet position, by moving the joints.
In order to verify that the obtained inverse kinematics is correct, an initial test is conducted as seen in figure 7.21. Here the robot is fixated at the hip link, hovering above the ground plane. The inverse kinematics are given foot references with constant values of 0 for the x, y positions. For the z references the inverse kinematics are given a sine wave reference, with a bias of -0.6 and an amplitude of 0.2. This causes the robot to move the feet up and down, purely along the z-axis, while the x, y positions remain constant.

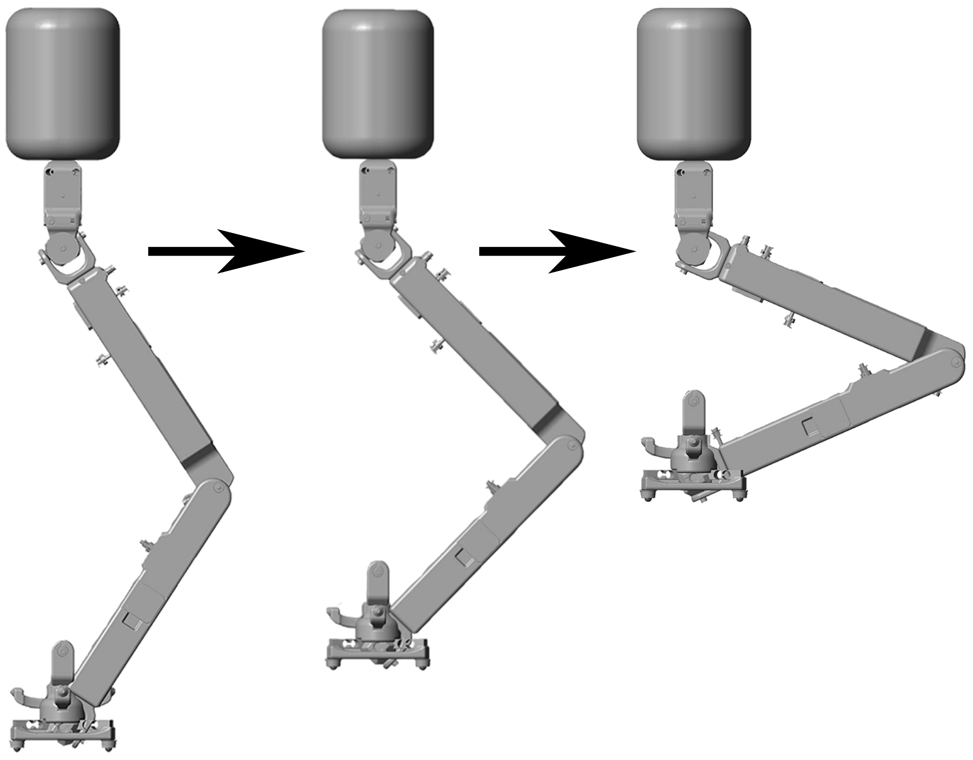


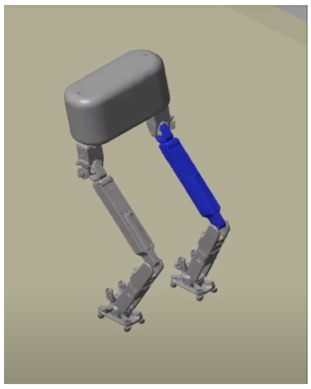
Figure 7.21 Testing the inverse kinematics of the robot. The robot is fixated at the hip link. The system is tasked with moving the legs straight up.

Walking Controller Performance Test

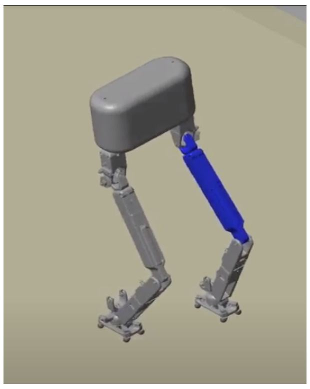
The 9 snapshots shown in figure 7.22 on the next page shows a test of the walking controller implemented on the simulated robot. The test was terminated when the robot fell. During the test, the robot managed to move the COM above the support foot, lift the swing foot, and take a step. This completes half a gait cycle. However when tasked with taking the next step, the walking controller fail. The robot attempts to move the COM above the new support foot, however it lifts the new swing foot too early, and causes the fall of the robot.

(a)

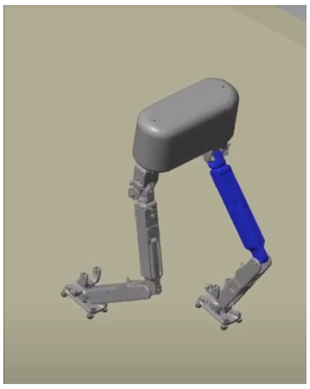
(b)

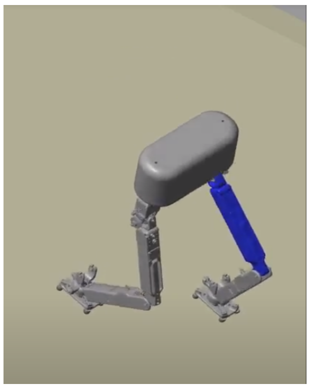

(c)

(d)


(e)

(f)


(g)


(h)

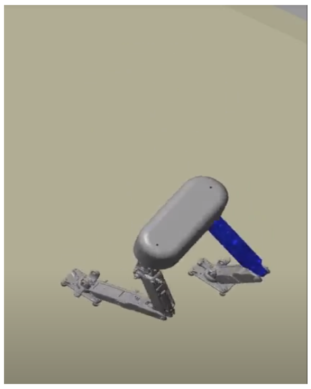

(i)

Figure 7.22 These 9 figures are taken as snap pictures from the gait simulation starting with a initial position in (a), beginning a step in (c)/(d), swinging the foot in (e) and impacting the ground in (f)/(g). The robot then tumbles completely in (h)/(i)

7.3 Simulation Evaluation

The simulation had a proven LIP model, reference generator and MPC controller. The inverse kinematic block performed as intended when performing a separate Z test motion. But when the dynamic gait was initialised the robot would fail as seen in figure 7.22. The X,Y motion was also tested with the inverse kinematic block but as an undocumented preliminary visual test. The source of the gait problem was guesstimated to be in the inverse kinematic block or the R-matrix of the base frame but no conclusive evidence was found, with the error being unknown.

The result of the walking controller test in section 7.2 suggests that the initial conditions of the gait, described in section 5.5 on page 49 are not properly adjusted for the system. Had the COM of the robot moved at a higher forward velocity, it is possible that the fall could had been prevented.

CHAPTER | 8
Implementation

The following chapter will describe the implementation part of the project spilt into hardware, setup and communication. However due to physical limitations to the hardware and test facilities the project implementation was hindered and thus a much greater focus was put on simulation.
The assembled state of the robot showed a promise of being able to perform dynamic walk as the joints could be operated as intended. The chapter is split into three section: Hardware, Proposed Setup and Communication with the implemented hardware first.

8.1 Hardware

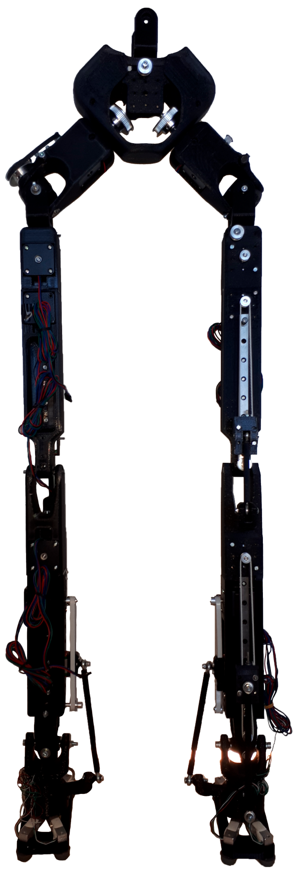


Figure 8.1 Assembled state of the biped robot

The robot was built with the initial assembly in mind and therefore has a 45° hip joint. The assembled state can be seen in figure 8.1. When assembling, minor production inaccuracies in 3D printing and miscellaneous parts affects the dimensions of the robot and thus affected the range of motion slightly. The upper body attached to the hip was not created for the prototype but can be added for future endeavours. The joint angles of the physical robot were measured and are noted in table IV in chapter 2. A cable harness for the power and I2C communication was created allowing the motors to be powered in parallel and for communication between uSteppers, IMU and ESP32.

The 3D print proved its value being both solid and lightweight, it was deemed future improvements of minimising the link dimensions was possible with the material, however unnecessary until the design had performed a complete burn in test.

8.2 Proposed Setup

The setup proposed for the biped robot would comprise of a wheeled frame with wires as to catch the weight of the robot in the case of a fall. This insures a safer testing environment as the falling weight may cause damage or injury of its surrounding. This will also prevent the robot from breaking apart from the kinetic impact force created by a fall. The wheeled frame is pushed in the direction the robot is walking and should not support the robot, or impact the dynamics, in any way.

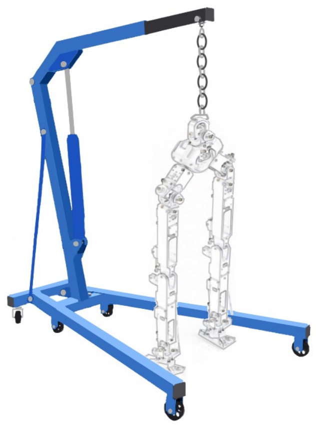


Figure 8.2 Proposed setup for dynamic walking, note the lift is wheeled and would be pushed forward so as not the hinder the walk and only catch the robot when errors occur and it falls

As preliminary tests would drain the batteries many times over, with a worst case battery life of approx 13 minutes, as noted in chapter 2, a cabled power supply capable of supplying the same voltage and ampere would be connected instead with wiring suspended from the wheeled frame. The proposed setup can be seen in figure 8.2.

To document the dynamic walk a Vicon motion tracking system could have been used to check joint angles and velocities. The system is vision based with 12 cameras capturing reflective spheres attached to the joints and links of the robot. It has a capture rate of 100 *Hz* and can thus calculate the velocity of the joints.
The Vicon system would be used to verify the physical motion to the controller output, proving if the controller performs as expected.

8.3 Communication

The parts of robot communicate with each other with I^{2}C. Some initial code between the uStepper's, strain gauges, ESP32 and IMU was created thus sudo code is presented in figure 8.3 displaying the ESP32 master algorithm.

The ESP32 calculates the joint positions reference by the use of the walking controller designed in the simulation. The references are forwarded by I^{2}C to each individual motor using the joint angle command. The message is split into three parts being the ID, joint velocity and rotation direction, clockwise or counter clockwise. The pseudo code is based upon the time based gait as it is simpler to implement. An event based gait should also be created and tests of the systems carried out to collect data on which strategy is superior.

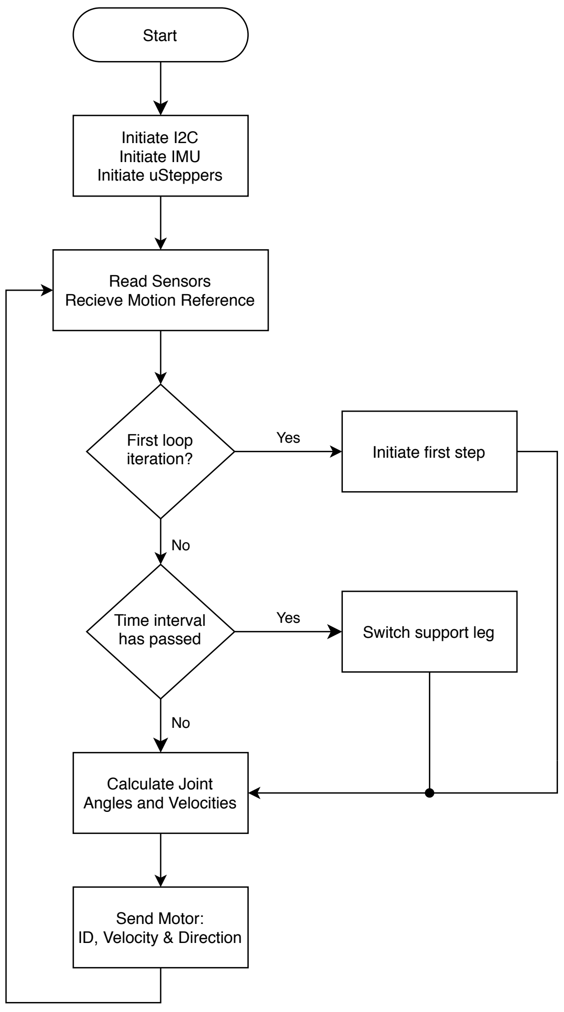


Figure 8.3 Sudo code for ESP32 receiving a motion reference and command the legs to move by switching support leg after a leg movetime interval has passed

CHAPTER | 9
Conclusion & Discussion

9.1 Conclusion

A state of the art analysis of existing biped robots was conducted. Human gait was analysed and a robot gait strategy approximating the human gait was developed. Appropriate hardware for the instrumentation of the robot was researched, and the most suitable candidates were described.

A lower body biped robot was constructed using primarily 3D printed materials. Each joint of the robot successfully fulfilled the ROM requirements for dynamic walk of a human. The links lengths of the robot closely resembles that of a human. The friction of the soles of the feet were never tested, so it is unknown whether they suffice or not. A communication network was designed to ensure communication across all peripherals.

The ability of the biped robots motors to reach their required velocities was never tested. A program that samples the sensors and actuators at the required rate was never created. Inverse kinematics was derived for both the 45° and 90° robots, and were successfully tested in their own environments, however successful implementation into the simulation environment for either was not obtained.

Favoured biped robot modelling strategies were analysed and their potential for this project was described. A trajectory generator was designed using MPC and a solution for a closed-loop trajectory generator using the measured ZMP was proposed. A simulation environment was developed to test the designed walking controller and inverse kinematics with the test environment including a full mechanical model of the robot, contact forces, friction and joint actuation. This project did not conclude in the robot being able to walk at the desired velocity of 0.5 m/s. At the termination of the project the robot was able to walk a few steps before falling, as shown in figure 7.22

9.2 Discussion

During the project period the unfortunate event of a global pandemic known as COVID-19 occurred and affected the project by denying access the to university, group room and test/production facilities. The physical robot was not completed before the lock down as access to production tools like 3D printers was denied resulting in the construction of the robot not being fully completed. Further, the closure of the test facilities resulted in tests such as sole friction, motor velocity, sensor and actuator sample rate was not possible. The project was thus shifted from practical oriented to simulation oriented with a simulation of the biped robot performing dynamic walk.

The robot was constructed with a 45° hip as such a hip is more anatomically correct compared to a regular 90° hip. It would also allow for a more evenly distribution of the torque required to move the legs, split between the hip motors. However implementation of the 45° and 90° inverse kinematic model in the simulation did not work flawlessly. This is assumed to be a problem with the rotation matrix R which describes the orientation of the feet in regards to the base, since the inverse kinematic worked separately, and the problem arose when changes in the orientation of the robot base occurred. The simulation worked as intended for movement along the z-axis as is shown in figure 7.21 on page 75.

The hardware communication was initially developed with SPI in mind with time invested in data transfer between the units being the uStepper S, IMU and EPS32. The communication worked when handling simple tests and small networks, however the uStepper S was found not suitable as it could not be used as a SPI slave due to Slave Select restrictions in the software. Instead I^2C was chosen for the communication protocol and saw promising development. With project time constraints and focus on simulation the I^2C communication saw only initial development and the Bluetooth communication was not initiated.

During preliminary friction simulation tests of the ankle height offset proved a problem. The offset would allow the friction forces to not be handled correctly and the forces affecting the foot in the vertical and horizontal plane, when combined, created a vector which would exceed the dimensions of the foot and thus tip the foot and cause slippage. The solution was to lower the axis of rotation for the ankle joints in simulations. However, to implement the solution on the physical robot it would need to have the ankle joints completely redesigned with focus on still having the a range of motion allowing dynamic walk.

The model accuracy of the simulation had some inaccuracies, in regards to resembling the physical robot. These inaccuracies were caused by parameters such as motor constraints not being modelled, as the development of the simulation was limited by time.

Future development on the system could include improved energy conservation by making the robot more humanoid. The development of arms for an upper body should be looked into for that purpose. By introducing arms to the system the energy consumption when walking can be reduced and also make walking easier [13]. Research into the control possibilities of adding the spine to the robot should be done, in order to connect a upper body to the system.

APPENDIX | A
Appendix

A.1 SPI

Serial Peripheral Interface-bus, or in short terms SPI, can be used as a communication protocol for sending data between the different hardware parts on a robot. It is based upon a master-slave setup as seen in figure A.1 with four wires between the units noted as: SCLK, MOSI, MISO and SS. The clock SCLK is set by the master and can be any speed the hardware can handle. MOSI and MISO are the master and slave data communication wires and SS is the slave select controlled by the master.

The SPI protocol is not limited to the use of 8 bit words and can used the function *SPI.transfer.(data);* for a regular 8 bit message and *SPI.transfer16.(data);* for a 16 bit message. A larger message is use full for specifying the velocity resolution, from 8 bit with a resolution of 256, to 16 bit with a resolution of 65.535.

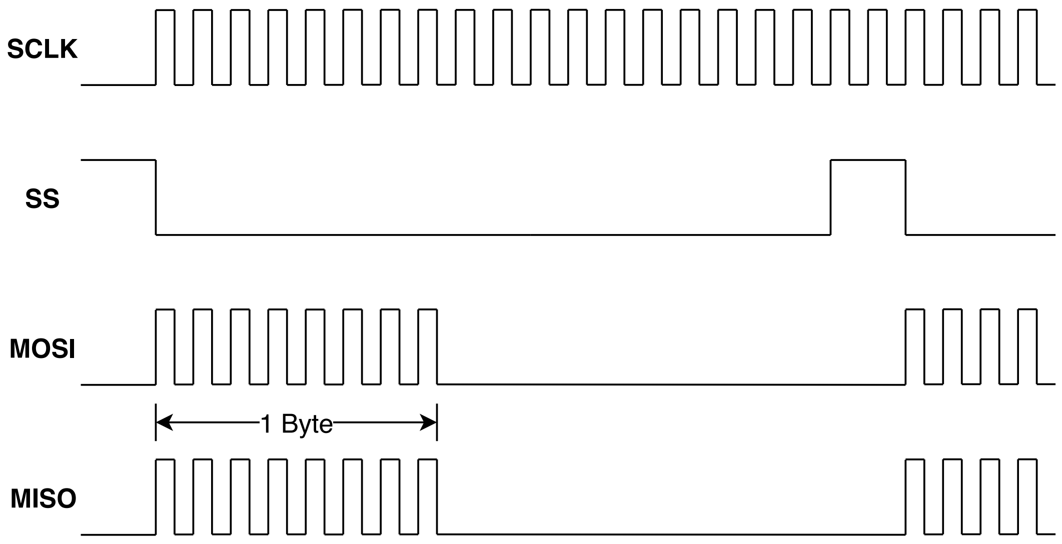


Figure A.1 SPI communication example. 8 high bits are sent by both the master and the slave at the same time when SS is triggered by the master.

With push-pull drivers i SPI offers reliability and high speeds in networks with many attached units. The software implementation allows most setting to be configured on the master while the slave is setup with a interrupt service routine and listens on the SS pin for data transmissions.

The communication is handled by the master which pulls the SS pin low to inform the slaves a data transmission is imminent. The data is then transferred over the MOSI pin in accordance to the SLCK frequency.

The uStepperS NEMA motor driver used is however unable to run as a SPI slave and can only be used as a master. A SPI network can only consist of one master and several slaves therefore SPI cannot solve the communication dilemma for this project.

A.2 Trigonometric Identities

The sine and cosine for the sum of difference of angles θ_1 and θ_2

$$\begin{aligned} cos(\theta_1+\theta_2) &= c_{12} = c_1c_2 - s_1s_2 \\ sin(\theta_1+\theta_2) &= s_{12} = c_1s_2 + s_1c_2 \\ cos(\theta_1-\theta_2) &= c_1c_2 + s_1s_2 \\ sin(\theta_1-\theta_2) &= s_1c_2 - c_1s_2 \\ tan(\theta_1 \pm \theta_2) &= \frac{tan(\theta) \pm tan(\theta)}{1 \mp tan(\theta_1)tan(\theta_2)} \\ arctan(x) \pm arctan(y) &= arctan(\frac{x \pm y}{1 \mp xy}) \end{aligned} \tag{A.1}$$

The sum of the squares of the sine and cosine of the same angle is unity:

$$c^2\theta + s^2\theta = 1 \tag{A.2}$$

Pythagorean identities:

$$\begin{aligned} \text{sin in term of cos} = sin\theta &= \pm\sqrt{1 - cos^2\theta} \\ \text{cos in term of sin} = cos\theta &= \pm\sqrt{1 - sin^2\theta} \\ \text{tan in term of sin} = tan\theta &= \pm\frac{sin\theta}{\sqrt{1 - sin^2\theta}} \\ \text{tan in term of cos} = tan\theta &= \pm\frac{\sqrt{1 - cos^2\theta}}{cos\theta} \end{aligned} \tag{A.3}$$

Inverse function of tan:

$$\begin{aligned} tan\theta &= \frac{sin\theta}{cos\theta} \\ tan\theta^{-1} = arctan = atan2arctan &= atan2 \end{aligned} \tag{A.4}$$

A.3 Inverse Kinematics for 45° robot

The 45° model consist of of seven transformation matrices which describe the transform from base to end-effector, denoted as seen in equation A.5

$$T_8^1 = \prod_{i=1}^{8} T_i^{i-1} = T_2^1\, T_3^2\, T_4^3\, T_5^4\, T_6^5\, T_7^6\, T_8^7 = \begin{bmatrix} x_8 & y_8 & z_8 & p_8 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.5}$$

The 45° model has two other transformations denoted as T_0^B, and T_1^0, which is fixed and contains the transformation seen in equations A.6 and A.7. These are the transformations from the base(located at the centre of the hip) down to the hip-joint intersection point for each leg. The sign of $L1$ depends on which leg the transformation is used for. The right leg has positive signs.

$$T_0^B = \begin{pmatrix} 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & \pm L_1 \\ 0 & 1 & 0 & -L_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.6}$$

$$T_1^0 = \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.7}$$

$$T_3^2 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.8}$$

The remaining transformation matrices from the point of the three intersecting hip joints, to the end-effector, i.e. the foot, can be found as seen in equations A.9. note that angles θ_0, θ_3 and θ_1 are fixed, as these are parts of the fixed transform from the base to the point where the three hip joints intersect.

$$T_2^1 = \begin{pmatrix} C_2 & 0 & S_2 & 0 \\ S_2 & 0 & -C_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T_6^5 = \begin{pmatrix} C_6 & -S_6 & 0 & L_4\,C_6 \\ S_6 & C_6 & 0 & L_4\,S_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
$$T_4^3 = \begin{pmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T_7^6 = \begin{pmatrix} C_7 & 0 & S_7 & 0 \\ S_7 & 0 & -C_7 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.9}$$
$$T_5^4 = \begin{pmatrix} C_5 & -S_5 & 0 & L_3\,C_5 \\ S_5 & C_5 & 0 & L_3\,S_5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad T_8^7 = \begin{pmatrix} C_8 & -S_8 & 0 & L_5\,C_8 \\ S_8 & C_8 & 0 & L_5\,S_8 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T' = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} n'_x & s'_x & a'_x & p'_x \\ n'_y & s'_y & a'_y & p'_y \\ n'_z & s'_z & a'_z & p'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = T_1^8 \tag{A.10}$$

$$p' = \begin{pmatrix} L_3\,C_8\,S_6\,S_7 - L_4\,C_7\,C_8 - L_3\,C_6\,C_7\,C_8 - L_5 \\ S_8(\,L_3\,C_{67} + L_4\,C_7) \\ -L_3\,S_{67} - L_4\,S_7 \\ 1 \end{pmatrix} \tag{A.11}$$

$$p'_x = L_3\,C_8\,S_6\,S_7 - L_4\,C_7\,C_8 - L_3\,C_6\,C_7\,C_8 - L_5 \tag{A.12}$$

Equation A.12 can be reformulated using rule A.1 on page 85, which results in equation A.13, that are on the same form as equations A.14 and A.15.

$$p'_x + L_5 = -\,C_8\,(L_3\,C_{67} + L_4\,C_7) \tag{A.13}$$
$$p'_y = S_8\,(L_3\,C_{67} + L_4\,C_7) \tag{A.14}$$
$$p'_z = -\,L_3\,S_{67} - L_4\,S_7 \tag{A.15}$$

By squaring and adding equations A.13, A.14 and A.15, and then isolating for C_6 results in equation A.16.

$$C_6 = \frac{-{L_3}^2 - {L_4}^2 + {L_5}^2 + 2\,L_5\,p'_x + {p'_x}^2 + {p'_y}^2 + {p'_z}^2}{2\,L_3\,L_4} \tag{A.16}$$

Equation A.16 can be rewritten as in equation A.17

$$C_6 = \frac{-L_3^2 - L_4^2 + (L_5 + p'_x)^2 + {p'_y}^2 + {p'_z}^2}{2\, L_3\, L_4} \tag{A.17}$$

Then by applying rule A.3, S_6 can be found, as shown in equation A.18

$$S_6 = \pm\sqrt{1 - \frac{(-L_3^2 - L_4^2 + (L_5 + p'_x)^2 + {p'_y}^2 + {p'_z}^2)^2}{2^2\, L_3^2\, L_4^2}} \tag{A.18}$$

Now θ_6 can be calculated by using $atan2(S_6, C_6)$, which is an arc-tangent function that returns $tan^{-1}(\frac{S_6}{C_6})$ adjusted to the proper quadrant.

$$\theta_6 = atan2\,({S_6}^2,\, C_6) \tag{A.19}$$

By squaring equations A.13 and A.14, adding and expanding, the resulting equation A.20 is derived.

$$C_7\,(C_6\, L_3 + L_4) - S_6\, S_7\, L_3 = \pm\sqrt{(p'_x + L_5)^2 + {p'_y}^2} \tag{A.20}$$

Then by expanding equation A.15, equation A.21 can be obtained.

$$-\, p'_z = S_7\,(L_4 + L_3\, C_6) + L_3\, C_7\, S_6 \tag{A.21}$$

$$\begin{aligned} k_1 &= L_4 + L_3\, C_6 \\ k_2 &= L_3\, S_6 \end{aligned} \tag{A.22}$$

Using k_1 and k_2 a new variable r is defined in equation A.23.

$$r = \sqrt{(L_5 + p'_x)^2 + {p'_y}^2 + {p'_z}^2} \tag{A.23}$$

Another variable γ can be defined from k_1 and k_2 as:

$$\gamma = atan2((L_3\, S_6), (L_4 + L_3\, C_6)) \tag{A.24}$$

Let $(L_4 + L_3C_6)) = rC_\gamma$ and $(L_3S_6)) = rS_\gamma$, and substituting them into equations A.20 and A.21 gives:

$$rC_{7\gamma} = \pm\sqrt{(p'_x + L_5)^2 + {p'_y}^2} \tag{A.25}$$

$$rS_{7\gamma} = -p'_z \tag{A.26}$$

Then by dividing equation A.26 by A.25, results in $tan(\theta_7 + \gamma)$, which by using rule A.4 gives:

$$\frac{S_7 + atan2((L_3\, S_6), (L_4 + L_3\, C_6))}{C_7 + atan2((L_3\, S_6), (L_4 + L_3\, C_6))} = -\frac{p'_z}{\sqrt{(L_5 + p'_x)^2 + {p'_y}^2}} \tag{A.27}$$

Then from equation A.27, θ_7 can be defined as:

$$\theta_7 = atan2(-p'_z, \pm\sqrt{(p'_x + L_5)^2 + p'^2_y}) - \gamma \tag{A.28}$$

In order to find θ_8 equation A.14 are divided by A.12 resulting in:

$$-\frac{S_8}{C_8} = \frac{p'_y}{L_5 + p'_x} \tag{A.29}$$

Then using rule A.4, θ_8 can be derived:

$$\theta_8 = atan2((p'_y), (-L_5 - p'_x)) \tag{A.30}$$

$$G_{8\to2} = \begin{cases} G_8^{LHS} = G_8^{RHS} \\ G_7^{LHS} = G_7^{RHS} \\ G_6^{LHS} = G_6^{RHS} \\ G_5^{LHS} = G_5^{RHS} \\ G_4^{LHS} = G_4^{RHS} \\ G_3^{LHS} = G_3^{RHS} \\ G_2^{LHS} = G_2^{RHS} \end{cases} \tag{A.31}$$

$$\begin{aligned} G_n^{RHS} &= T_7^8\, T_6^7 \ldots T_{9-n}^{10-n} \\ G_8^{RHS} &= T_7^8\, T_6^7\, T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2 \\ G_7^{RHS} &= T_7^8\, T_6^7\, T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2 \\ G_6^{RHS} &= T_7^8\, T_6^7\, T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2 \\ G_5^{RHS} &= T_7^8\, T_6^7\, T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2 \\ G_4^{RHS} &= T_7^8\, T_6^7\, T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2 \\ G_3^{RHS} &= T_7^8\, T_6^7\, T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2 \\ G_2^{RHS} &= T_7^8\, T_6^7\, T_5^6\, T_4^5\, T_3^4\, T_2^3\, T_1^2 \end{aligned} \tag{A.32}$$

$$G_8^{RHS} = T_1^8 = \begin{bmatrix} n'_x & s'_x & a'_x & p'_x \\ n'_y & s'_y & a'_y & p'_y \\ n'_z & s'_z & a'_z & p'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.33}$$

$$G_n^{LHS} = G_n^{RHS\prime}\, T_1^8 \tag{A.34}$$

This is visualised as:

$$\begin{aligned}
G_8^{LHS} &= T_1^8 \\
G_7^{LHS} &= T_8^7\, T_1^8 \\
G_6^{LHS} &= T_8^7\, T_7^6\, T_1^8 \\
G_5^{LHS} &= T_8^7\, T_7^6\, T_6^5\, T_1^8 \\
G_4^{LHS} &= T_8^7\, T_7^6\, T_6^5\, T_5^4\, T_1^8 \\
G_3^{LHS} &= T_8^7\, T_7^6\, T_6^5\, T_5^4\, T_4^3\, T_1^8 \\
G_2^{LHS} &= T_8^7\, T_7^6\, T_6^5\, T_5^4\, T_4^3\, T_3^2\, T_1^8
\end{aligned} \tag{A.35}$$

Only G_7 is used as the remaining thetas can be found using this equation. The left side of G_7 is seen in equation A.36.

$$G_7^{LHS} = \begin{bmatrix} n'_x\, C_8 - n'_y\, S_8 & s'_x\, C_8 - s'_y\, S_8 & a'_x\, C_8 - a'_y\, S_8 & C_8\,(L_5 + p'_x) - p'_y\, S_8 \\ n'_y\, C_8 + n'_x\, S_8 & s'_y\, C_8 + s'_x\, S_8 & a'_y\, C_8 + a'_x\, S_8 & S_8\,(L_5 + p'_x) + p'_y\, C_8 \\ n'_z & s'_z & a'_z & p'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.36}$$

The right side of G_7, called G_7^{RHS} can be seen in equation A.37, and consists of the transformation T_1^7:

$$G_7^{RHS} = \begin{bmatrix} C_2\, C_4\, C_{567} - S_2\, S_{567} & S_2\, C_4\, C_{567} + C_2\, S_{567} & S_4\, C_{567} & -L_3\, C_{67} - L_4\, C_7 \\ -C_2\, S_4 & -S_2\, S_4 & C_4 & 0 \\ C_2\, C_4\, S_567 + S_2\, C_{567} & S_2\, C_4\, S_{567} - C_2\, C_{567} & S_4\, S_{567} & -L_3\, S_{67} - L_4\, S_7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.37}$$

Then by comparing elements (2,3) of the left and right side of G_7 gives:

$$C_4 = a_y\, C_8 + a_x\, S_8 \tag{A.38}$$

which by using rule A.3, gives S_4 from:

$$\sqrt{1 - (a_y\, C_8 + a_x\, S_8)^2} \tag{A.39}$$

In order to find θ_4 equation A.39 are divided by A.38 resulting in:

$$-\frac{S_4}{C_4} = \frac{\sqrt{1 - (a_y\, C_8 + a_x\, S_8)^2}}{a_y\, C_8 + a_x\, S_8} \tag{A.40}$$

Then using rule A.4, θ_4 can be derived:

$$\theta_4 = atan2(S_4, C_4) = atan2(\sqrt{1 - (a_y\, C_8 + a_x\, S_8)^2}, a_y\, C_8 + a_x\, S_8) \tag{A.41}$$

By comparing the elements of $(2,1)$ and $(2,2)$ of G_7^{LHS} and G_7^{RHS}, two equations are given as:

$$S_2S_4 = -S_{34}S_2 \tag{A.42}$$

$$C_2S_4 = -S_{34}C_2 \tag{A.43}$$

By dividing these two equations results in $tan(\frac{S_2}{C_2})$, from which the joint solution for θ_2 can be obtained as:

$$\theta_2 = atan2(S_2, C_2) \tag{A.44}$$

Then by comparing the elements of $(1,3)$ and $(3,3)$ of G_7^{LHS} and G_7^{RHS}, result in two equations, which if divided by each other gives:

$$\frac{C_{567}}{S_{567}} \tag{A.45}$$

by which the joint angle θ_{567} is given as:

$$\theta_{567} = atan2(C_{567}, S_{567}) \tag{A.46}$$

From this the joint solution θ_5 is given as:

$$\theta_5 = \theta_{567} - \theta_6 - \theta_7 \tag{A.47}$$

A.4 LIPM simulation

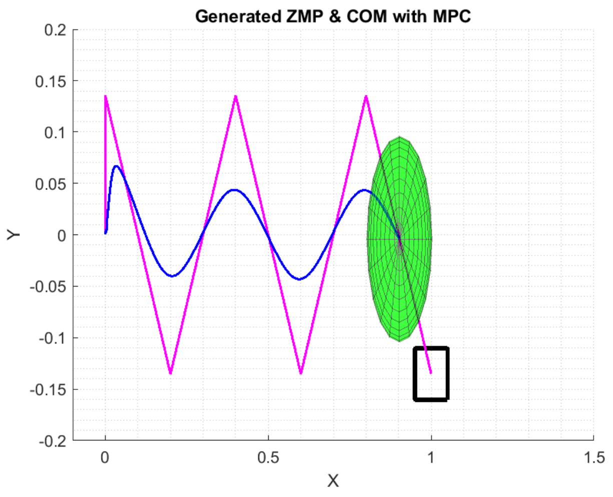


Figure A.2 LIPM simulation walking speed of 0.4 m/s with a step length of 0.2 m. Blue line indicate COM and purple is foot ref

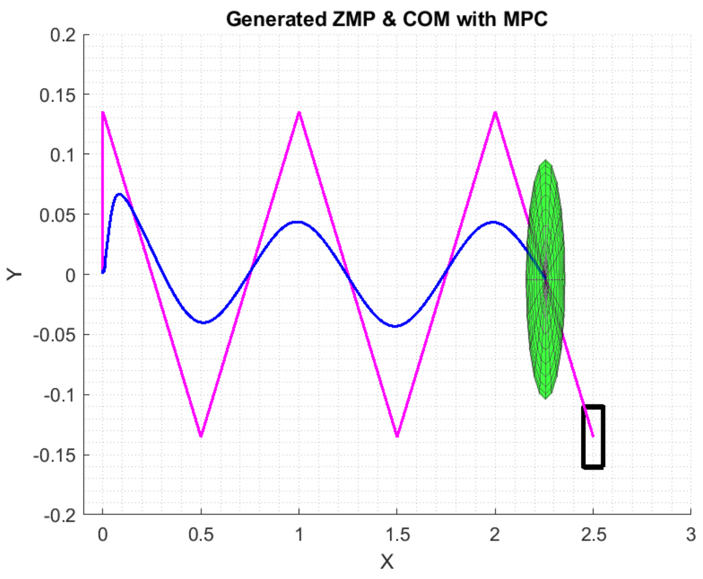


Figure A.3 LIPM simulation walking speed of 1.0 m/s with a step length of 0.5 m. Blue line indicate COM and purple is foot ref

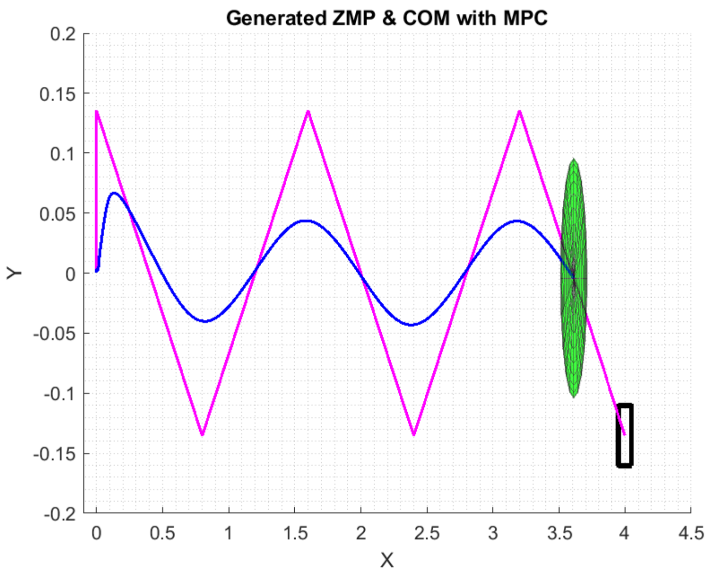


Figure A.4 LIPM simulation walking speed of 1.6 m/s with a step length of 0.8 m. Blue line indicate COM and purple is foot ref

Bibliography

[1] C. Laschi *et al.*, “Grasping and manipulation in humanoid robotics”, Scuola Superiore Sant’Anna, Centro INAIL RTR, Italy, Tech. Rep., 2014-05.

[2] (2012-07-27). Thirteen advanced humanoid robots for sale today. Accessed: 2020-03-30, [Online]. Available: `https://www.smashingrobotics.com/thirteen-advanced-humanoid-robots-for-sale-today/`.

[3] (). Hubo 2. Accessed: 2020-03-30, [Online]. Available: `https://robots.ieee.org/robots/hubo/`.

[4] (2020-04-06). Atlas. Accessed: 2020-05-11, [Online]. Available: `https://www.bostondynamics.com/atlas`.

[5] S. Fengy, X. Xinjilefu, C. G. Atkeson, and J. Kim, “Robust dynamic walking using online foot step optimization”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 5373–5378.

[6] (2020-03-23). Agility robotics: Digit & cassie. Accessed: 2020-05-11, [Online]. Available: `https://www.agilityrobotics.com/robots#cassie`.

[7] M. Reher Jacob; Wen-Loong and A. D. Ames, “Dynamic walking with compliance on a cassie bipedal robot”, *European Control Conference (ECC)*, 2019-04. [Online]. Available: `https://arxiv.org/abs/1904.11104`.

[8] (2020-04-06). Asimo specifications. Accessed: 2020-05-11, [Online]. Available: `https://asimo.honda.com/asimo-specs/`.

[9] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, “The intelligent asimo: System overview and integration”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2002, 2478–2483 vol.3.

[10] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, “Footstep planning for the honda asimo humanoid”, in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 629–634.

[11] (2013-11-30). Biped humanoid robot group wabian-2r. Accessed: 2020-05-11, [Online]. Available: `http://www.takanishi.mech.waseda.ac.jp/top/research/wabian/`.

[12] S. Lohmeier, T. Buschmann, and H. Ulbrich, "Humanoid robot lola", *IEEE International Conference on Robotics and Automation Kobe International Conference Center*, pp. 775–780, 2009.

[13] S. H. Collins, "Dynamic walking principles applied to human gait", PhD thesis, University of Michigan, 2008.

[14] B. F. Mentiplay, M. Banky, R. A. Clark, M. B. Kahn, and G. Williams, "Lower limb angular velocity during walking at various speeds", *Gait & Posture*, vol. 65, pp. 190–196, 2018.

[15] K. Kanjanapas and M. Tomizuka, "7 degrees of freedom passive exoskeleton for human gait analysis: Human joint motion sensing and torque estimation during walking", *6th IFAC Symposium on Mechatronic Systems, The International Federation of Automatic Control*, vol. 46, pp. 285–292, 2013-04.

[16] N. D. Nordin, A. G. Muthalif, and M. K. M. Razali, "Control of transtibial prosthetic limb with magnetorheological fluid damper by using a fuzzy pid controller", *Low Frequency Noise, Vibration and Active Control*, 2018.

[17] N. Hamilton, W. Weimar, and K. Luttgens, *Kinesiology : scientific basis of human motion.—12th ed.* The McGraw-Hill Companies, Inc., 2012, Appendix B: Joint Range of Motion.

[18] C. L. Brockett and G. J. Chapman, "Biomechanics of the ankle", 2016, Accessed: 2020-05-11. [Online]. Available: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4994968/`.

[19] T. Buschmann, A. Ewald, H. Ulbrich, and A. B¨uschges, "Event-based walking control – from neurobiology to biped robots", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1793–1800, 2012.

[20] (2009). Density of selected solids. Accessed: 2020-05-11, [Online]. Available: `https://www.engineeringtoolbox.com/density-solids-d_1265.html`.

[21] (2020-05-05). Pla vs. abs: What's the difference? Accessed: 2020-05-11, [Online]. Available: `https://www.3dhubs.com/knowledge-base/pla-vs-abs-whats-difference/#strength`.

[22] (). Ustepper s. Accessed: 2020-05-27, [Online]. Available: `https://ustepper.com/store/ustepper-boards/27-46-ustepper-s.html#/28-connector-dc_jack/31-stepper_motor-no_stepper_motor`.

[23] (2020-05-29). Load cell - 10kg, straight bar (tal220). Accessed: 2020-05-29, [Online]. Available: `https://www.sparkfun.com/products/13329`.

[24] (2013-08-25). What's the difference between dc, servo & stepper motors? Accessed: 2020-05-27, [Online]. Available: `https://thepihut.com/blogs/raspberry-pi-roundup/whats-the-difference-between-dc-servo-amp-stepper-motors`.

[25] (2020-04-20). Stealthchop torque comparison. Accessed: 2020-05-11, [Online]. Available: https://www.trinamic.com/fileadmin/assets/Support/Appnotes/AN021-stealthChop_Performance_comparison.pdf.

[26] (2011). Qmot stepper motors. Accessed: 2020-05-11, [Online]. Available: https://www.elfadistrelec.dk/Web/Downloads/_m/an/QSH4218_eng_man.pdf?pid=%5C$product.code.

[27] (2016-06-20). Mpu-9250 product specification revision 1.1. Accessed: 2020-05-11, [Online]. Available: https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf.

[28] (2017-05-17). Aeat-8800-q24 encoder datasheet. Accessed: 2020-05-11, [Online]. Available: https://docs.broadcom.com/doc/pub-005892.

[29] (2006). 8-/6-/4-channel das with 16-bit, bipolar input, simultaneous sampling adc. Accessed: 2020-05-28, [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/AD7705_7706.pdf.

[30] (2020). Esp32 datasheet. Accessed: 2020-04-09, [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[31] (2020-05-08). Turnigy 2200mah 3s 25c lipo pack. Accessed: 2020-05-11, [Online]. Available: shorturl.at/cdBMX.

[32] R. M. ALEXANDER, "Energetics and optimization of human walking and running:the 2000 raymond pearl memorial lecture", *AMERICAN JOURNAL OF HUMAN BIOLOGY*, pp. 641–648, 2002.

[33] L. Wang, *Model Predictive Control System Design and ImplementationUsing MATLAB*. Springer, 2009.

[34] J. J. Craig, *Introduction to Robotics: Mechanics and Control, Third Edition*. Edinburg Gate, Harlow, Essex, Great Britain: Pearson, 2014.

[35] (2014-03-21). Classic-dhparameters.png. Accessed: 2020-05-11, [Online]. Available: https://commons.wikimedia.org/wiki/File:Classic-DHparameters.png.

[36] H. Muhammad A. Ali, A. Park, and C. S. G. Lee, "Closed-form inverse kinematic joint solution for humanoid robots", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 704–709, 2010-10, Taipei, Taiwan.

[37] R. P. Paul and B. Shimano, "Kinematic control equations for simple manipulators", in *IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes*, 1978, pp. 1398–1406.

[38] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to Humanoid Robotics*. Springer, 2005.

[39] (2020-04-12). Friction. Accessed: 2020-06-03, [Online]. Available: `https://en.wikipedia.org/wiki/Friction`.

[40] (2020-04-21). Gift wrapping algorithm. Accessed: 2020-05-11, [Online]. Available: `https://en.wikipedia.org/wiki/Gift_wrapping_algorithm`.

[41] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum mode a simple modeling for a biped walking pattern generation", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 239–246, 2001.

[42] S. KAJITA *et al.*, "Biped walking pattern generation by using preview control of zero-moment point", *IEEE International Conference on Robotics & Automation Taipei, Taiwan*, vol. 2, pp. 1620–1626, 2003-09.

[43] K. E. Chlouverakis and J. Sprott, "Chaotic hyperjerk systems", *Chaos, Solitons and Fractals 28 (2006)*, pp. 739–746, 2005.

[44] J. Maciejowski, *Predictive Control with Constraints*. Pearson Education, 2000.

[45] (2020). Mathworks: Specify constaints. Accessed: 2020-05-11, [Online]. Available: `https://se.mathworks.com/help/mpc/ug/specifying-constraints.html`.

[46] (2004). Friction and friction coefficients. Accessed: 2020-05-25, [Online]. Available: `https://www.engineeringtoolbox.com/friction-coefficients-d_778.html`.

[47] (2019-12-20). Mathworks student competitions team (2020). matlab and simulink robotics arena: Walking robot. Accessed: 2020-06-01, [Online]. Available: `https://www.github.com/mathworks-robotics/msra-walking-robot`.