Faculty of Engineering and Built Environment

School of Engineering

Mechanical Engineering Project B

Semester 2 - 2022 MECH4841 B

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

FINAL YEAR PROJECT

PROJECT TITLE

NUgus Walking Gaits

NAME & STUDENT NUMBER

Thomas O'Brien c3304696

SUPERVISOR

Joel Ferguson

Final Year Project Part B

2021

Thomas O'Brien [1]

[1] *Student of Mechatronics Engineering,*
The University of Newcastle, Callaghan, NSW 2308, AUSTRALIA
Student Number: 3304696
E-mail: thomas.obrien@uon.edu.au

Covid Impact Summary

Covid-19 considerably impacted the outcomes of this project. The initial goal of the project was to implement walking routines on the real robotics hardware. Due to the locks downs and restrictions imposed by Covid-19, access to the hardware and lab throughout the 2nd semester was severely limited or entirely impossible. I was granted limited access towards the end of the semester which allowed system identification of the hardware to be conducted. During the granted extension period, physical implementation of the walking routines was intended to take place, however, it was discovered at this point that hardware related problems out of the scope of this project prevented any physical experiments to take place. Furthermore, the social distancing restrictions prevented any form of collaboration during the semester with NUbots team members more equipped in areas of hardware maintenance, further preventing the ability to resolve issues holding back the project.

To combat the impact of Covid-19, a large focus of the project was put on physics simulation modelling and experimentation. While there is expected to be a considerable simulation to reality gap, large emphasis was placed on accurately modelling the real hardware such that the work completed could eventually be transferred to the real robot.

I am very grateful for my supervisors and course coordinators willingness to grant an extension for this project and provide lab access where possible. However, due to prior arranged full-time work commitments the amount of free time I had during the extension period was severely limited. Furthermore, during the extension period I tested positive for Covid-19 which hindered my ability to make any progress on the project while I recovered.

Declaration

I declare that this thesis is my own work unless otherwise acknowledged and is in accordance with the University's academic integrity policy available from the Policy Library on the web at http://www.newcastle.edu.au/policylibrary/000608.html

I certify that this assessment item has not been submitted previously for academic credit in this or any other course.

I acknowledge that the Faculty of Engineering may, for the purpose of assessing this thesis:

- PReproduce this thesis and provide a copy to another member of the Faculty; and/or
- Communicate a copy of this assessment item to a plagiarism checking service (which may then retain a copy of the item on its database for the purpose of future plagiarism checking).
- Submit the assessment item to other forms of plagiarism checking.
- Where appropriate, provide a copy of this thesis to other students where that student is working in/on a related project.

I further acknowledge that the Faculty of Engineering may, for the purpose of sector wide Quality Assurance:

- Reproduce this thesis to provide a copy to a member of another engineering faculty.

I certify that the electronic versions of this thesis I have submitted are identical to this hardcopy version. Furthermore, I have checked that the electronic version is complete and intact on the submitted location.

Student Name: **Thomas O'Brien**

Signature: 

I have not left a mess in the lab declaration page.

I, Thomas O'Brien, having completed the project "NUgus Walking Gaits", and using laboratory spaces within the ES building on Callaghan campus, hereby submit that the area has been cleaned to level equal to or better than when I commenced my work. All items I borrowed have been returned, and the area is now suitable for inspection and sign off by the appropriate laboratory manager.

Student Name: **Thomas O'Brien**

Signature: 

Dot Point Summary

Through the completetion of this project I have gained a deep understanding in the following:

- Mathematics for modelling and controlling robotics platforms
- System identification of physical robotic hardware
- Control theory
- Optimisation
- Robotic systems and software design

The following dot points outline the objectives achieved:

- Performed system identification to determine the length, mass and effective CoM of each of the limbs.
- Developed software pipeline to generate URDF file containing model parameters stored within Onshape model
- Developed a kinematic model (2D and 3D) for the walking robot and created a Matlab visualisation.
- Developed a model and control interface for the of NUgus platform in the WeBots simulation environment
- Developed modular walking gait generation software for generating gait patterns within MATLAB
- Developed C++ implementations of forward and inverse kinematics and conducted benchmarks
- Implemented both a quasi-static and dynamic walking routine on the NUgus platform (open-loop) in the simulation environment WeBots.

Abstract

This project explores the design and implementation of both static and dynamic walking gaits for the NUgus humanoid robotics platform, a robot which competes in the international robotic soccer competition, Robocup. Robotic soccer is a fast-paced dynamic environment which requires highly responsive, stable and dynamic bipedal locomotion.

Static walking gaits are generated via the inclusion of non-linear constraints in a numerical optimisation based inverse kinematics solver. Dynamic walking gaits are generated via the use of a simple linear inverted pendulum model (LIPM) and a preview controller, which outputs dynamically balanced center of mass trajectories. Physics simulation results show that both the proposed methods are capable of providing stable and reliable walking gaits.

To transfer the simulation results onto the real platform, system identification was conducted for the NUgus robotics platform, utilising a combination of CAD software and real world experiments to estimate kinematic and inertial properties.

Acknowledgements

I would like to express my sincere thanks to my supervisor Joel Ferguson for his support and guidance throughout this project.

Contents

List of Figures

List of Tables

1. Introduction and project scope

For many decades robots have been successfully utilised in industrial settings, performing repetitive constrained tasks with endurance, speed, and high precision. It is estimated that 2.7 million industrial robots were operational in the year 2020 [1]. The next generation of robots are expected to interact with their environments in increasingly more complex ways such as performing unfamiliar tasks and work collaboratively with other robots or human workers. These increased capability demands require increases to overall performance, construction at smaller scales, decreased response times and reduced cost and weight. The sophistication level of these robots should allow them to safely operate in shared environments and interact with humans in homes, workplaces, and communities, providing support in services, entertainment, education and healthcare [3],[27].

One potential robot suitable for integrating into these environments are humanoid robots, which are human-like robots, as shown in Figure 1. The field of humanoid robotics is motivated by the humanoid form factor, as it is very suited for functioning in the human environment. The structure of humanoids allows them to be more easily accepted by the general public, and their kinematic structure allows them to easily navigate typical obstacles of the human environment, which makes them good candidates for elderly care, nursing and more. Furthermore, one of the most interesting aspects of a humanoid is its locomotion technique: walking. General robot motion is typically wheeled motion, however, legged motion can be far more efficient for navigating difficult rocky or soft terrain. This can be especially valuable, for example, in disaster response situations, allowing robots to navigate dangerous environments in search and rescue missions preventing the need to place emergency response individuals at risk.

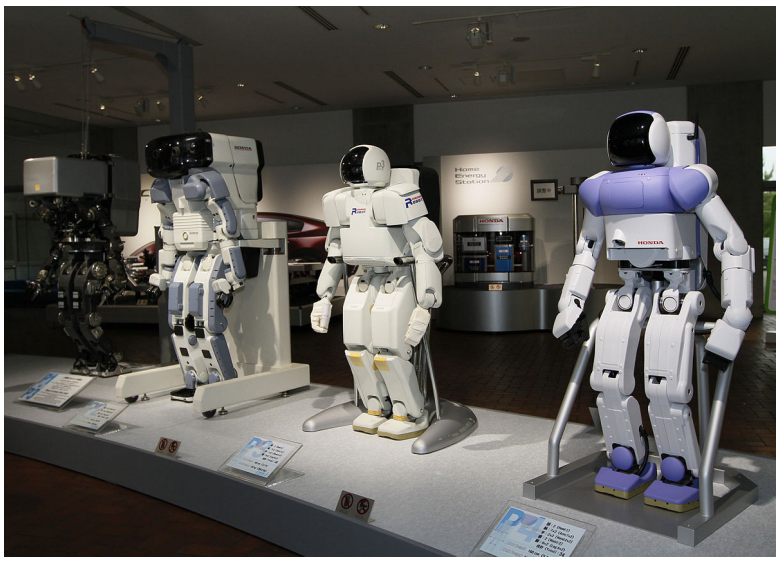

Figure 1: Humanoid robots.

1.1. Humanoids and bipedal locomotion

For many decades there has been a large interest in the robotics community in the field of humanoid and legged robotics. A variety of different legged robots exist, often distinguished by the number of legs. If a legged robot has two legs, it is often called a biped which mimics human-like walking.

Honda
Automobile manufacturer Honda has been at the frontier of significant advancements in bipedal locomotion, beginning in 1986 with a secret research program. Due to the commercial nature of Honda's work, very few scientific papers have been published on their designs, however, the scientific community has been highly influenced by the patents submitted by Honda. Many robotic researchers have adopted the approach of Honda, using six DoF legs, force/torque sensors in the feet, an IMU in the torso and electric motors in combination with harmonic drive gears [5]. Honda's most impactful design has been the Asimo robot, which stands for "Advanced Step in Innovative Mobility". Asimo is a bipedal robot developed with a goal to coexist with and be useful to people since it was first introduction in 2000 [15]. The walking patterns of the Asimo are generated online in real time, with smooth turning and foot placement planning to allow walking in any direction [14]. Trajectory planning and control based on the lumped mass model and controlling upper body orientation through contact moments has been widely adopted in the research community as a result of the Asimo platform [5].

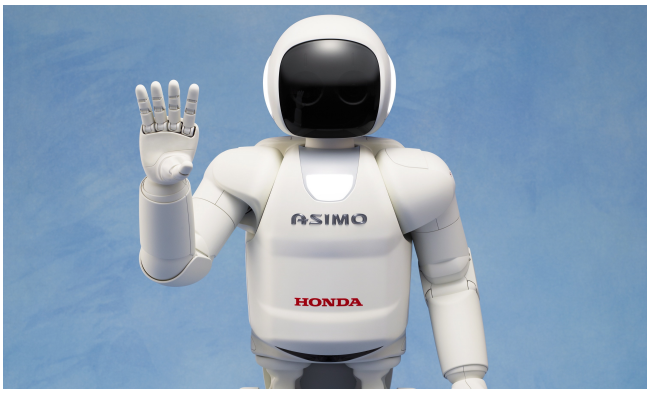

Figure 2: Asimo [9].

Boston Dynamics

Boston Dynamics have had a large influence on the humanoid community in recent years with the development of the Atlas platform, shown in Figure 3 [19]. The design and production of Atlas was overseen by DARPA, an agency of the United States Department of Defense. In August 2021, Boston Dynamics released a video of Atlas running a parkour course which included balance beams, jumps and vaults. All the computation for control and perception for these routines was done on three onboard computers [4]. A distinguishing feature of the Atlas platform is the hydraulically powered joints. This hydraulic design enables Atlas to deliver high power to any of its 28 hydraulic joints and support its weight of 89kg [12].

Figure 3: Atlas [30].

Passive dynamic walkers

While the majority of robots are built with electrical or hydraulic powered servo motors, not all need joints that are actively actuated in order to walk [11]. These unactuated robots are known as passive-dynamic walkers which are simple mechanical devices, composed of solid parts connected by joints [10]. These robots exploit passive dynamics of there mechanical designs to allow them to walk stably down sloped surfaces. While there is little practicality for these robots, the control community has been able to design walking controllers which mimic the dynamics observed in these mechanisms to produce highly energy efficient walking gaits [11].

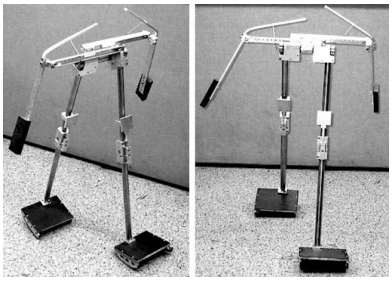

Figure 4: Passive dynamic walker [23].

1.2. NUbots, NUgus and Robocup

The NUbots team consists of students from a range of disciplines across the University of Newcastle and has been conducting robotic research since 2002. The main focus of the team is to develop software for robotic soccer and competes every year in the international RoboCup competition. RoboCup is an international research and education initiative. Its goal is to foster artificial intelligence and robotics research by providing a standard problem where a wide range of technologies and concepts can be integrated and examined in comparison to other teams. The ultimate goal of the RoboCup initiative is "By the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team" [18, 8]. I am honoured to be the lead of the Motion team within NUbots, focusing on motion related areas such as walking, kicking and getting up from falls. The current robot platform used by the NUbots team is the NUgus platform, a modification of the igus Humanoid Open Platform originally developed by Bonn university, shown in Figure 5.

Figure 5: NUgus robot.

The NUbots codebase can be broken down into the categories listed in Figure 6. For this project, the major focus will be the `Motions` component, which deals with any physical motion undertaken by the NUgus platform, such as kicking, getting up and walking. The current implementation of the walk engine for NUgus platform is refereed to as the `Quintic Walk`, which is an open loop controller based on quintic splines, originally developed by Quentin "Leph" Rouxel and team Rhoban [22]. The aim of this project is to implement and compare fundamental stability methods which can then be integrated into the `Motions` system of the NUbots codebase.

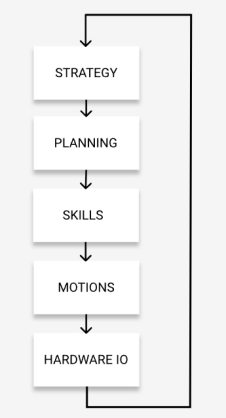

Figure 6: NUbots Overview.

1.3. Project scope

The scope of this project is to implement and compare a quasi-static walking routine and a dynamic walking routine on the NUbots humanoid robot. Implementation of these routines will result in greater control of the robot platform, improving the team's competitiveness in the Robocup competition. The project involves system identification, control, and optimisation tools. The objectives of the project can be broken down as follows:

- Perform system ID to determine the effective CoM of each of the limbs.
- Calibrate the motors and develop an interface for control.
- Develop a kinematic model (2D and 3D) for the walking robot and create a Matlab visualization.
- Implement quasi-static walking routine on the Nugus platform (open-loop).
- Implement zero-moment principle based dynamic walking on the Nugus platform (open-loop).

The thesis is organised as follows. The next Chapter (2) briefs the NUbots and NUgus platform, mathematical notation, terminology and core bipedal locomotion concepts. Chapter 3 describes the modelling and system identification procedure undertaken for the NUgus platform. The fundamentals of gait generation are introduced in Chapter 4. Chapter 5 explores the implementation of a static walking routine, and Chapter 6 investigates the implementation of a dynamic walking routine. Experimental walking results and performances of the employed control algorithms in the simulation environment WeBots are presented in Chapter 7. Finally, conclusions and potential future work are presented in Chapter 10.

2. Background

The following chapter explores the mathematical notation, terminology and some core bipedal locomotion concepts used throughout the report.

2.1. Mathematical notation

2.1.1. Basis

A basis is a maximal set of linearly independent vectors. Given a basis with three vectors b_1, b_2, b_3, any three dimensional vector u can be written as:

$$r = a_1 b_1 + a_2 b_2 + a_3 b_3 \tag{2.1}$$

2.1.2. Vectors

Any vector can be represented as a linear combination of the elements of the basis, expressed as $r^a_{B/A}$ which is the vector of point B from point A expressed in frame a, illustrated in Figure 7

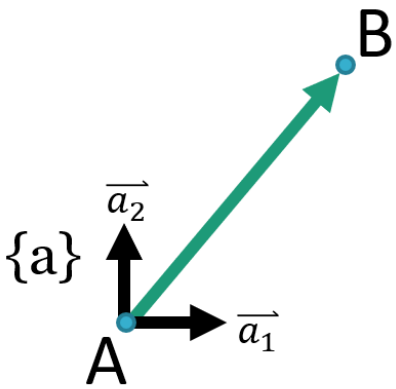

Figure 7: Vector $r^a_{B/A}$.

2.1.3. Coordinate systems and reference frames

A **reference frame** is the perspective from which the motion of a body or an object is described by an observer. A reference frame can be defined by a set of at least 3 non-collinear points in space that are rigidly connected. The term 'space' is also used to refer to reference frames. This terminology is used extensively throughout this report when referring to reference frames such as torso space, support foot space and world space.

A **coordinate system** is a mathematical entity that allows us to establish a one-to-one correspondence between vector magnitudes and scalars called coordinates. A basis defines a coordinate system.

2.1.4. Rotations

Rotations are treated as matrices that rotate the vectors of the one basis into the vectors of another basis b. For example, R_b^a is the rotation matrix that relates the coordinates of a vector in {b} to its components in {a}.

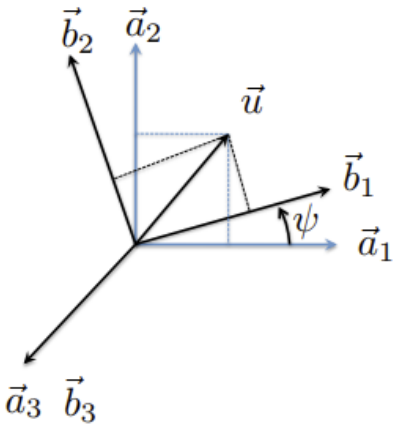

Figure 8: Rotation.

2.1.5. Homogeneous transformations

A homogeneous transformation matrix in three dimensions is a 4x4 matrix containing a **rotation** component and a **translation** component.

If we have some rotation $\mathbf{R}_a^b$ and a translation component $\mathbf{r}_{AB}^b$, then we have a transformation matrix that transforms from space a to space b:

$$\mathbf{H}_a^b = \begin{bmatrix} \mathbf{R}_a^b & \mathbf{r}_{AB}^b \\ \mathbf{0} & 1 \end{bmatrix}$$

2.2. Kinematic chains

Robotic platforms can be modelled as a tree of links and joints. It is common in robotics to utilise a framework known as a direct or forward kinematic model (FKM), which is the mathematical representation of the system. A FKM is a function which given joint variables (q) will output the pose of the end effector. Both revolute and prismatic joints can be incorporated, however, for all the biped FKM's considered in this report, only revolute joints are required. A revolute joint rotates about an axis, and conventionally, the z-axis is chosen.

This mathematical model can be obtained through a simple product of homogeneous transforms between the base and end-effector of a robot. For example, considering the planar manipulator shown in Figure 9, the FKM can be mathematically represented as equation 2.2. This same process can be applied to any n-link manipulator using equation 2.5, greatly reducing the complexity for mathematically modelling robotic systems.

$$H_B^O(q) = H_A^0(q_1)H_B^A(q_2) \tag{2.2}$$

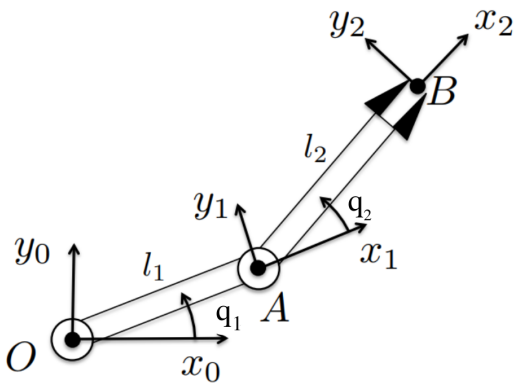

Figure 9: Two-degree-of-freedom planar robotic manipulator. [29]

$$H_n^0(q) = H_1^0(q_1)H_2^1(q_2)\ldots H_n^{n-1}(q_n) \tag{2.3}$$

Additionally, a FKM can have the centre of mass location of each link incorporated in the model, as illustrated in Figure 10. This allows the effective centre of mass of the whole manipulator to be derived. Considering the two degree of freedom manipulator depicted in Figure 10, the effective centre of mass relative the the base $\{0\}$ can be calculated as follows:

First, the transform to each links centre of mass is derived:

$$\begin{aligned} H^0_{m_1} &= R_z(q_1)Tran_x(l_{c1}) \\ H^0_{m_2} = R_z(q_1)Tran_x(l_1)R_z(q_2)Tran_x(l_{c2}) &= H^0_1 H^1_{m_2} \end{aligned} \tag{2.4}$$

where R_z is a 4×4 rotation matrix about z and $Tran_x$ is a 4×4 translation matrix in x.

Next, the effective CoM is calculated using the following equation:

$$r^0_{C/0} = \frac{\sum_{n=1}^{i} r^0_{c_i/0} m_i}{\sum_{n=1}^{i} m_i} \tag{2.5}$$

where $r^p_{c_i/0}$ is the translation component extracted from the transforms of Eq 2.4.

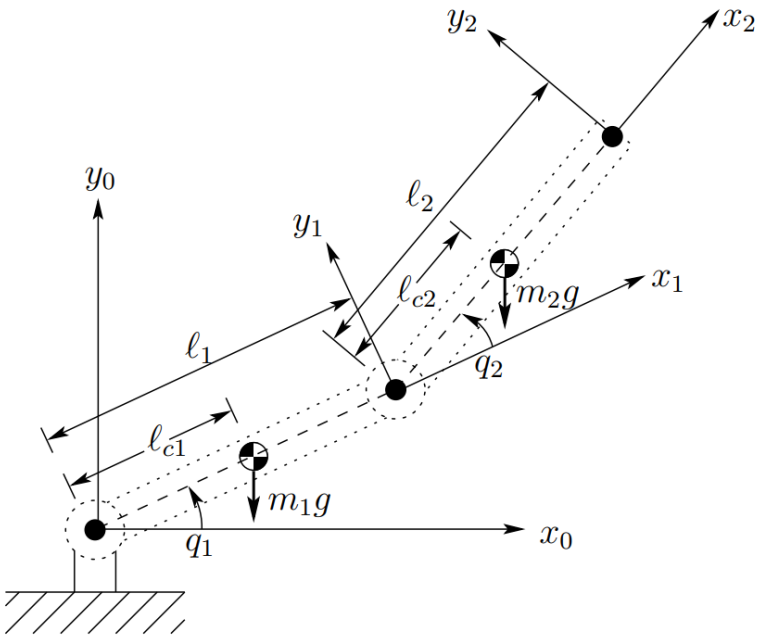

Figure 10: Two-degree-of-freedom.

2.3. Terminology

The following outlines some of the key terms used when discussing gait sourced from [11, 33].

Walk: Walk is defined as: "Movement by putting forward each foot in turn, not having both feet off the ground at once."

Gait: Gait is defined as: "Manner of walking or running".

Periodic gait: If the gait is realized by repeating each step in an identical way, it is a periodic gait.

Double Support (DS): This term is used for situations where the biped has two feet in contact with the floor.

Single Support: (SS): This term is used for situations where the biped has only one contact surface with the floor.

Support Polygon (SP): The Support Polygon is formed by the convex hull of the support points of the biped. This is usually formed by, but not limited to, the feet of the biped.

Support Foot (P): This is the foot of the biped which is in contact with the floor during a step. This will be denoted as P, for 'planted' to resolve the double up of the letter S.

Swing Foot (S): This is the foot of the biped which is raised off the floor during a step.

Step length [m]: This is the distance [m] in the forward's direction between steps.

Step width [m]: This is the width [m] between footsteps.

Step height [m]: This is the peak height [m] the foot is raised between steps.

Step time [s]: This is the time [s] between a single footstep, the period where the swing foot is raised off the floor and placed back down.

2.4. Gait stages

When a biped is in a periodic gait, it can be divided into four stages:

1. Double Support (DS)
2. Left Single Support (LS)
3. Double Support (DS)
4. Right Single Support (RS)

These four phases form the walking gait illustrated in Figure 11.

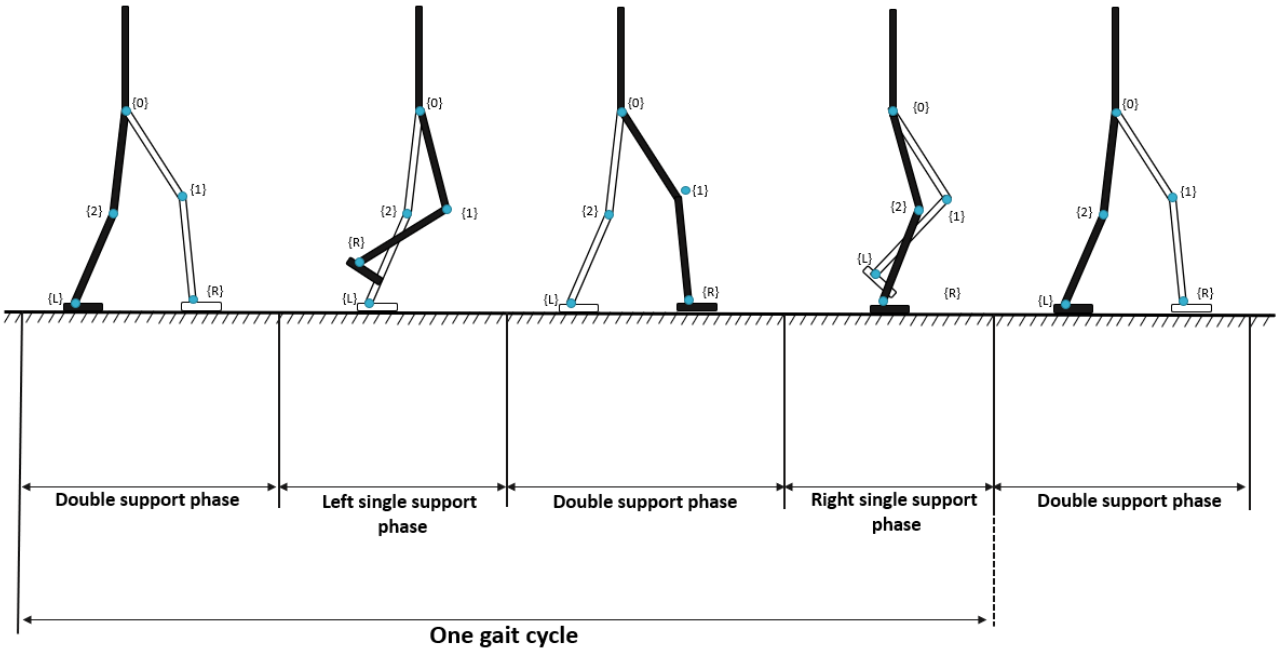

Figure 11: Gait stages.

To simplify, an assumption can be made that the time taken for the switch between the left and right support phase (double support phase) is instantaneous, thus, reducing the gait cycle to simply an alternating state of single left and right support phases, illustrated in Figure 12. Furthermore, it is worth noting that assuming the biped begins with its feet aligned horizontally, a single half step must first be taken to initiate the walking gait.

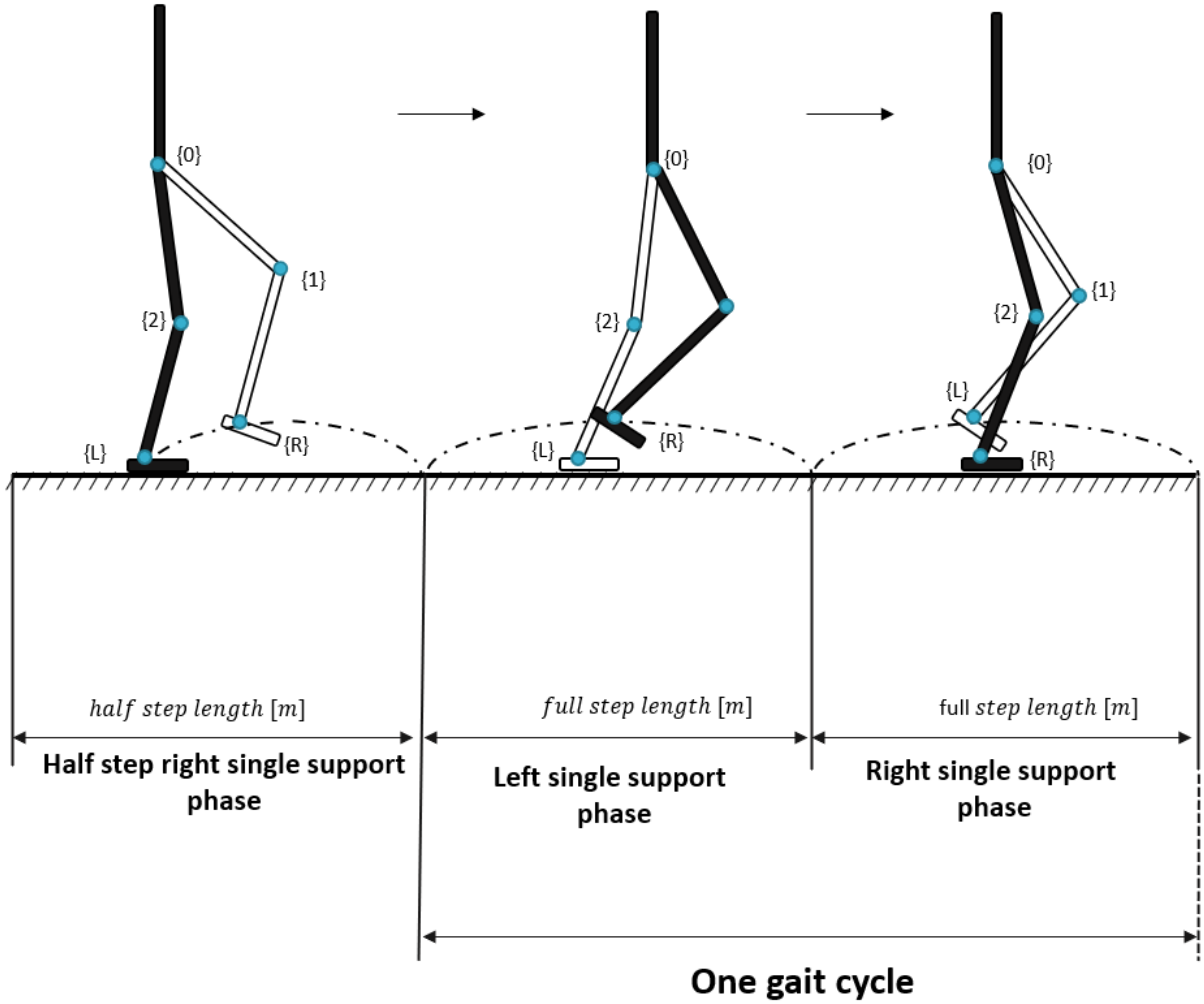

Figure 12: Simplified gait.

2.5. Support polygon

For a rigid object in contact with a fixed environment and acted upon by gravity in the vertical direction, its support polygon is a horizontal region over which the center of mass must lie to achieve static stability. It can be calculated as the convex hull of all the points in contact with the environment. In the context of bipedal locomotion, the support polygon is commonly the region generated due to the contact of the bipeds feet with the ground. A visualisation of the support polygon of the generated by a biped during the single and double support phase is shown in Figure 13. Additionally, the WeBots simulation environment provides a useful tool to visualise the support polygon generated by a biped model in its environment, shown in Figure 14.

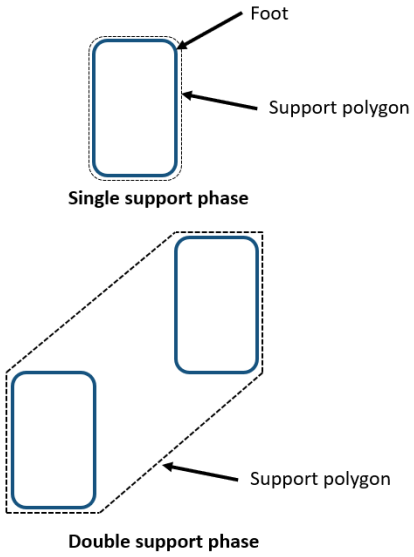

Figure 13: Support polygon

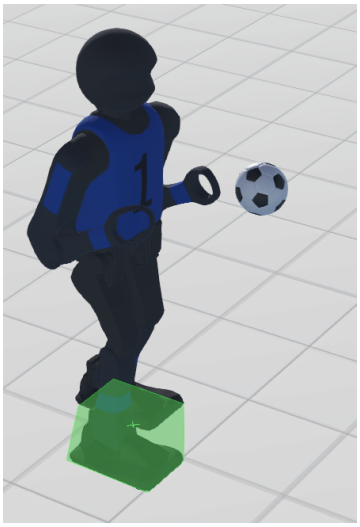

Figure 14: Support polygon WeBots

2.6. Stability Criterion

In order to design a controller that generates gait patterns that do not result in the robot tipping or falling over, a set of stability criterion is required. A stable gait can be defined as, *"A gait where the only contact between the biped and the floor is realized with the soles of the foot or feet, i.e. no other extremity of the biped is in contact with the floor" [11].*

2.6.1. Center Of Mass (CoM) and Floor Projection of CoM (FCoM)

The equivalent Center of Mass (CoM) of a biped is calculated using the following equation 2.6.

$$r^p_{C/P} = \frac{\sum_{n=1}^{i} r^p_{c_i/P} m_i}{\sum_{n=1}^{i} m_i} \tag{2.6}$$

where $r^p_{c_i/P} m_i$ and m_i is vector to the CoM and mass of each link respectively.

The floor projection (FCoM) of the CoM of the biped can be derived by simply taking the horizontal (x and y) components of $r^p_{C/P}$, assuming the support foot (P) is fixed to the floor. If a biped is motionless and the FCoM is within the SP, the biped is deemed stable. However, if the motion of the biped becomes too large, the dynamic forces generated will overcome the static forces causing the robot to fall [11].

2.6.2. Zero Moment Point (ZMP)

A very popular term used in biped locomotion is the term "Zero Moment Point" and is widely known by the acronym ZMP. The ZMP, labelled as P in Figure 15, is similar to the FCoM, however, it takes into account both static and dynamic forces. It is defined as, "the contact point for which the moments due to gravity exactly balance out the moments due to contact with the ground" [11]. The derivations of the ZMP are covered in more detail in Section 6.2.

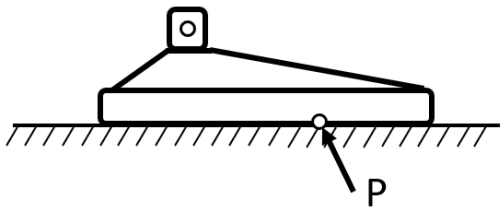

Figure 15: ZMP

2.6.3. Statically Stable Gait

A gait is deemed statically stable if both the FCoM and ZMP remain inside the SP during the entire motion of the gait. Therefore, if during any stage of the motion the movement stopped, the robot would still remain stable [11]. Consequently, this results in gaits that only allow for low walking speeds.

2.6.4. Dynamically Stable Gait

A gait is deemed dynamically stable if the FCoM leaves the SP but the ZMP remains inside the SP. This type of gait allows for stable and faster walking speeds [11].

2.7. Numerical Optimisation

Optimisation is one of the most powerful tools used within Mechatronics and the greater engineering field. Optimisation is the process of finding the "best" solution from a pool of possible candidate solutions [24]. The quality or score of the solution is quantified using a cost function, which can be either minimised or maximised. This search process is completed subject to the model of the system and can have restrictions included, known as constraints. A sensible approach to this problem is to iteratively evaluate the function (model) and take small steps down the slope, referred to commonly as numerical optimisation. The act of computing the gradient and moving down the slope is known as gradient decent, which is visually illustrated in 16. This is a first-order iterative optimisation algorithm for finding a local minimum and requires the function be differentiable.

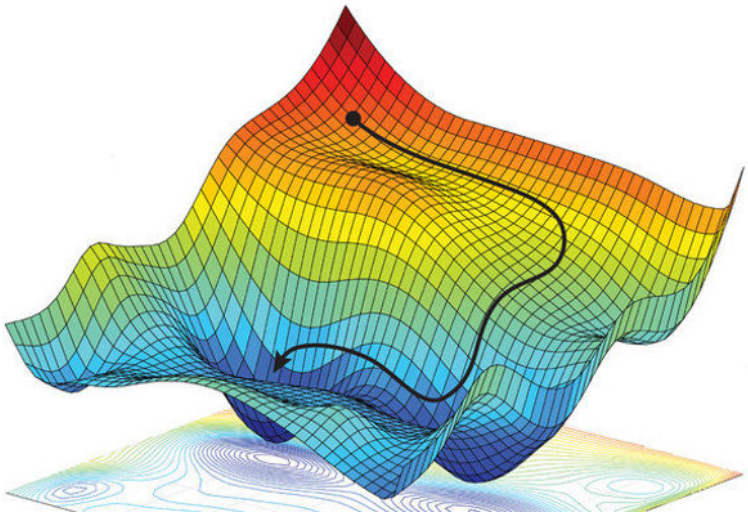

Figure 16: Gradient decent visualized. [2]

An optimisation problem can be mathematically represented as the following:

$$q^* = \underset{q}{\operatorname{argmin}}\, f(q)$$

with the following constraints:

$$\begin{aligned} c(q) &\geqq 0 \\ c_{eq}(q) &= 0 \\ Ax &\leqq b \\ A_{eq}x &= b_{eq} \\ l_b &\leqq x \leqq u_b \end{aligned}$$

where b and b_{eq} are vectors, A and A_eq are matrices, c(q) and $c_{eq}(q)$ are functions that return vectors, and f(q) is a function that returns a scalar. Depending on the optimisation algorithm used, f(q), c(q), and $c_{eq}(q)$ can be nonlinear functions.

3. Modelling and system identification

The following chapter outlines the mathematical modelling and system identification completed for the NUgus hardware platform. The NUgus platform is a biped robot consisting of 20 joints and 21 links. It is a modification of the igus Humanoid Open Platform, with the majority of its components being 3D printed using fused deposition modelling [22]. One of the three currently operational robots has modified metal legs, developed to reduce the impact of flex present in the 3D printed legs, as shown in Figure 17. The system identification completed includes results for both the regular 3D printed legs and the metal variant.

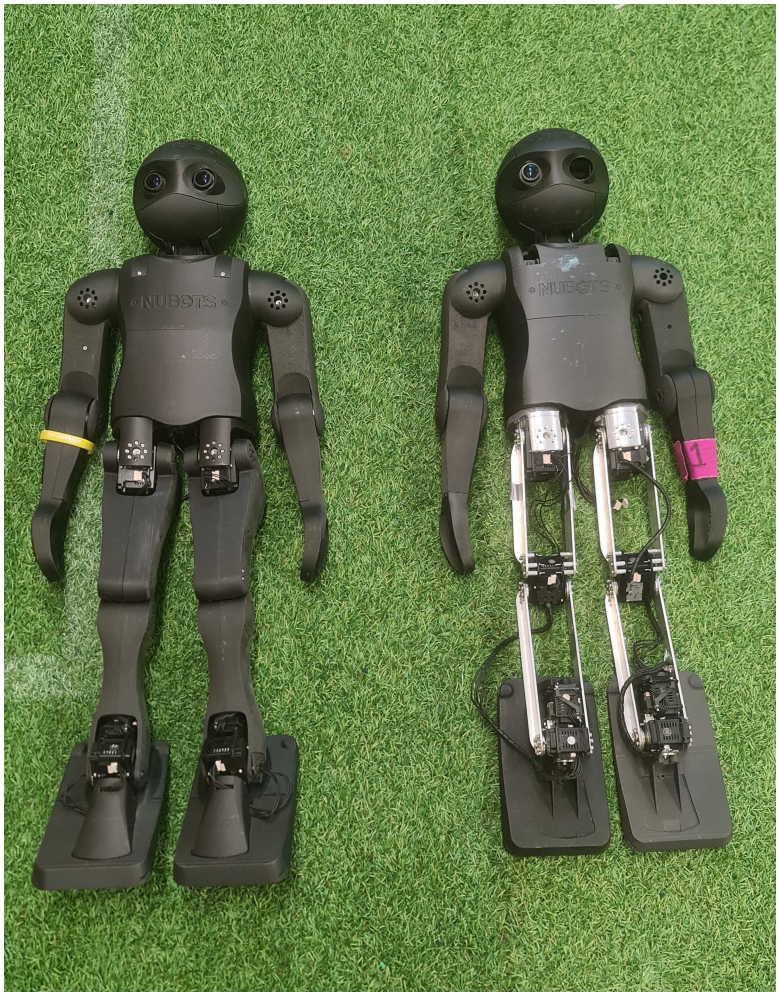

Figure 17: NUgus platform.

3.1. 2D Forward Kinematics

A simplified model of a 2D biped was created, which is utilised as a starting point to validate a variety of core components investigated in this report, such as inverse kinematics and CoM calculations, as it greatly reduces the complexity of the dynamics. A forward kinematic model for this simple 5 link 2D biped robot is illustrated in Figure 18. It has a total of 6 revolute joints that rotate in pitch, with 2 in the hip, 2 for each of the knees and 2 in the ankle. The base of the model is a fixed joint {0} which lies on top of hip pitch joints {1} and {2} connected to the upper body which does not include arms and is modelled as a single link.The model parameters chosen for this simple 2D kinematic biped were arbitrarily chosen and are listed in Table 8.

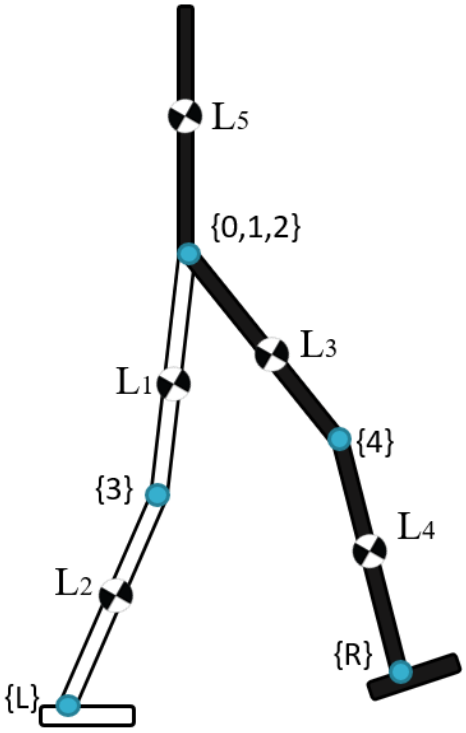

Figure 18: 2D Kinematics.

To calculate the transform from the base {0} to the left foot {L}, the following product of homogeneous transform can be used:

$$H_L^0(q) = H_1^0 H_3^1 H_L^3 \tag{3.1}$$

To calculate the transform from the base {0} to the right foot {R}, the following product of homogeneous transform can be used:

$$H_R^0(q) = H_2^0 H_4^2 H_R^4 \tag{3.2}$$

Considering a situation where the right foot {R} is in the single support phase, the transform from the support foot to swing foot can then be calculated as follows:

$$H_L^R(q) = (H_1^0 H_R^1)^{-1} H_2^0 H_L^2 \tag{3.3}$$

Inversely, the transform from the support foot to swing foot for when the left foot {L} is the support foot can be calculated as follows:

$$H_R^L(q) = (H_1^0 H_R^1)^{-1} H_2^0 H_L^2 \tag{3.4}$$

Table 1: 2D model parameters

Link Name	Length [m]	Mass [kg]
L0	0.2	0.5
L1	0.2	0.25
L2	0.2	0.25
L3	0.2	0.25
L4	0.2	0.25

3.2. 3D Forward Kinematics of NUgus

The simplified 2D FKM was extended to a full 3D kinematic model, which aims to mathematically resemble the NUgus' legs and upper body, illustrated in Figure 19. This model consists of 13 joints and 13 links, with a single link representing the whole upper body, a result of an assumption that the upper body is kept rigid, whereas in reality, the robot has additional links and joints for the upper body. The torso frame {0} is positioned at the midpoint between left and right hip yaw joints and is treated as the base of the FKM.

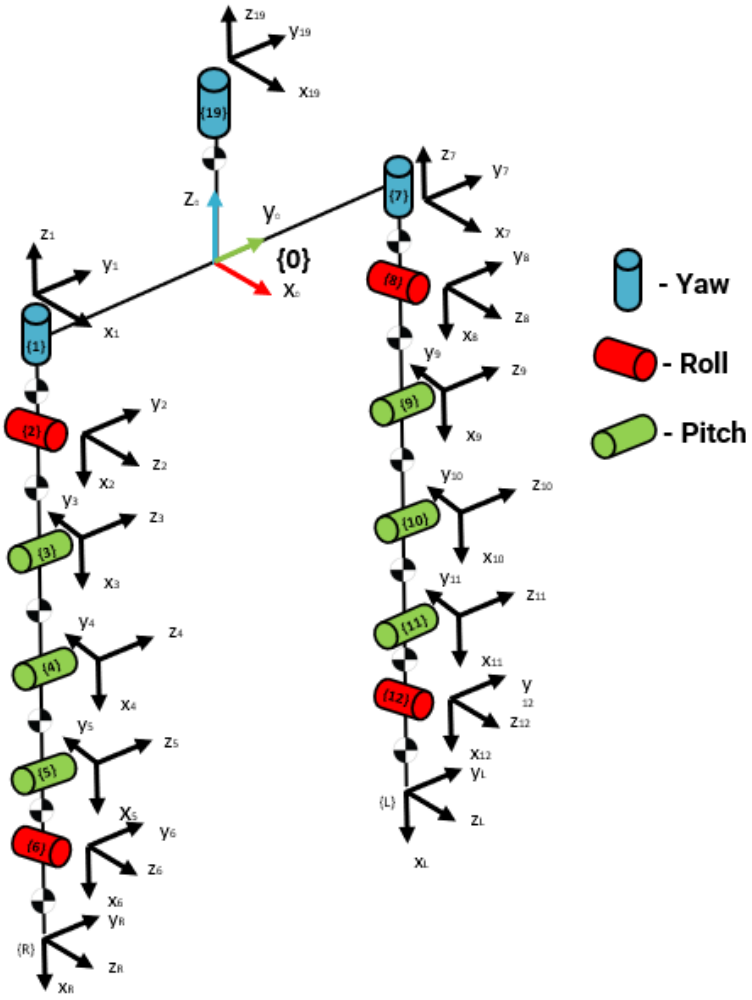

Figure 19: Simple 3D Kinematics.

To calculate the transform from the base {0} to the left foot {L}, the following product of homogeneous transform can be used:

$$H_L^0(q) = H_7^0 H_8^7 H_9^8 H_{10}^9 H_{11}^{10} H_{12}^{11} H_L^{12} \tag{3.5}$$

To calculate the transform from the base {0} to the right foot {R}, the following product of homogeneous transform can be used:

$$H_R^0(q) = H_1^0 H_2^1 H_3^2 H_4^3 H_5^4 H_6^5 H_R^6 \tag{3.6}$$

Considering a situation where the right foot {R} is in the single support phase, the transform from the support foot to swing foot can then be calculated as follows:

$$H_L^R(q) = (H_1^0 H_2^1 H_3^2 H_4^3 H_5^4 H_6^5 H_R^6)^{-1} H_7^0 H_8^7 H_9^8 H_{10}^9 H_{11}^{10} H_{12}^{11} H_L^{12} \tag{3.7}$$

Inversely, the transform from the support foot to swing foot for when the left foot {L} is the support foot can be calculated as follows:

$$H_R^L(q) = (H_7^0 H_8^7 H_9^8 H_{10}^9 H_{11}^{10} H_{12}^{11} H_L^{12})^{-1} H_1^0 H_2^1 H_3^2 H_4^3 H_5^4 H_6^5 H_R^6 \tag{3.8}$$

Additionally, a complete FKM for the NUgus was derived as illustrated in Figure 20. This FKM consists of 20 joints and 21 links. The lower body of the kinematic structure is identical to Figure 19, however the upper bodies arms and neck joints are included. Furthermore, the model includes the pose of the two FLIR Blackfly S cameras located in the head of the robot ({16} and {15}). The inclusion of the upper body links provides the ability to utilise the arms within gait generation techniques to extend the region of stability. The software implementation of the 3D forward kinematics can be found in the MATLAB class `Kinematics.m` within the `MATLAB Gait Generation Code` linked in Appendix B.

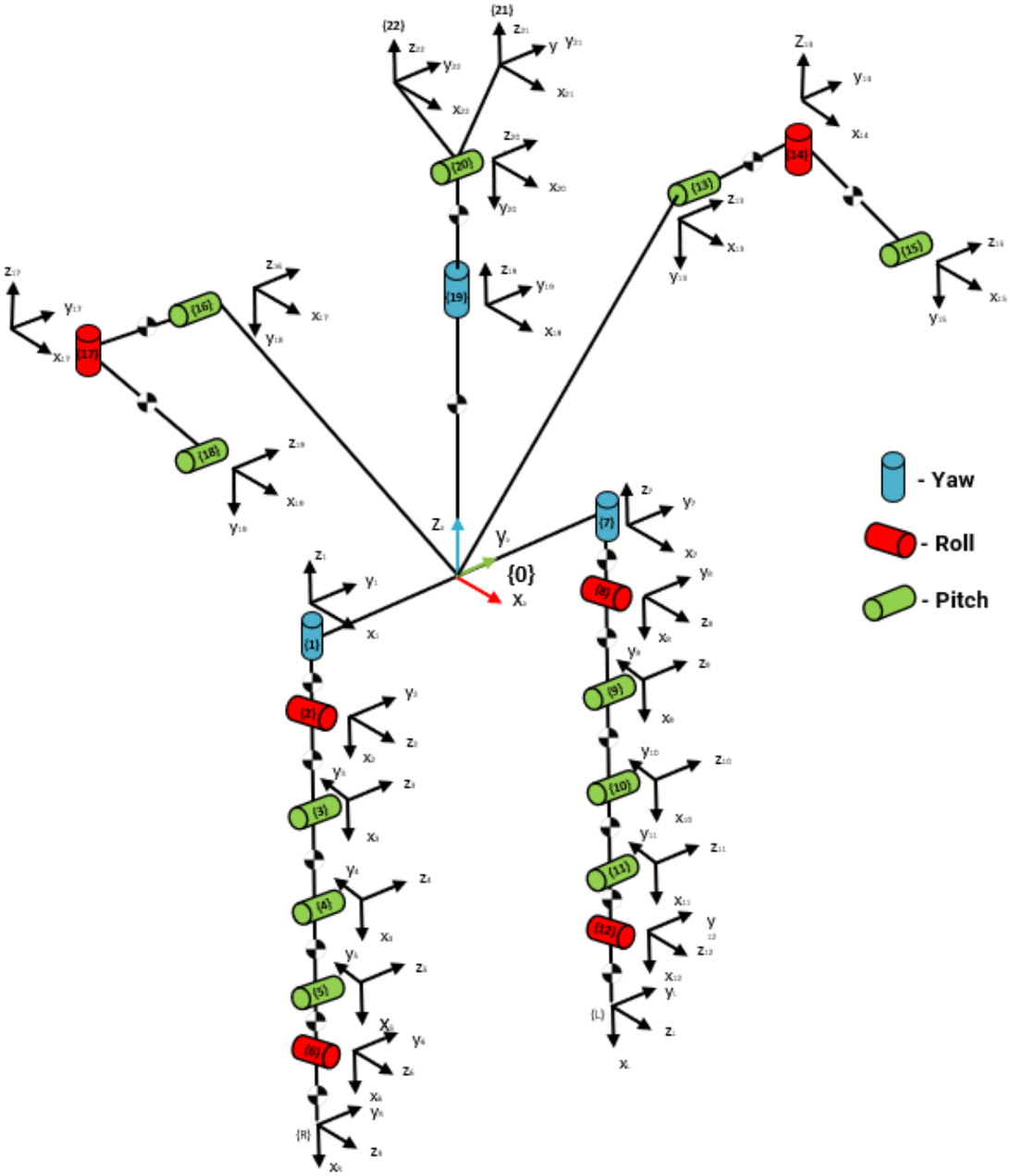

Figure 20: Complete 3D Kinematics.

The transform from the base {0} to the head pitch joint {20} can be calculated as follows:

$$H_2^0 0(q) = H_1^0 9 H^1 9_2 0 \tag{3.9}$$

The transform from the base {0} to the left lower arm pitch joint {15} can be calculated as follows:

$$H_1^0 5(q) = H_1^0 3 H^1 3_1 4 H^1 4_1 5 \tag{3.10}$$

The transform from the base {0} to the right lower arm pitch joint {18} can be calculated as follows:

$$H_1^0 8(q) = H_1^0 6 H^1 6_1 7 H^1 7_1 8 \tag{3.11}$$

Table 9 listed in Appendix G includes the position and orientation of all the joints and links required to replicate both 3D FKM's 19 and 20.

3.3. CAD System Identification

The NUgus robot is built with various 3D printed components using fused decomposition modelling. Using the individual 3D models of each component sourced from the NUbots team, an assembly of the robot was created in CAD software Fusion 360, as shown on the left in Figure 21. Fusion 360 provided an adequate amount of tools to develop a 3D assembly which could be used to obtain accurate system identification parameters such as link lengths, Centre of Masses (CoM) and moments of inertia. However, the 3D assembly was eventually migrated to the CAD software Onshape, shown on the right in Figure 21. Both a link to the Onshape model and an assembly drawing are located in Appendix C. The rationale behind migrating to the Onshape model was that it provided better cloud support for collaborating with team members and provides support for an API, Onshape-to-robot [25], that allows automatic generation of Unified Robotic Description Format (URDF) files. By specifying the joint positions of the Onshape assembly, a URDF was generated that maps all the inertial and kinematic properties into a format that can be used in robotic simulation environments. This greatly reduces the tediousness of calculating and manually rotating inertial parameters into the correct orientation for each individual joint frame. Furthermore, any modification of the physical hardware can be updated easily in the Onshape model and synced with the kinematic parameters by simply running a single script. The instructions for generating a URDF file of the NUgus platform are listed in Appendix B.

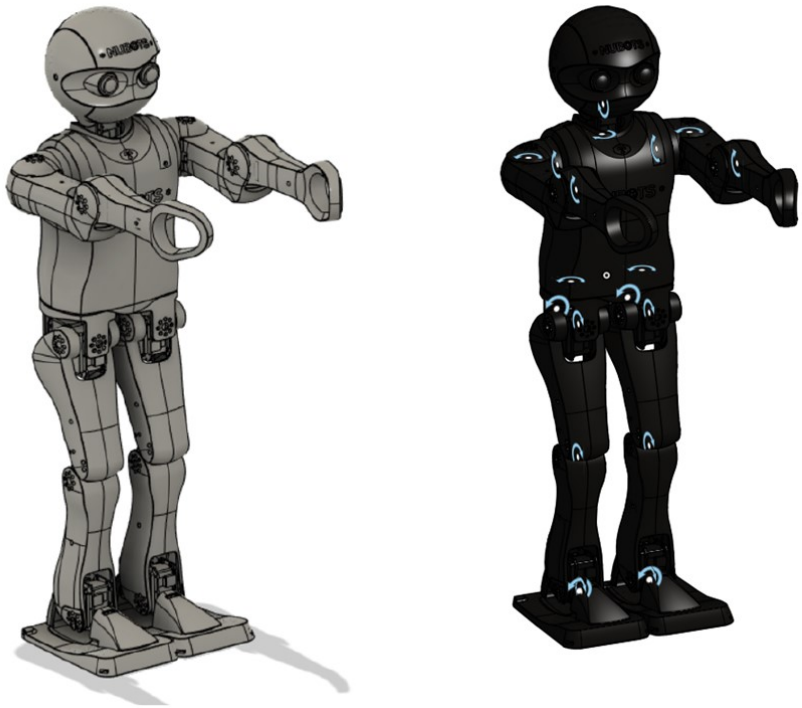

Figure 21: 3D Assembly Fusion360 vs Onshape.

The Unified Robot Description Format (URDF) is a commonly used XML based format for representing a robot model. The format was originally developed for the robotics software platform ROS, but is now integratable with a number of different software packages, such as MATLAB and Gazebo. It is best suited for robotic platforms represented as rigid links connected via joints, in a tree-like structure. The core components are the `joint` and `link` nodes shown in Listing 1 and 2 respectively. Within the links `inertial` node, the following parameters can be specified: Mass [kg], Centre of Mass [m] and the Moment of Inertia matrix. Additionally, the `inertial` and `collision` nodes can specify CAD files for visualisation and collision boxes for collision simulation. Within the joints `parent` and `child` node, the child and parent `link` is specified for the joint, to construct the tree-like structure. Additionally, the type of joint (revolute or linear), the pose, axis of rotation and physical characteristics can be incorporated in the `joint` node. A completed URDF file of the NUgus platform can be found in the Github repository linked in Appendix B.

```
<link name="torso">
<inertial>
<mass value="0"/>
<origin rpy="0 0 1.571" xyz="0 0 0.15"/>
<inertia ixx="0.0166572872071" ixy="0.00154518589069"
ixz="0.00493865774288" iyy="0.00422523541751"
iyz="0.00120275338963" izz="0.0555077839106"/>
</inertial>
<collision name="torso_collision">
<origin rpy="0 0 1.57079" xyz="0  0  0"/>
<geometry>
<mesh filename="mesh/torso.stl" scale="1 1 1"/>
</geometry>
</collision>
<visual name="torso_visual">
<origin rpy="0 0 1.57079" xyz="0  0  0"/>
<geometry>
<mesh filename="mesh/torso.stl" scale="1 1 1"/>
</geometry>
</visual>
</link>
```

Listing 1: Example URDF joint node.

```
<joint name="right_shoulder_pitch" type="revolute">
<parent link="torso"/>
<child link="right_shoulder"/>
<origin rpy="-1.57079  0       1.57079" xyz="0.005 -0.115  0.18"/>
<axis xyz="1  0  0"/>
<limit effort="7.3" lower="-3.14159265359" upper="3.14159265359" velocity="8.16814"/>
</joint>

```

Listing 2: Example URDF joint node.

The major benefit a URDF file of the NUgus platform provides is the ability for it to integrate into external robotics software packages. Both a 2D and 3D visualisation of the kinematic structures derived in Section was developed within MATLAB, shown in Figure 22 and 23, made possible through a combination of the URDF file and the `Robotics System Toolbox` MATLAB extension. This visualisation, shown in Figure 23, proved highly valuable in validating walking gaits generated, as it allowed the task space motions to be observed and any collisions between links to be identified.

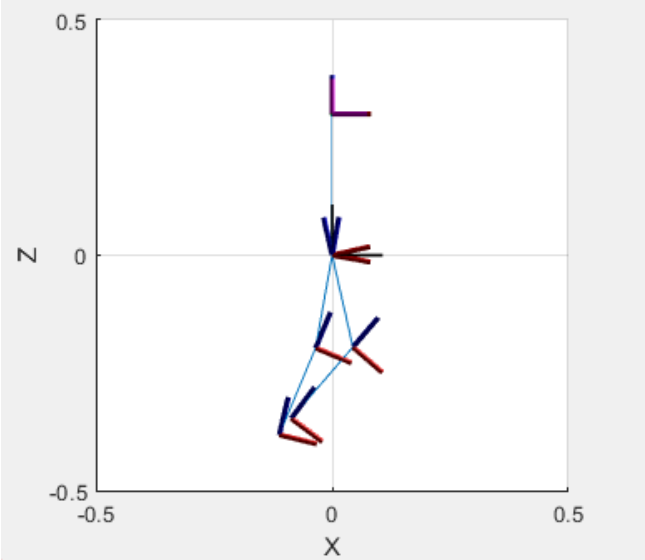

Figure 22: 2D MATLAB visualisation.

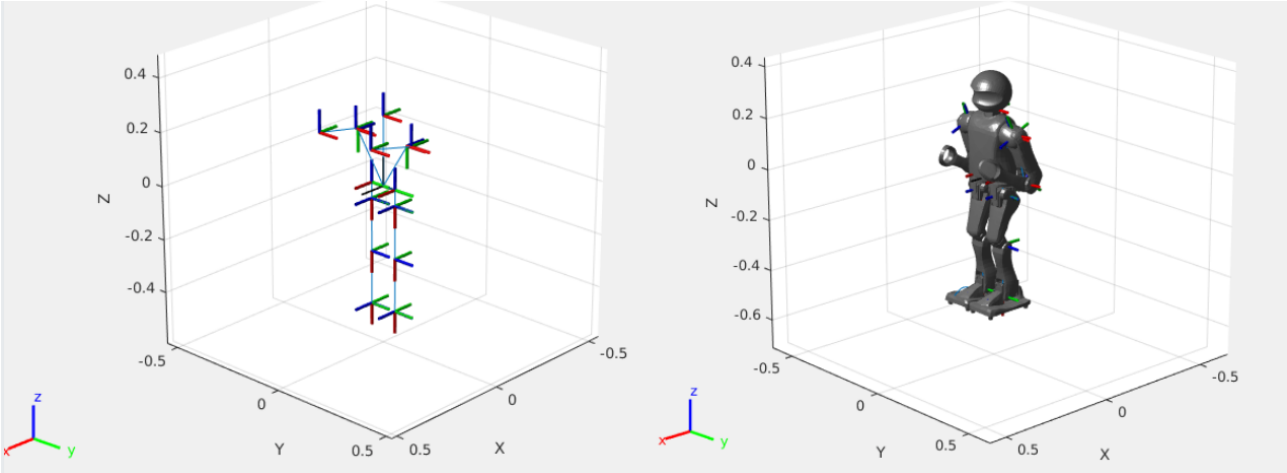

Figure 23: 3D MATLAB visualisation.

Listed in Table 2 are the centre of mass calculations retrieved from the CAD software estimations. The reference frame for the CoM measurements is the parent joint of the link.

Table 2: NUgus Link Mass and CoM Locations

Link Name	Mass [g]	CoM (x,y,z) [m]
torso	2954.9	(-0.0021828 -0.00894798 -0.0247563
left lower arm	287.0	(0.0604361 0.000764293 0.0233663)
left upper arm	302.6	(0.0542236 0.000478209 -0.0234837)
left shoulder	54.0	(0.00254925 -3.61369e-05 0.0287427)
left hip yaw (onyx)	48.0	(-0.0021828 -0.00894798 -0.0247563)
left hip yaw (metal)	177.3	(-0.0021828 -0.00894798 -0.0247563)
left hip roll	365.6	(0.015733 -0.000656594 0.0472427)
left upper leg (onyx)	411.5	(0.14746545, -0.00207665,-0.00198983)
left upper leg (metal)	342.3	(0.13051421, 0.00935887,0.00122207)
left lower leg(onyx)	171.9	(0.100171 0.00428532 0.00169225)
left lower leg(metal)	129.8	(0.1 0.00388549 2e-8)
left ankle block	365.6	-0.015733 0.0219762 -0.000469844)
left foot	202.3	(0.02481 0.00798543 0.0333477)
right lower arm	287.0	(0.0604361 0.000764293 0.0233663)
right upper arm	302.6	(0.0542236 0.000478209 -0.0234837)
right shoulder	54.0	(0.00254925 -3.61369e-05 0.0287427)
right hip yaw (onyx)	48.0	(-0.0021828 -0.00894798 -0.0247563)
right hip yaw (metal)	177.3	(-0.0021828 -0.00894798 -0.0247563)
right hip roll	365.6	(0.015733 -0.000656594 0.0472427)
right upper leg (onyx)	411.5	(0.14746545, -0.00207665,-0.00198983)
right upper leg (metal)	342.3	(0.13051421, 0.00935887,0.00122207)
right lower leg(onyx)	171.9	(0.100171 0.00428532 0.00169225)
right lower leg(metal)	129.8	(0.1 0.00388549 2e-8)
right ankle block	365.6	-0.015733 0.0219762 -0.000469844)
right foot	202.3	(0.02481 0.00798543 0.0333477)
neck	280.6	(2.08516e-08 0.000118161 -0.0273091)
head	473.5	(0.0665563 0.0103216 -0.000100617)

For bipedal stability, it is often required that a constraint is met involving the support polygon generated by the feet of the robot. In order to estimate the region of the support polygon, during a single support phase, dimensions of the NUgus's foot are required. To obtain these, a technical drawing of the NUgus' foot was created, which provided a maximum width and length of the foot. The reference frame $\{L\}$ in which the support polygon calculations are made lies horizontally in the centre of the foot, however, it does not sit directly in the centre of the foot length-wise, thus, the distance to $\{L\}$ is measured from both the back of the heel and front of the foot. Additionally, the region was reduced by 2cm in both length and width to help ensure any constraints involving the support polygon have some margin for error when implemented on the real platform. The measurements made are illustrated in Figure 24 and listed in Table 3.

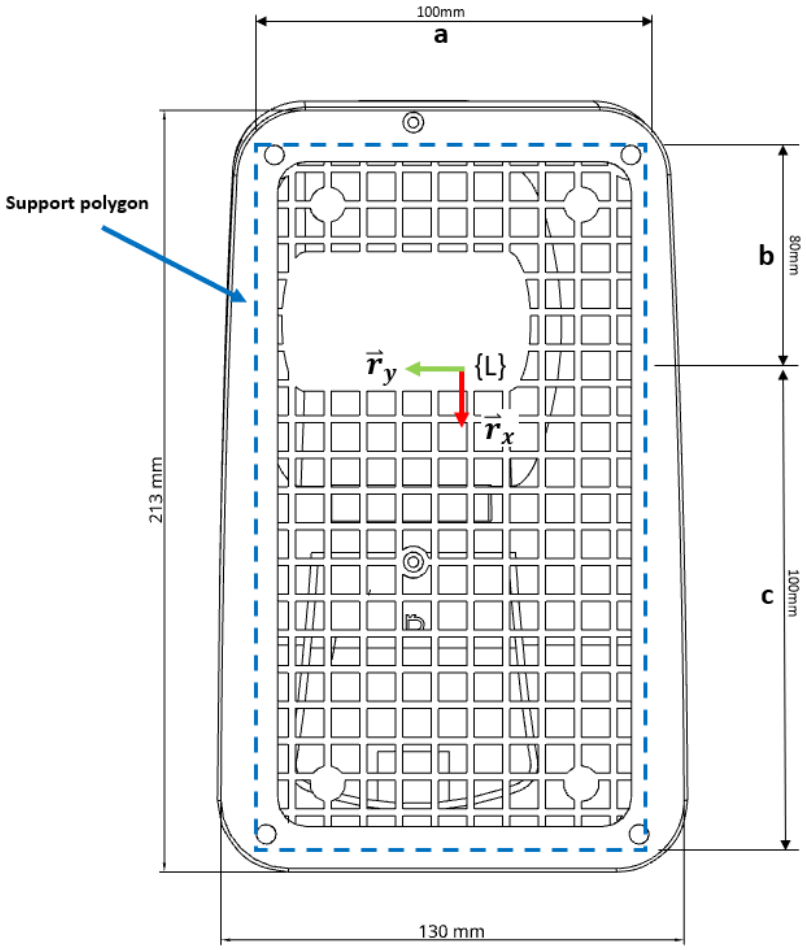

Figure 24: NUgus support foot dimensions.

Table 3: Support foot measurements

Measurement	Length [mm]
a	100
b	80
c	100

3.4. Experimental Mass Validation

An accurate approximation of the CoM of each of the links of the robots is required for various stability criterion explored in this report. A CAD based approach was implemented to estimate the CoM of each component, thus, accurate density values for each the individual components are required such that the mass is equivalent to the real component. To calculate the densities, the following equation was utilised:

$$\rho = \frac{M}{V} \tag{3.12}$$

where M is the mass of the component in [kg], ρ is the material density of the component, and V is the CAD estimated volume.

First, each individual component of the biped which consisted of a different material was measured using scales to obtain a mass value M. Some examples of measuring the components are shown in Figure 25. Next, the volume, V, of these components was sourced from the respective CAD file and used in Equation 3.12 to calculate the density value. A list of these values calculated are listed in Table 4. Finally, the components of the Onshape assembly model were updated with these values for use in CoM calculations.

The rationale behind this approach stemmed from poor estimates of Onyx 3D printed components mass values. Initially, a density of 1200 kg/m^3 was chosen based on values sourced from the Onyx data sheet, however, it was very apparent after comparing the CAD estimates that the mass of the real components with the data sheet density value was a poor approximation. By measuring the mass of various 3D printed Onyx parts and applying Equation 3.12 an approximation of the Onyx density parameter (764.1 kg/m^3) was derived.

Figure 25: Measuring Link Masses

Table 4: NUgus component densities

Component/Material	Density [kg/m^3]
Battery	2569.1
MX106 DYNAMIXEL Servo	1906.9
MX64 DYNAMIXEL Servo	2288.5
Onyx	764.1
Intel NUC	833.3
Aluminium legs	3917.8

3.5. Experimental Centre of Mass Validation

To validate the CoM of the NUgus links approximated by the Onshape CAD software, a method known as the 'Swing Test' can be utilised. Each link is approximated as a pendulum sub-system, shown in Figure 26, where L is the length from the pivot point to the CoM. A link, such as the NUgus upper leg is then attached to a pivoting joint. The link is dropped from an offset angle such that it swings, and using an encoder, the angular position and sample time are collected and stored. The experimental rig developed to collect this data is shown in Figure 27. Finally, using numerical optimisation, the data is fit to the model shown in Equation 3.13, using the cost function listed in Equation 3.14.

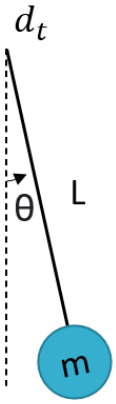

Figure 26: Pendulum

$$\begin{aligned} \dot{x}_1 &= \frac{-d_t x_1 - lsin(x_2) m_p g}{m_p l^2};, \\ \ddot{x}_2 &= x_1, \end{aligned} \tag{3.13}$$

where $x_1 = \dot{\theta}$, $x_2 = \theta$, d_t is the friction coefficient, l is length to CoM [m], g is gravity $[m/s^2]$ and m_p is the mass of the link [kg].

$$cost = ||(\theta - f(t, param))|| \tag{3.14}$$

where θ is the measured encoder angle and $f(t, param)$ is a function which given time and a parameter structure containing optimisation parameters l (CoM length), d_t, $theta0$ (initial θ), and $\dot{\theta}$ (initial $\dot{\theta}$) returns the models predicted angular displacement [rad].

To begin, a very simple and symmetrical object (lower metal leg) was swung in the swing test rig, shown in Figure 28. This symmetrical object was chosen as it was a safe assumption that the centre of mass should lie roughly in the centre of the part, illustrated by the *Estimated CoM* in Figure 28. Thus, acting as a good test object to ensure the swing test method produces sensible results.

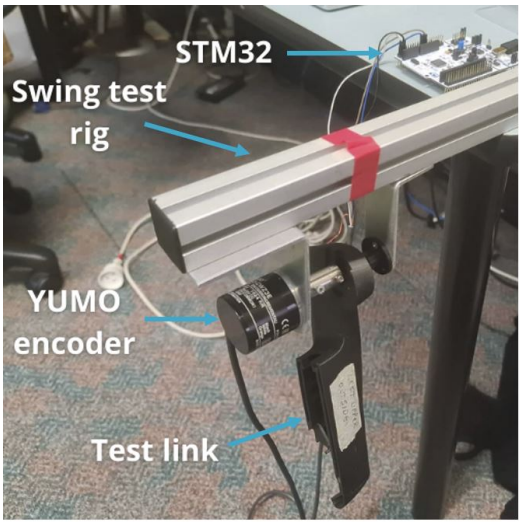

Figure 27: Swing Test Experimental Rig

Figure 28: Lower Leg (Metal)

The results of the swing test optimisation routine for the simple object are shown in Figure 29, which, as observed is a very good fit. However, the estimated length produced by the solver was 0.1436 [m], which is around 3-5cm off the hypothesis of where the CoM should lie. To further illustrate how poor this estimate is, Figure 30 shows where the swing test roughly estimates the CoM, which is evidently not a good estimate.

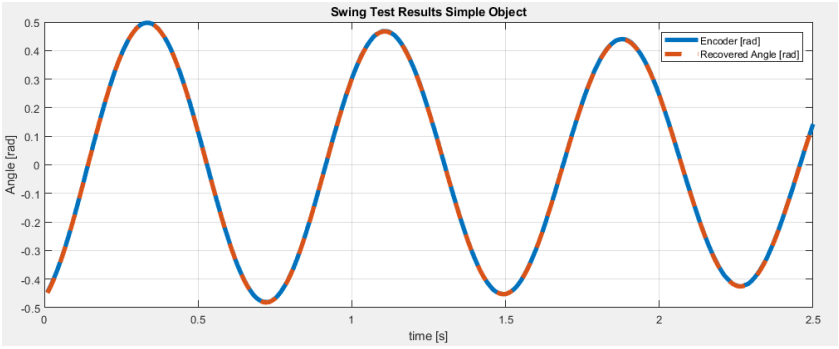

Figure 29: Lower Leg (Metal) Swing Test Result

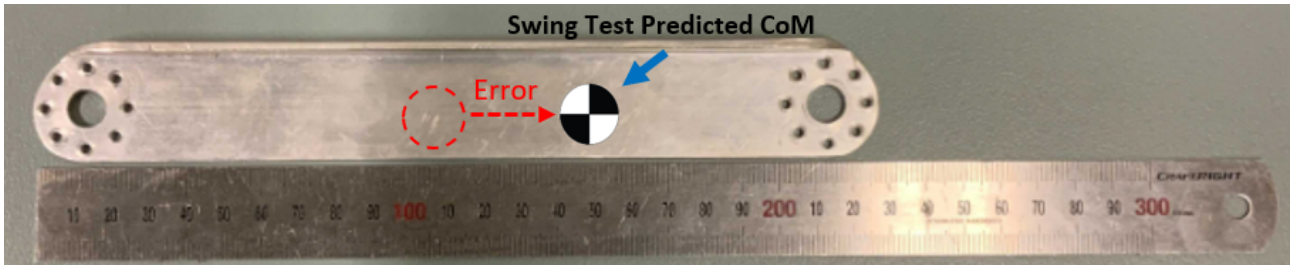

Figure 30: Poor Swing Test Result

This poor result could be attributed to the assumption that the link is a point mass with an inertia term of the following form, $I = m_p l^2$, where m_p is the mass of the link, and l is the length from the pivot to the point mass. To improve the results, a slightly more advanced swing test procedure was proposed. The updated method involved optimising over two data sets simultaneously, collected from both ends of the link. An assumption is then made that the Centre of Mass lies on the line between the two pivot points, thus, allowing a constraint to be included in the optimiser which ensures the two centre of mass lengths calculated add to the distance between the two points. A successful fit was achieved using this method, illustrated in Figure 31, for the simple component shown in Figure 28. A CoM of 0.1001 [m] and inertia of 0.00093622 $[kg/m^3]$ was estimated, a far more sensible result. This result could be further validated against the CAD estimated CoM and inertia which are 0.1 [m] and 0.000908 $[kg/m^3]$ respectively, resulting in a very small difference of 0.0001 [m] and 00002822 $[kg/m^3]$ for the CoM and inertia, a very promising result.

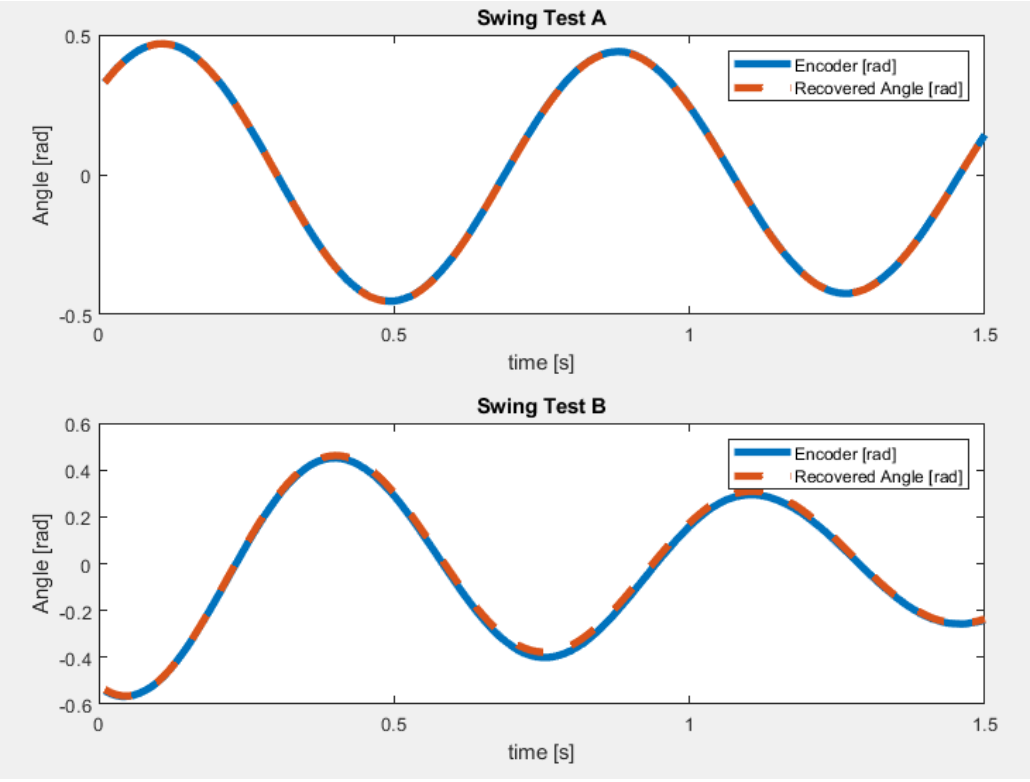

Figure 31: Improved Swing Test Result

Unfortunately, the success of this method did not transfer well to more complicated components. The poor estimates for these components could be attributed to the false assumption that the CoM lies on the line between the two pivot points, where in reality it is offset. For example, shown in Figure 32 is the CAD model of the upper leg, with the line between the pivot points drawn in red. As observed the CoM does not lie on the red line, thus, if this CAD estimated CoM is accurate, the swing test method would be expected to produce a poor result. In conclusion, it was decided that the CoM estimates from the Onshape CAD model were a good enough approximation as they could be validated for some parts. Furthermore, an honours thesis completed by Tanja Flemming, a member of a competing team in the Robocup competition revealed that the CAD approximations were a valid and accurate approach to CoM estimation [13].

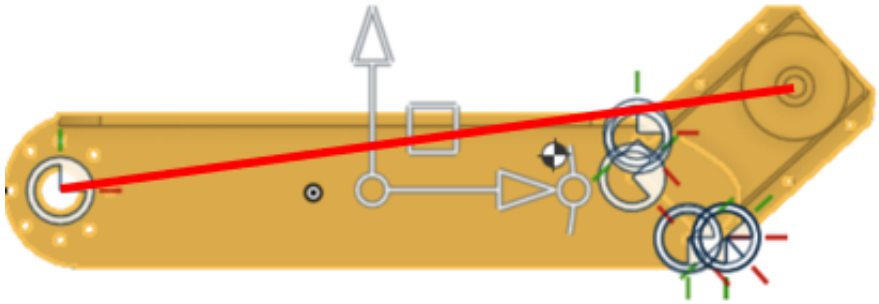

Figure 32: Line between upper leg pivot points

4. Gait generation fundamentals

The following chapter outlines the core components developed for generating walking gaits. The components include a `gait parameter structure`, `foot step planner`, `trajectory generation`, `support foot switch` logic and `inverse kinematics`. For each of these components a MATLAB function or class was created which acts as a module that integrates with the MATLAB program linked in Appendix B and the structure of the pipeline proposed is shown in Figure 33.

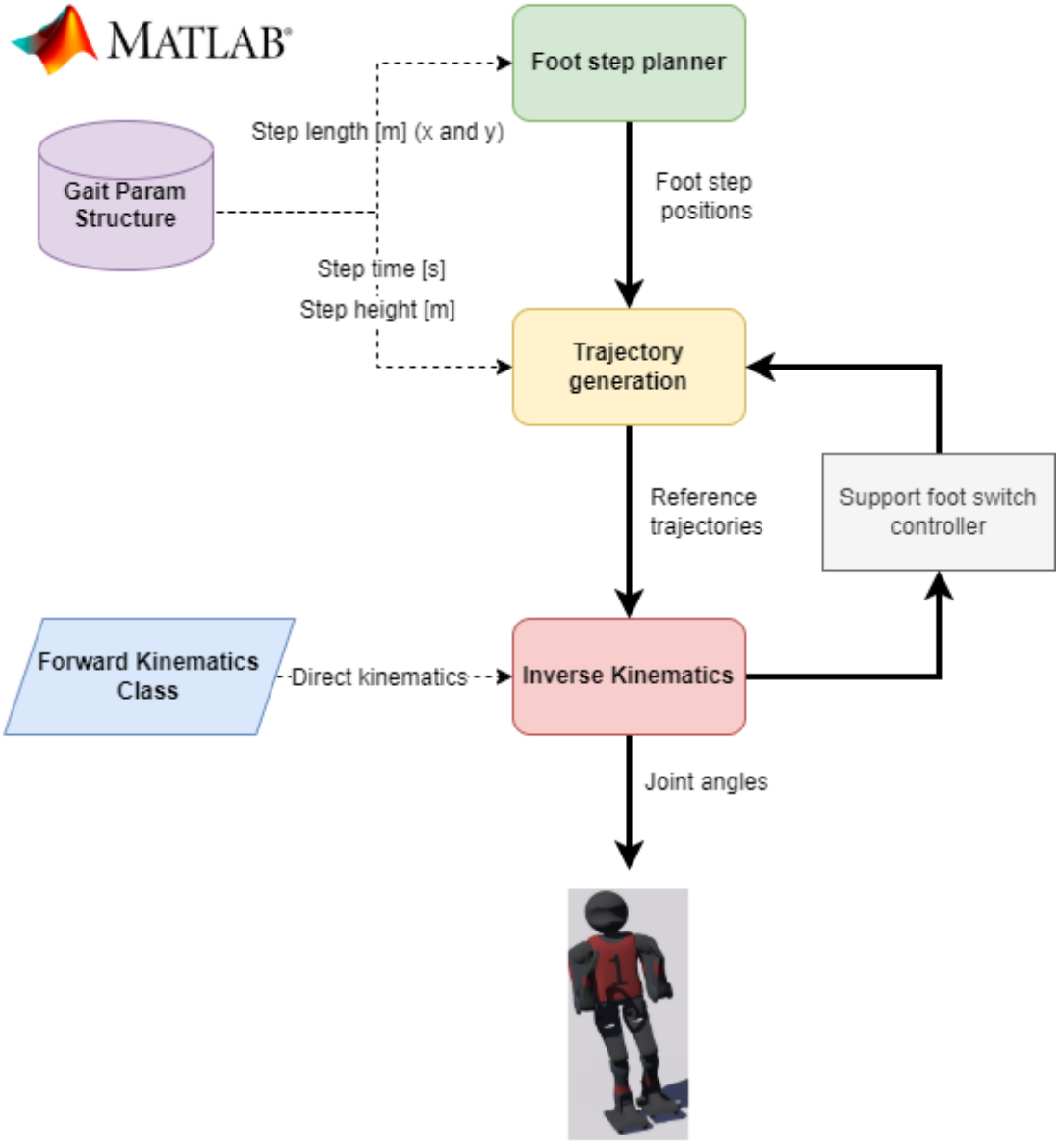

Figure 33: MATLAB software design

The following summarises the core modules of the gait generation method proposed and adds reference to the relevant script/class in the MATLAB code `MATLAB Gait Generation Code` linked in Appendix B. The subsequent sections of this chapter will go into further detail on how these modules function.

Gait parameter structure

The gait parameter structure stores the parameters listed in Table 5 which are used throughout each of the modules of the system.

Table 5: Gait generation parameter structure

Parameter	Description
step_time	Time for a single full step [s]
step_length_x	Distance in x for a single step [m]
step_length_y	Distance in y for a single step [m]
step_height	Max height of foot during step [m]
step_width	Horizontal distance between feet of robot in 0 state
Ts	Sampling time [s]
num_bodies	Number of links of the robot
N	Number of samples per step (step_time/Ts)
initial_conditions	Initial joint positions of the robot
num_footsteps	Number of footsteps

Code implementation: `gaitParameters.m`

Footstep planner

The footstep planner generates a feasible footstep plan for a bipedal robot to follow and passes the result to the trajectory generation module.

Code implementation: `generateFootsteps.m`

Trajectory generation

The trajectory generation module generates task space motion trajectory for the biped to follow such as a swing foot and centre of mass trajectories depending on the gait generation method.

Code implementation: `generateFootTrajectory.m` and `generateCoMTraj.m`

Inverse kinematics

Provided with a reference task space trajectory, the inverse kinematics module solves for the joint angles that result in the robot tracking these trajectories.

Code implementation: `inverseKinematics.m` and `inverseKinematicsZMP.m`

Forward kinematics class

Provided with a configuration of joint angles the forward kinematics class calculates the transforms to joints such as the swing foot and support foot of the NUgus platform. Additionally, it also includes a function for calculating the effective CoM of the whole robot.

Code implementation: `Kinematics.m`

4.1. Footstep planner

A footstep planner generates a feasible footstep plan for a bipedal robot to follow. The footstep planning approach implemented in this project involves a predefined number of finite steps, calculated based on specified step lengths in the x and y direction relative to the ground frame {g} shown in Figure 34. A $N \times 3$ matrix $\mathbf{F}$ is generated, where each row represents the position of the following step in the sequence. The footstep matrix is used throughout this report and is of the following form:

$$\mathbf{F} = \sum_{k=1}^{N} \begin{bmatrix} \frac{l_x k_1}{2} & -1^k \frac{w k_1}{2} + \frac{l_y k_1}{2} & 0 \\ \frac{l_x k_2}{2} & -1^k \frac{w k_2}{2} + \frac{l_y k_2}{2} & 0 \\ \dots & \dots & \dots \\ \frac{l_x k_N}{2} & -1^N \frac{w k_N}{2} + \frac{l_y k_N}{2} & 0 \end{bmatrix} \tag{4.1}$$

where w is the horizontal offset between the feet and l_x and l_y are step lengths in x and y.

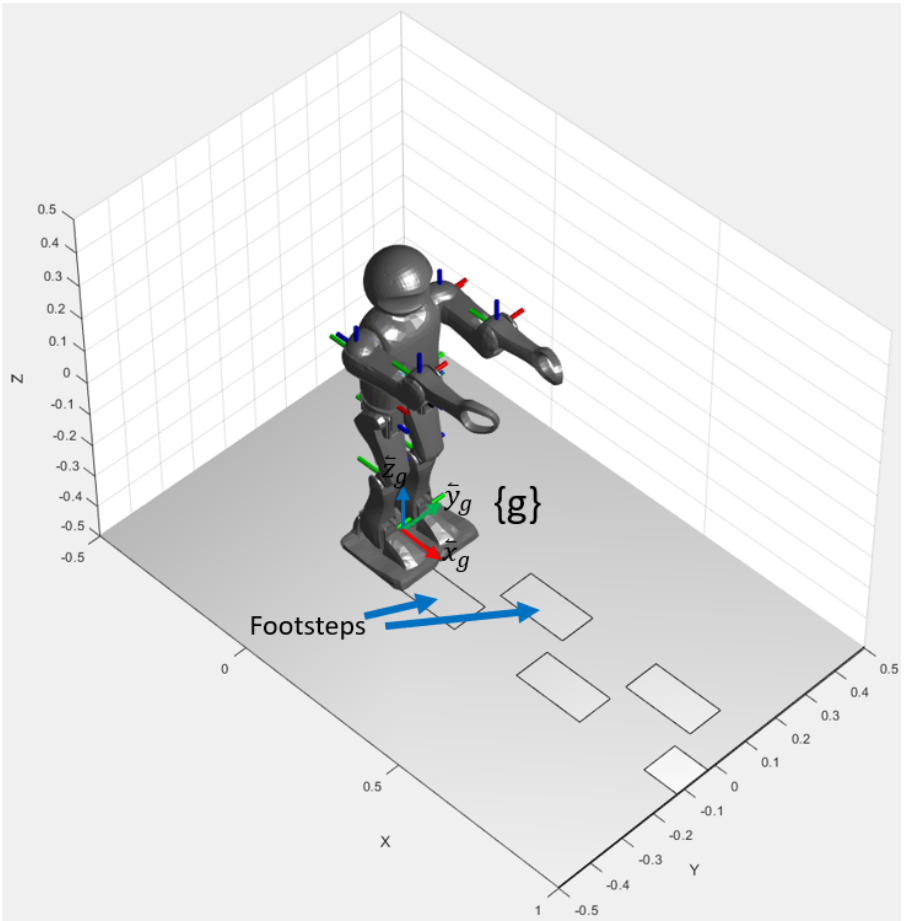

Figure 34: Footstep planner for step length 0.4 [m] in x

4.2. Trajectory generation

For gait generation, desired trajectories and motions for the biped to follow need to be generated, such as swing foot and CoM trajectories. A swing foot trajectory is a motion plan for the swing foot to follow between desired footstep positions. Computing this is achieved using polynomial splines, a curve fitting technique that computes a polynomial trajectory that smoothly interpolates between way-points comprised of a position and velocity. Each component (X,Y,Z) of the trajectory is decoupled, and splines are computed with way-points specified based on gait parameters `step_time`, `step_height` for Z, `step_time`, `step_length_x` for X and `step_time`, `step_length_y` for Y. Furthermore, an extension can be made such that the translation way points, `step_length_x` and `step_length_y` are chosen based on the footstep planner shown in Section 4.1. Provided with a foot planner matrix $\mathbf{F}$, we specify the way-point variables as follows:

$$\begin{aligned} \texttt{step\_length\_x} &= \boldsymbol{F(i+2,1) - F(i,1)} \\ \texttt{step\_length\_y} &= \boldsymbol{F(i+2,2) - F(i,2)} \\ \texttt{step\_height} &= \textit{constant} \end{aligned} \tag{4.2}$$

where i is the step index from $i = 1$ to N.

For example, the way-points and spline generated for a swing foot trajectories Z component is visually illustrated in Figure 35.

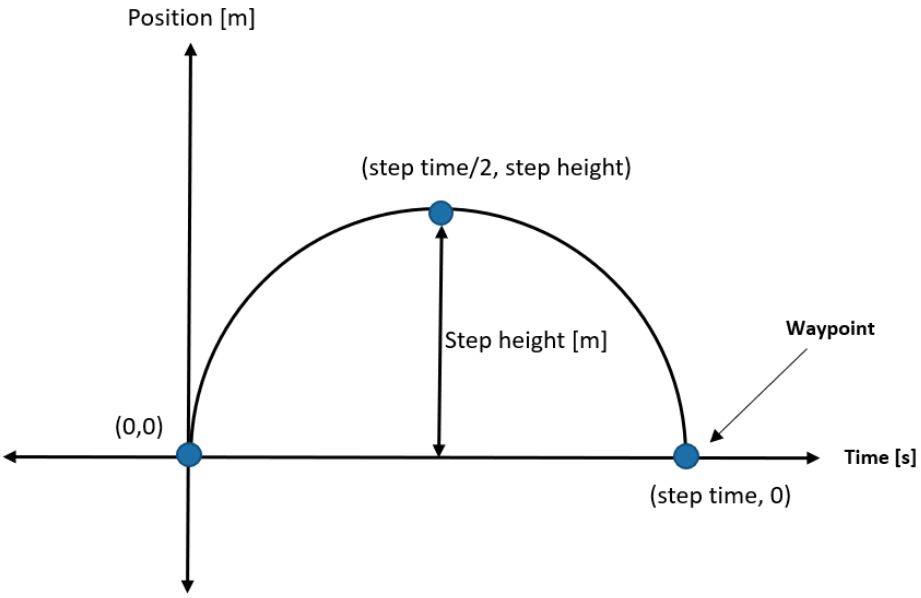

Figure 35: Spline Swing Foot Trajectory Z.

Mathematically, the Z swing foot trajectories can be calculated as follows:

$$
\begin{aligned}
\textbf{Waypoint 0: } Z_0 &= \begin{bmatrix} 0 & 0 \end{bmatrix} \\
t_0 &= 0 \\
tt_0 &= \begin{bmatrix} 1 \\ t_0 \\ t_0^2 \\ t_0^3 \end{bmatrix} \\
D &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \end{bmatrix} \\
T_0 &= [tt_0, Dtt_0] \\
\textbf{Waypoint 1: } Z_1 &= \begin{bmatrix} step\_height & 0 \end{bmatrix} \\
t_1 &= \frac{step\_time}{peak\_time} \\
tt_1 &= \begin{bmatrix} 1 \\ t_1 \\ t_1^2 \\ t_1^3 \end{bmatrix} \\
T_1 &= [tt_1, Dtt_1] \\
\textbf{Waypoint 2: } Z_2 &= \begin{bmatrix} 0 & 0 \end{bmatrix} \\
t_2 &= step\_time \\
tt_2 &= \begin{bmatrix} 1 \\ t_2 \\ t_2^2 \\ t_2^3 \end{bmatrix} \\
T_2 &= [tt_2, Dtt_2] \\
C1 &= \begin{bmatrix} Z_0 & Z_1 \end{bmatrix} \begin{bmatrix} T_0 & T_1 \end{bmatrix}^{-1} \\
C2 &= \begin{bmatrix} Z_0 & Z_1 \end{bmatrix} \begin{bmatrix} T_0 & T_1 \end{bmatrix}^{-1} \\
ttt_1 &= t_0 : Ts : t_1 \\
ttt_2 &= t_1 : Ts : t_2 \\
T1 &= \begin{bmatrix} 1 \\ ttt_1 \\ ttt_1^2 \\ ttt_1^3 \end{bmatrix} \\
T2 &= \begin{bmatrix} 1 \\ ttt_2 \\ ttt_2^2 \\ ttt_2^3 \end{bmatrix} \\
\textbf{Z Trajectory: } z &= \begin{bmatrix} C1T1 & C2T2 \end{bmatrix}
\end{aligned}
\tag{4.3}
$$

Note, the *peak_time* parameters specifies at what point during the step the max step height (apex) is reached. For example, if a *peak_time* of 2 is specified, the apex of the swing foot motion occurs half way between the step. Notably, it was found during simulation testing that a *peak_time* of 5 was optimal, as it resulted in the robot raising the foot earlier in the step, preventing clipping of the feet with the ground.

For the swing foot trajectories in both the X and Y directions, they are calculated as follows:

$$
\begin{aligned}
\textbf{Waypoint 0: } Z_0 &= \begin{bmatrix} 0 & 0 \end{bmatrix} \\
t_0 &= 0 \\
tt_0 &= \begin{bmatrix} 1 \\ t_0 \\ t_0^2 \\ t_0^3 \end{bmatrix} \\
T_0 &= [tt_0, Dtt_0] \\
\textbf{Waypoint 1: } Z_1 &= \begin{bmatrix} step\_length\_x/y & 0 \end{bmatrix} \\
t_1 &= step\_time \\
tt_1 &= \begin{bmatrix} 1 \\ t_1 \\ t_1^2 \\ t_1^3 \end{bmatrix} \\
T_1 &= [tt_1, Dtt_1] \\
ttt_1 &= t_0 : Ts : t_1 \\
T1 &= \begin{bmatrix} 1 \\ ttt_1 \\ ttt_1^2 \\ ttt_1^3 \end{bmatrix} \\
\textbf{X/Y Trajectory: } x/y &= \begin{bmatrix} C1T1 \end{bmatrix}
\end{aligned}
\tag{4.4}
$$

Considering the FKM shown in Figure 20 and a left support phase, each component is then mapped into the correct basis for the support foot frame L as follows:

$$
\boldsymbol{r}_{\boldsymbol{T/L}}^{\boldsymbol{L}} = \begin{bmatrix} -z \\ y \\ x \end{bmatrix} \tag{4.5}
$$

Finally, once the trajectory is computed, it must then be shifted to begin at current position of the swing foot $\{S\}$ relative to the support foot frame $\{P\}$. Let us consider generating a left support foot swing foot trajectory with respect to the snippet of the 3D forward kinematic model of the NUgus platform, shown in Figure 36. Provided with a trajectory $\boldsymbol{r^L_{T/L}}$, it must be shifted such that is starts at the base of the swing foot {R} using the following:

$$\boldsymbol{r^L_{R/L}}* = \boldsymbol{r^L_{T/L}} + r^L_{R/L}(0) \tag{4.6}$$

where $\boldsymbol{r^L_{T/L}}$ is the spline based swing foot trajectory, and $r^L_{R/L}(0)$ is the vector between the swing foot and support foot at the initial conditions of the step.

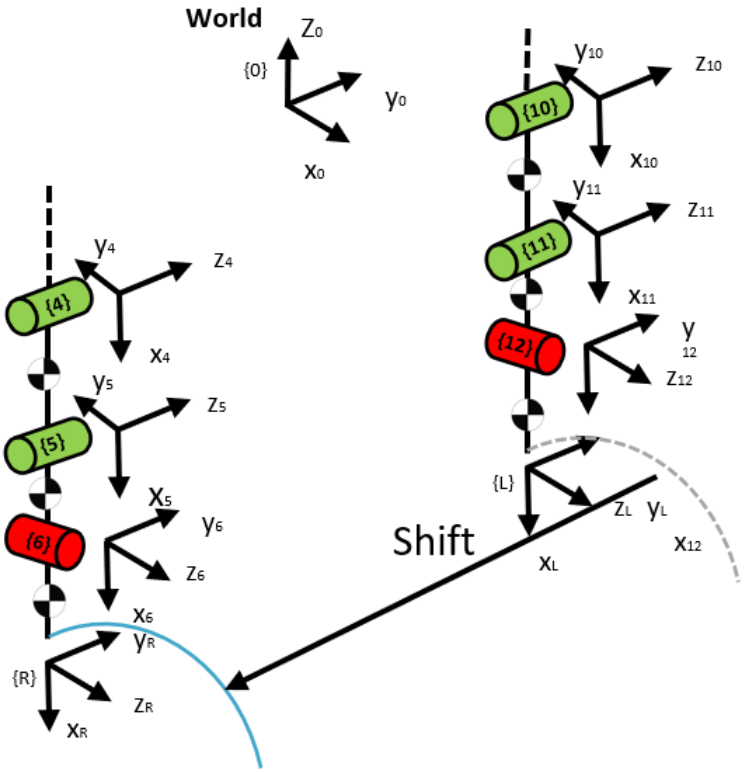

Figure 36: Kinematics Legs Snippet.

An example of a left support foot based swing foot trajectory generated within the MATLAB code developed is illustrated in Figure 37 and 38.

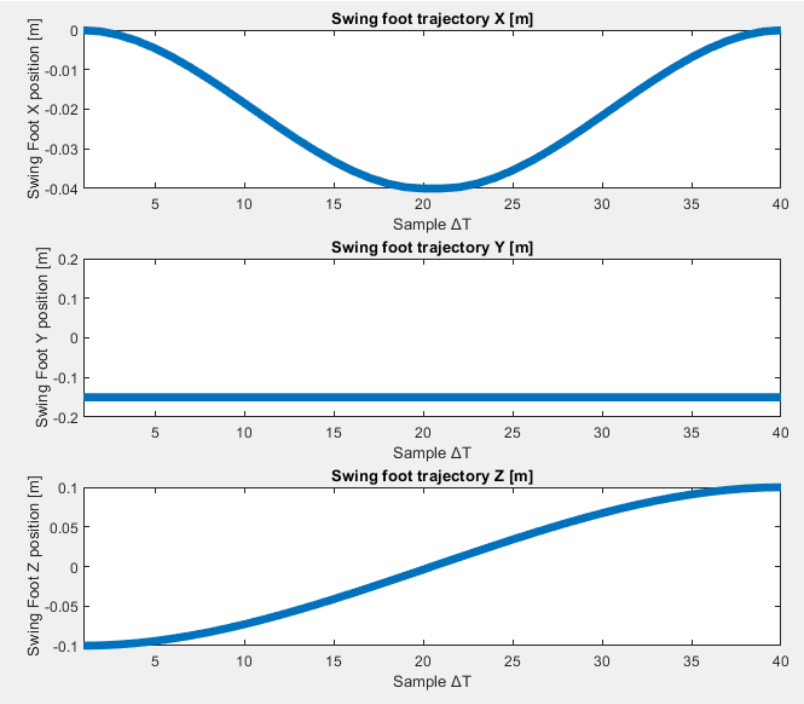

Figure 37: Spline Swing Foot Trajectory.

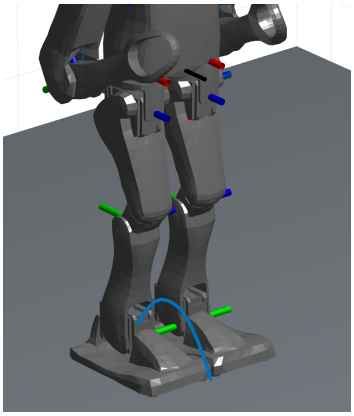

Figure 38: Trajectory Generation.

Together the footstep planner and swing foot trajectory generator can accept arbitrary walk commands in x or y directions, increasing the agility of the walking pattern generator, a crucial requirement in the Robocup competition as the soccer environment is rapidly changing. This enables the robot to walk forwards and backwards, diagonally and strafe, however, this method doesn't support rotational walk commands, rendering the method incapable of turning the robot around on the spot or following a curved desired path. Furthermore, the `trajectory generation` module also includes centre of mass trajectory generation, however, this is most critical for the zero-moment point method which is discussed further in Section 6.

4.3. Support foot switch

A major assumption made for stable walking in bipedal locomotion is that during a step, the support foot is fixed to the floor. For this reason, the support foot frame is chosen as the reference frame for tracking reference swing foot and CoM trajectories. However, during a walking cycle the support foot frame switches between feet after a step is taken, requiring calculations to be updated dynamically upon the switch. To achieve this switch within the algorithms developed, a support foot `flag` was utilised in the global parameter structure. This flag, when toggled, changes the current support foot frame used in calculations. To calculate the transform from the support foot to the swing foot we utilise Equation 4.7:

$$H_S^P(q) = (H_P^0(q))^{-1} H_S^0(q) \tag{4.7}$$

where P is the planted foot frame, and S is the swing foot frame and 0 is the torso frame.

Considering the 2D biped shown in Figure 39, initially in the left support phase, the transform from the support foot to the swing foot can be calculated as follows:

$$H_S^P(q) = H_R^L(q) = (H_2^0 H_L^2)^{-1} H_1^0 H_R^1 \tag{4.8}$$

After a support foot switch occurs to the right foot, the support-foot (P) to swing-foot (S) transform $H_S^P(q)$ must be dynamically updated to:

$$H_S^P(q) = H_L^R(q) = (H_1^0 H_R^1)^{-1} H_2^0 H_L^2 \tag{4.9}$$

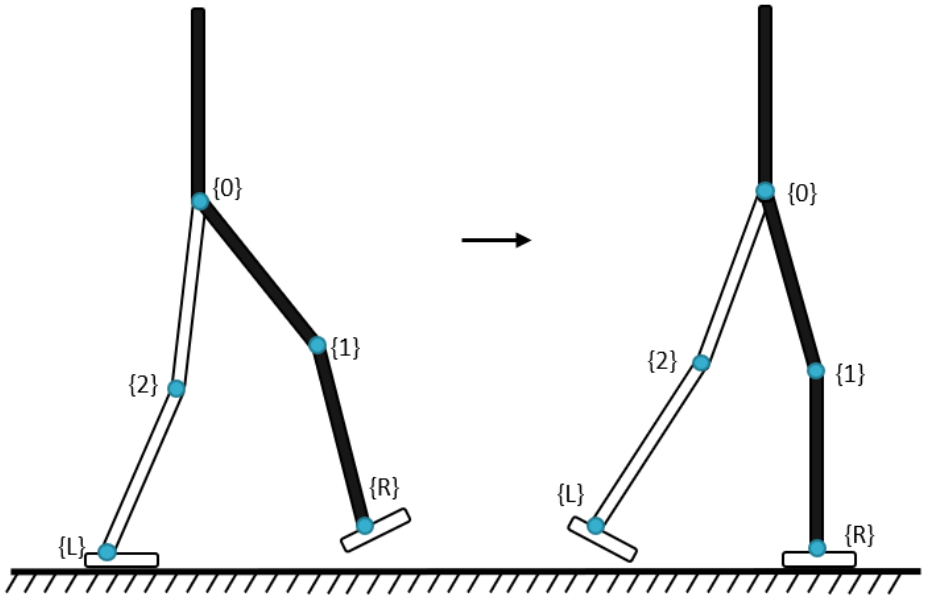

Figure 39: Support Foot Switch.

4.4. Inverse Kinematics

Inverse kinematics (IK) is the process of determining joint variables (q_n) given a desired end-effector pose (x_e^*). The position and orientation of a kinematic chain can be calculated directly with forward kinematics using a simple application of homogeneous transformations. However, the reverse operation is often much more challenging and the solution to the problem is of quite importance in order to transform desired operational space motions into corresponding joint space motions that can be executed on the real robots servos [29]. For simple manipulators an analytical solution to the inverse kinematics problem is often available, however, due to the complicated nature of the NUgus's kinematic structure, the approach taken to solve IK in this project was the use of numerical optimisation. To implement an optimisation based approach, a cost function of the following nature can be used:

$$q^* = \underset{q}{\operatorname{argmin}} \, ||q_k - q_0|| + ||r^p_{S/P}(q_k) - r^{p^*}_{S/P}(k)| \tag{4.10}$$

where q_k are joint angles at index k and q_0 is the previous solutions joint angles, $r^p_{S/P}(q)$ is the swing foot pose provided by forward kinematics, $r^{p^*}_{S/P}(k)$ is a desired swing foot pose at index k.

The most notable term in the cost function is the norm of the error between the swing foot pose and desired pose which is a function of joint angles. By minimising this error term, a set of joint angles that achieve the desired pose is found. Tracking of a swing foot trajectory is achieved point wise, where the desired pose $r^{p^*}_{S/P}(k)$ is a point k along the interpolated swing foot trajectory spline, implemented using the following algorithm:

Algorithm 1 Swing foot trajectory tracking

```
1: function INVERSEKINEMATICS
2:     for k ← 1 to N do
3:         q_0 = q_k                                   ▷ Warm start optimiser
4:         q_k = minimizeCost(r^{p*}_{S/P}(k), q_0)
5:         q(:, k) = q_k
6:         k = k + 1
7:     end for
8:     return q
9: end function
```

The returned variable q is a $[N\_joints \times N]$ matrix, where each column represents set of joint angles which results in the swing foot being positioned at the k-th point along the interpolated swing foot trajectory. Furthermore, to decrease the time taken for the optimiser to find a solution, a technique known as 'warm starting' is utilised. Since the joint angles that solve for the k-th point are not expected to greatly differ from the (k-1)th point, by setting the initial conditions of the optimiser to the (k-1)th solution, the time taken to find a solution is greatly reduced. Additionally, inclusion of the term $||q_k - q_0||$ adds a cost on how different the solution is from the previous. This term is crucial, as it prevents the solver from finding drastically varying joint angle solutions between points which prevents erratic motions.

MATLAB provides a numerical optimisation tool `fmincon` for nonlinear multi variable functions. This was chosen as the tool within MATLAB implementations of IK as it can ensure non-linear constraints are satisfied, which can be exploited to include stability criterion directly in the IK solver. Shown in Figure 40 are inverse kinematics solutions for various desired swing foot positions solved using `fmincon`.

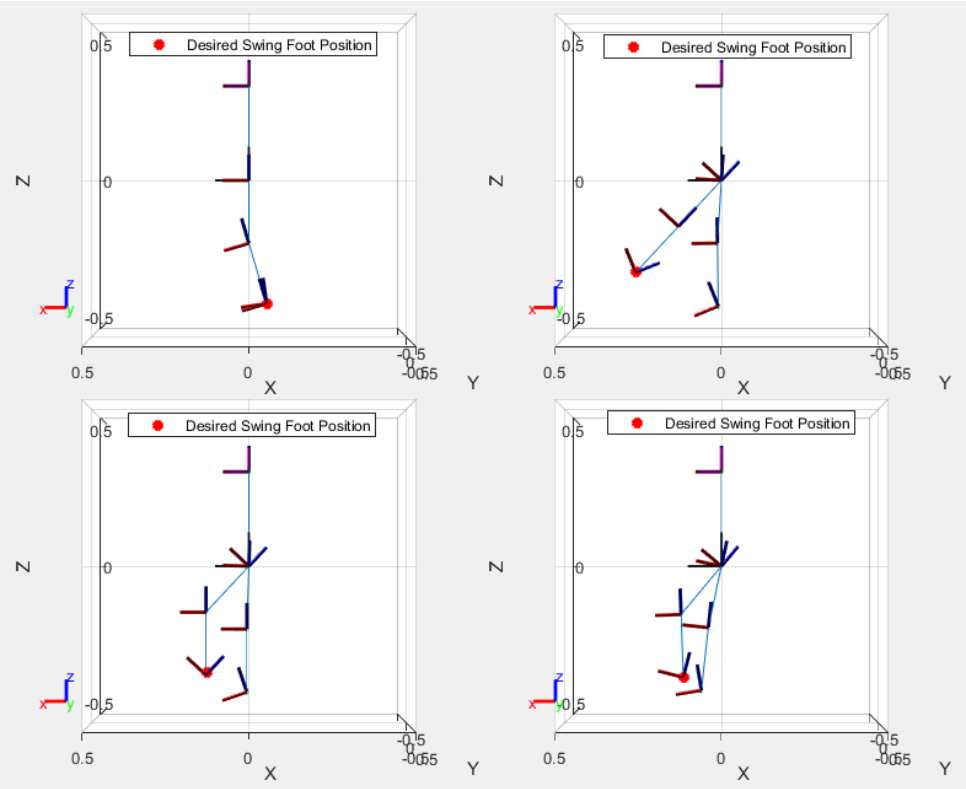

Figure 40: IK solving various swing foot positions.

The NUbots codebase which drives the NUgus platform is written entirely in C++, therefore, with the ultimate goal to transfer and implement the developed walking routines onto the real hardware platform, a C++ implementation of the inverse kinematics solver is required. The NUgus robot uses the `Intel NUC7i7bnh` single board computer which has performance limitations that need to taken into consideration, especially when dealing with a numerical iteration based method. For this reason, numerous forward and inverse kinematics solutions were developed in C++ and their performance was evaluated. The code for this is linked under `C++ Inverse Kinematics Code` found in Appendix B and the performance benchmark conducted for all these solutions is documented in Appendix F.

5. Quasi-static method

The following chapter outlines the quasi-static method for bipedal locomotion. The most important task of a walking mechanism is to preserve its stability, which is achieved by ensuring the support foot is in contact with the ground at all times. The foots only contact with the environment is realised via the friction force and vertical force of the ground reaction [34]. The most common method in which stability is achieved is based on a 'Static' or 'Quasi-static' approach. From the definition of walking, it is assumed that one foot is always in contact with the ground. The static method exploits this and ensures that the projection of the Centre of Mass (FCoM) on the ground is always kept inside the support polygon created by the feet. If the motion generated causes negligible inertia such that the system is 'static' during all stages of the gait, a stable walk can be achieved. Consequently, this assumption renders the method only capable of slow walking speeds and requires fine tuning to produce human-like motion.

To further illustrate the idea of static stability, let us consider Figure 41, which depicts a rectangular box on a ground plane, with its CoM at {C}. In order for the box to remain stable, its centre of mass projection must remain within the support polygon highlighted in green, as illustrated in the left figure. However, if the FCoM lies outside of the support polygon, the box will evidently tip over, depicted in the figure on the right.

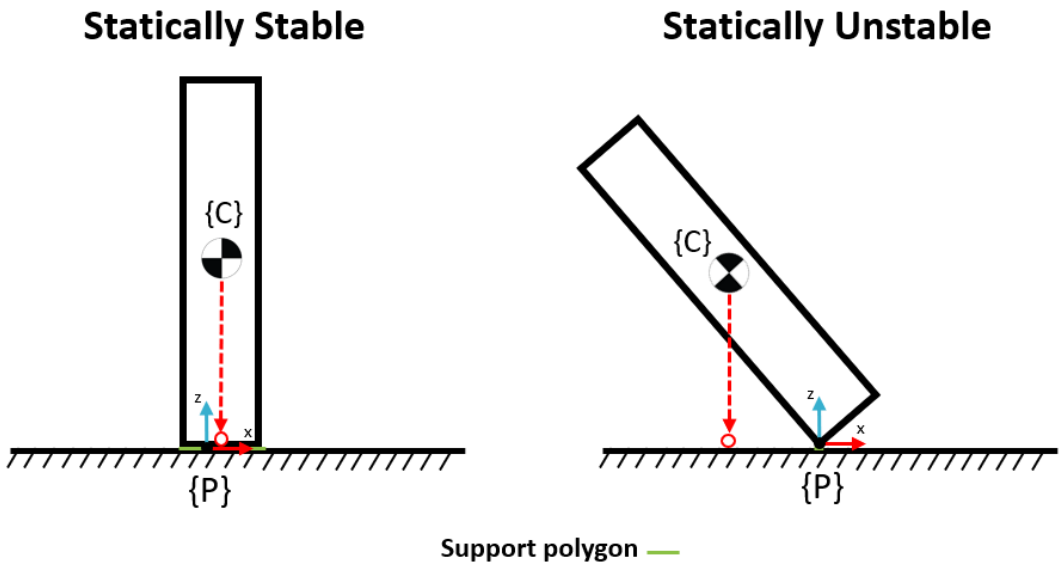

Figure 41: Static Stability.

During a quasi-static walking step, the FCoM is kept over the support foot, then, after the step is completed, shifted over to the next support foot polygon. A top down view of a typical quasi-static walking gait is visually depicted in Figure 42.

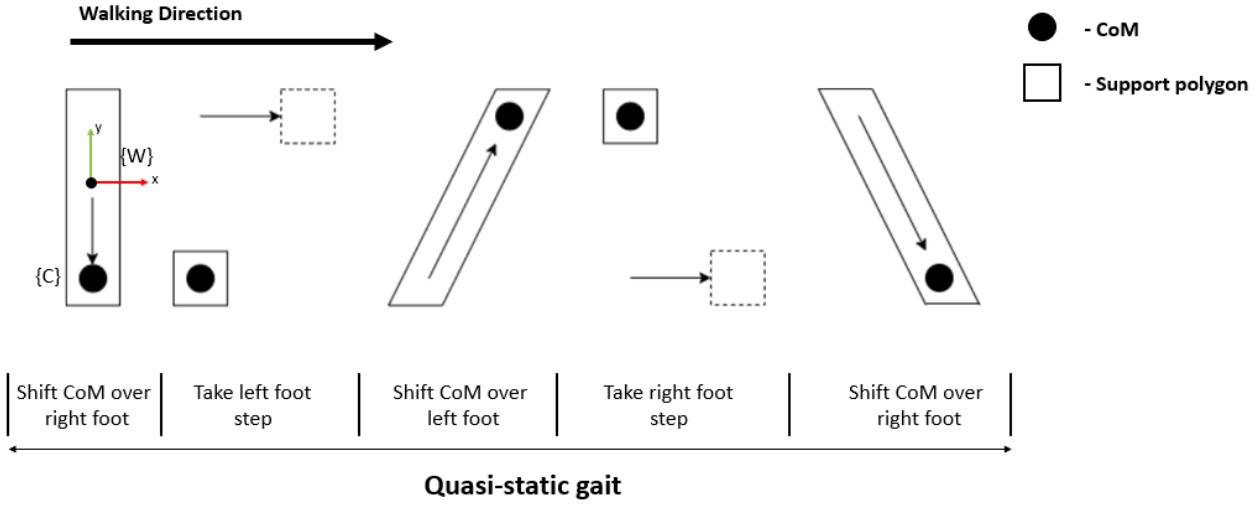

Figure 42: Quasi-static CoM gait.

5.1. 2D Quasi-static Implementation

A two-dimensional planar biped was chosen as the first platform to implement the quasi-static routine on, as the dynamics are considerably less complicated when compared to a three-dimensional model. The necessary conditions for quasi-static stability for a two-dimensional biped is visually illustrated in Figure 43. To ensure the quasi-static stability criterion is met during the walking cycle, we embed constraints in the numerical optimiser utilised to solve the inverse kinematics problem that tracks the swing foot trajectory $r^{p}_{S/P}*$ leading to the following cost function:

$$q^* = \underset{q}{\operatorname{argmin}} \, ||(q - q_0)|| + ||r^{p}_{S/P}(q) - r^{p^*}_{S/P}(k)|| \tag{5.1}$$

where $r^{p}_{S/P}(q)$ is the swing foot position relative to the support foot, $r^{p^*}_{S/P}(k)$ is the desired swing foot position at index k, q is a vector of joint angles and q_0 is a vector of the previous solutions joint angles.

Coupled with the following quasi-static constraints:

$$\begin{aligned} {e_1}^T r^{p}_{C/P} &<= \frac{l}{2}, \\ {e_1}^T r^{p}_{C/P} &>= -\frac{l}{2}, \end{aligned} \tag{5.2}$$

where $r^{p}_{C/P}$ is the the CoM of the 2D biped relative to the centre of the support foot, and l is the length on support foots support polygon generated.

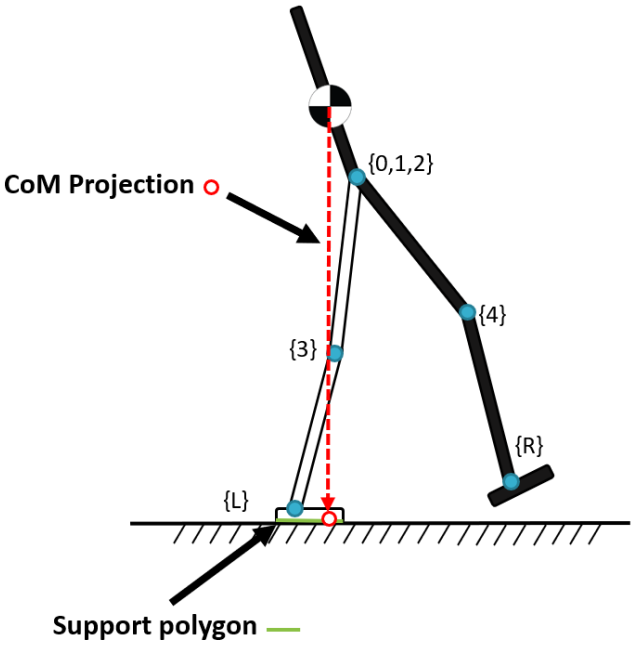

Figure 43: CoM and Support Polygon.

Considering a left support foot scenario, the swing foot position, $H_R^L(q)$, is calculated from the forward kinematic model as follows:

$$H_R^L(q) = (H_1^0 H_3^1 H_L^3)^{-1} H_2^0 H_4^2 H_R^4 = \begin{bmatrix} \mathbf{R}_R^L & \mathbf{r}_{R/L}^L \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.3}$$

Next, the swing foot position, $r_{S/P}^p(q)$ can be extracted from the translation component of the transform: $r_{S/P}^p(q) = \mathbf{r}_{R/L}^L$

The results of tracking a desired swing foot trajectory utilising the cost function shown in Equation 5.4 and Algorithm 1 are listed in Figure 44. As observed, the swing foot trajectory is successfully tracked with minimal tracking error and the CoM ($r_{C/P}^p$) resides within the support polygon for the whole duration of the step.

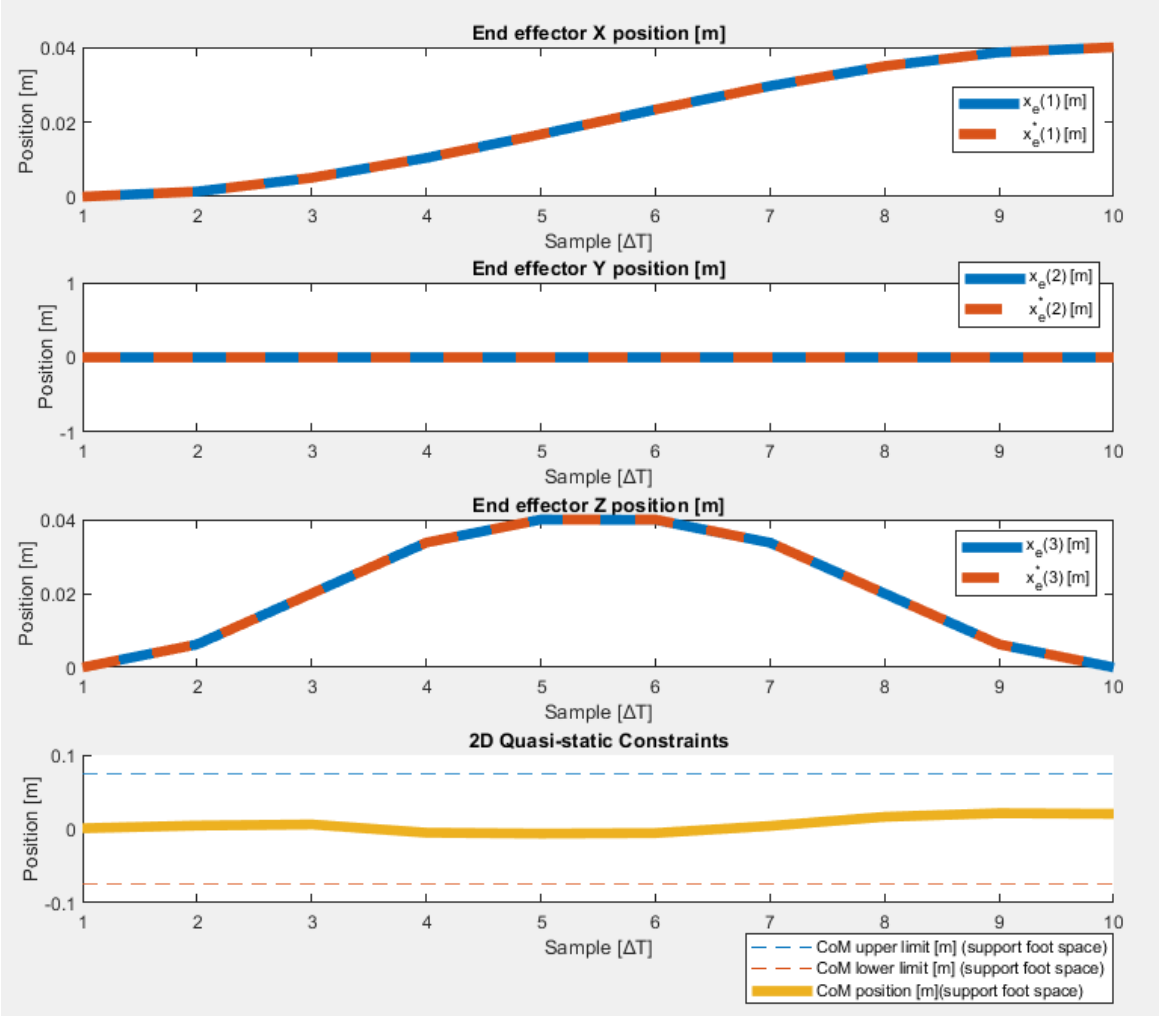

Figure 44: CoM and Support Polygon.

5.2. 3D Quasi-static Implementation

Once a suitable quasi-static walking gait was achieved in 2D, the control approach was then extended to the 3D kinematic chain depicted in Figure 19. A desired swing foot trajectory is provided to the inverse kinematics solver which utilizes numerical optimization to solve for the joint angles that result in the swing foot tracking the reference trajectory. The cost function of the inverse kinematics solver utilized is depicted in equation 5.4.

$$q^* = \underset{q}{\operatorname{argmin}} \, ||(q - q_0)|| + ||(r^p_{S/P}(q) - r^{p^*}_{S/P}(k))|| + \frac{1}{2}Tr(I - R_{S/P}) + \frac{1}{2}Tr(I - R_{6/0}) + \frac{1}{2}Tr(I - R_{12/0}) \tag{5.4}$$

where $r^p_{S/P}(q)$ is the swing-foot position, $r^{p^*}_{S/P}(k))$ is the desired swing-foot position at index k, q is a vector of joint angles, q_0 is a vector of the previous solutions joint angles.

The term $\frac{1}{2}Tr(I - R)$ is equivalent to $(1 - cos(\theta))$ where θ is the angle associated with the rotation between frames, which drives θ to 0 [21]. $R_{12/0}$ and $R_{6/0}$ are rotations between torso frame and ankle pitch and $R_{S/P}$ is the rotation between support and swing foot. These terms are included in the cost function to orientate the base of the swing foot such that it is parallel with the floor when taking a step.

Additionally, the following non-linear constraints, 5.5 and 5.6, are coupled with the cost function to ensure the quasi-static stability criterion are met:

$$\begin{aligned} e_3^\intercal r^p_{C/P} &\leqslant \frac{1}{2}c, \\ e_3^\intercal r^p_{C/P} &\geqslant -\frac{1}{2}b, \\ e_2^\intercal r^p_{C/P} &\leqslant \frac{1}{2}a, \\ e_2^\intercal r^p_{C/P} &\geqslant -\frac{1}{2}a, \end{aligned} \tag{5.5}$$

where $r^p_{C/P}$ is CoM relative to the centre of the support foot, a, b and c are the bounds of the support polygon of the support foot listed in Table 3.

Furthermore, to ensure the knees do not bend backwards during the walk, the following constraints were placed on the knee joints:

$$\begin{aligned} q(15) &\leqslant \pi/2, \\ q(15) &\geqslant 0, \\ q(4) &\leqslant \pi/2, \\ q(4) &\geqslant 0, \end{aligned} \tag{5.6}$$

where q is a vector of joint angles.

The results of the IK optimisation routine implemented in the MATLAB function `fmincon` are listed in Figure 45. These results are for a single step with the following gait parameters: `step length` = 0.075 [m], `step height` = 0.02 [m] and `step time` = 1 [s]. As observed, the optimiser is able to find solutions that track the desired trajectory with very good accuracy.

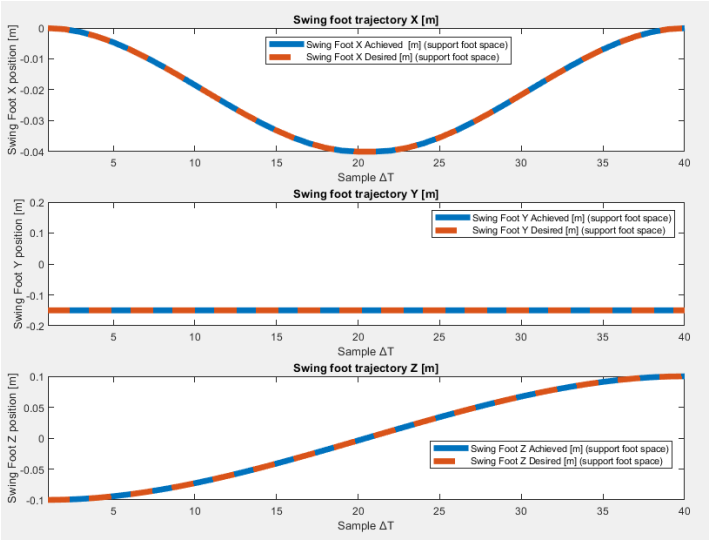

Figure 45: 3D Inverse kinematics results for single step.

Furthermore, the optimiser is able to ensure the quasi-static stability criterion are successfully satisfied during the whole step, as shown in Figure 46.

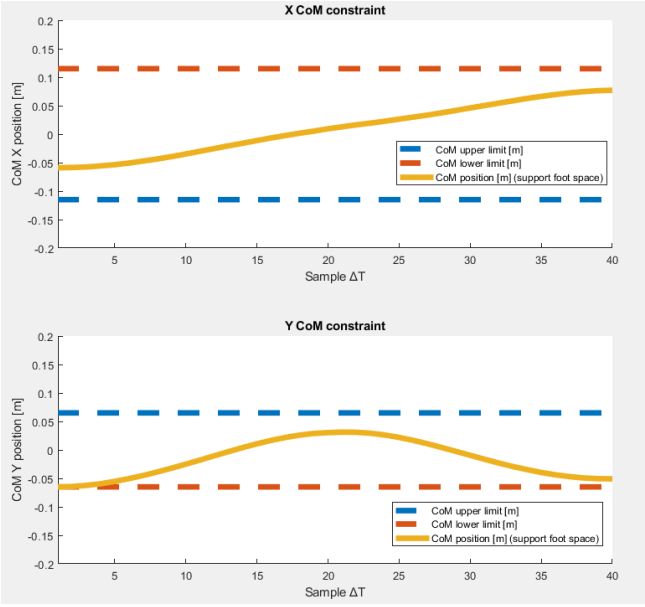

Figure 46: 3D CoM Constraints.

6. Zero moment point method

In recent years, the zero moment point (ZMP) has become quite an important concept in the field of bipedal stability, commonly used to generate dynamic walking routines. The major drawback of the quasi-static method is that only slow walking speeds can be achieved due to the underlying assumption that the biped is statically stable during all stages of the gait cycle. A zero moment point (ZMP) approach can be used to generate dynamic walking gaits that can potentially achieve faster walking speeds, as it allows for the FCoM to momentarily leave the support polygon. The control architecture of ZMP takes the central idea of the quasi-static approach, but applies it to the ZMP, ensuring it is always inside the support polygon. The ZMP is defined as "the contact point for which the moments due to gravity exactly balance out the moments due to contact with the ground" [7]. In order to derive and control the position of the ZMP, the complex non-linear model of the robot can be approximated with a simple Linear Inverted Pendulum Model (LIPM) positioned at the support foot, as illustrated in Figure 47 [7, 16].

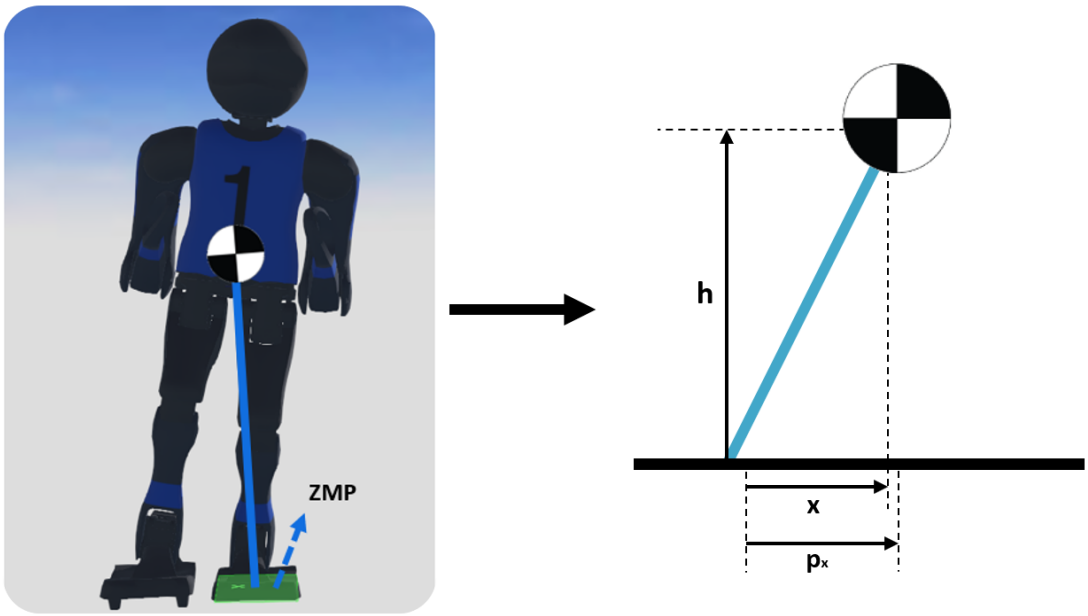

Figure 47: Simple Linear Inverted Pendulum.

6.1. Zero moment point (ZMP)

The ZMP, was first proposed by Miomir Vukobratovic in 1968 and is a point on the ground plane of a walking robot, introduced under the assumption that the floor is flat and the friction forces are strong enough to compensate the ground reaction forces tangential to the ground. Since the robot can only push on the ground, and cannot pull, the only forces that can compensate forces that tend to overturn the robot are the ground reaction forces. The ZMP is the point where the ground reaction force must be applied to compensate for other forces and must exist within the support area to ensure stability [28].

To illustrate this further, let us consider a biped in the single support phase, represented as a simple linear inverted pendulum, shown in Figure 48.

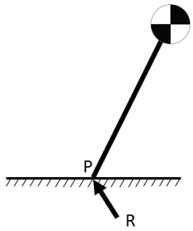

Figure 48: Simplified biped [34].

The dynamics can be simplified for the sake of analysis, such that the upper components of the biped can be neglected and replaced by a force F_A and moment M_A, depicted in Figure 49, where the weight of the foot acts at its centre of gravity, point G, and the foot also experiences a reaction force P. This point P is vital as it keeps the whole mechanism in equilibrium, with the force terms R (Rx, Ry, Rz) and moment terms M (Mx, My and Mz) representing the components of the ground reaction. We assume friction acts at the centre of the point of contact between the foot and ground, thus, since the foot is assumed to be at rest, the R and M components are balanced by friction in the horizontal plane. Therefore, the forces R_x and R_y balance out the horizontal force imposed by F_a and the reaction moment Mz balances the vertical component of the moment M_A and the moment induced by the force F_A [34].

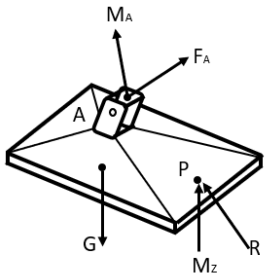

Figure 49: All reaction forces and moments [34].

Ultimately, if we assume there is no sliding between the feet and ground, the static friction will compensate for the vertical reaction torque (Mz) and the horizontal force components (Rx, Ry). Additionally, the vertical reaction force R_z represents the ground reaction that balances all the vertical forces. Finally, we must then consider the balancing of the horizontal component of the load moment. Since the ground reaction force induced by foot action is always oriented upwards, the ground horizontal components of all active moments can be compensated for simply by changing position of the reaction force R within the support polygon. Therefore, the horizontal component of the moment M_A will shift the reaction force to the corresponding position, to balance the additional load. To visually illustrate this, we can observe the simple planar case in the y-z plane, shown in Figure 50 [34].

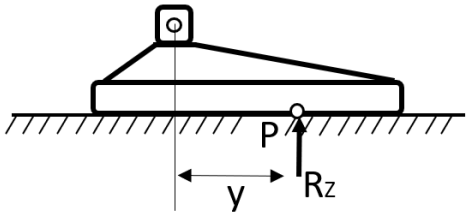

Figure 50: Simple planar case [34].

To balance the moment M_{Ax} the acting point of the force Rz is shifted, whose value is determined by the equation of balance of all the forces acting on the foot, by the corresponding distance y. It is important to note that the whole time the reaction force is within the support polygon, any increase in the ankle moment will be compensated for by simply changing the position of this force, and no horizontal components of the moments M_x and M_y will be present.However, if the support polygon is not large enough to encompass the necessary position of the force R to balance the action of external moments, the biped will undergo rotation about the foots edge, which will result in the biped falling. Therefore, we can say that the required condition for a biped to be in dynamic equilibrium is that the point P on the sole where the ground reaction force is acting exhibits horizontal moments of the following form [34]:

$$M_x = 0, M_y = 0 \tag{6.1}$$

In conclusion, since both components are equal to zero, the term "zero moment point" is chosen [34].

6.2. Linear Inverted Pendulum Model (LIPM)

As discussed, the complex non-linear model of the robot can be approximated with a simple constrained Linear Inverted Pendulum Model (LIPM) positioned at the support foot, illustrated in Figure 51. The LIPM is constrained such that the mass can only move along an arbitrary plane. Additionally, three assumptions must be made [7, 6, 16] :

- Assumption 1: the robot has enough joint torque to achieve the desired motion.
- Assumption 2: there is no angular momentum around the CoM.
- Assumption 3: the CoM keeps a constant height.

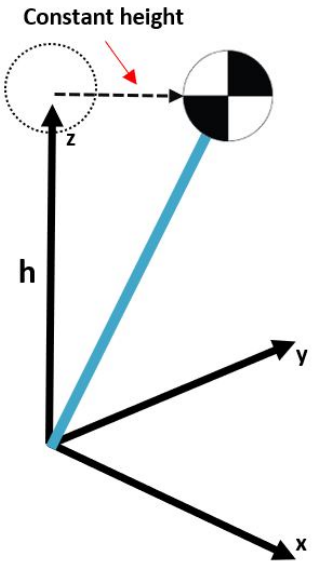

Figure 51: LIPM under constraint [16].

Under these assumptions the dynamics of the robot are reduced to 6.2 [16]:

$$\begin{aligned} \ddot{x} &= \frac{g}{h}(x - p_x), \\ \ddot{y} &= \frac{g}{h}(y - p_y), \end{aligned} \tag{6.2}$$

where g is gravity, h is the constant height of CoM, x and y are the horizontal distance to CoM and p_x and p_y are the zero-moment points. By defining a new variable $\frac{d}{dt}\ddot{x} = u_x$, the time derivative of the horizontal acceleration of the CoM, the dynamics can be converted into a strictly proper dynamical system:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_x$$

Where the output of the system is the ZMP as follows:

$$p_x = \begin{bmatrix} 1 & 0 & -z_c/g \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$$

To implement the LIPM dynamics on a computer based control system, it must first be discretised [16] with a sampling time of Ts, leading to the dynamics listed in Equation 6.2.

$$\begin{aligned} X(k+1) &= AX(k) + Bu(k), \\ p(k) &= CX(k), \end{aligned}$$

where

$$\begin{aligned} X(k) &= \begin{bmatrix} x(kT) & \dot{x}(kT) & \ddot{x}(kT) \end{bmatrix}^T \\ u(k) &= u_x(kT), \\ p(k) &= p_x(kT), \\ A &= \begin{bmatrix} 1 & T & T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}, \\ B &= \begin{bmatrix} T^3/6 & T^2/2 & T \end{bmatrix}^T, \\ c &= \begin{bmatrix} 1 & 0 & -z_c/g \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}. \end{aligned} \tag{6.3}$$

To increase the robustness of the controller, we can augment the states of system 6.3 via a process known as integral action [20]:

$$\begin{aligned} \Delta u(k) &= u(k) = u(k-1) \\ \Delta X(k) &= X(k) - X(k-1) \end{aligned} \tag{6.4}$$

with augmented state:

$$\Delta \tilde{X}(k) = \begin{bmatrix} p(k) \\ \Delta X(k) \end{bmatrix} \tag{6.5}$$

Leading to the following re-expressed state dynamics:

$$\begin{aligned} \tilde{X}(k+1) &= \tilde{A}\tilde{X}(k) + \tilde{B}\Delta u(k) \\ p(k) &= \tilde{C}\tilde{X}(k) \end{aligned} \tag{6.6}$$

where,

$$\tilde{A} = \begin{bmatrix} I & CA \\ 0 & A \end{bmatrix}, \tilde{B} = \begin{bmatrix} CB \\ B \end{bmatrix}, \tilde{C} = \begin{bmatrix} I & 0 & 0 & 0 \end{bmatrix}, \tag{6.7}$$

6.3. MPC Walking Pattern Generation

Model predictive control (MPC) is an advanced method of control that can be used to control a system while ensuring a set of constraints are satisfied. It is one of the few control methods outside of PID control that has been implemented extensively in industry. A model predictive controller requires a dynamic model of the system, most commonly, a linear model obtained through system identification. The key advantage of MPC is that it allows the control action to be optimised, by taking into account future system dynamics. This is achieved by optimising over a finite time-horizon, but only implementing the current times control action and then optimising again, repeatedly.

By specifying a ZMP trajectory that resides within the support polygon during the whole walking gait, stability can be achieved by tracking the reference trajectory via MPC. Provided with the footstep plan matrix **F** from the footstep planner discussed in Section 4.1, the following interpolation algorithm can be utilised to interpolate a ZMP trajectory between the support polygons:

Algorithm 2 ZMP reference trajectory interpolation

```
1:  function GENERATEZMPTRAJECTORY
2:      for i ← 1 to N × step_time do
3:          zmp_star(1,i) = F(k,1)
4:          zmp_star(2,i) = F(k,2)
5:          if i ≠ 0 and (i%t_step) == 0 then
6:              k = k+1
7:          end if
8:          i = i + Ts
9:      end for
10:     return zmp_star
11: end function
```

An example of the output of the ZMP reference generated by this algorithm is depicted in Figure 52 for gait parameters: `step length x` = 0.125 and `step length y` = 0.

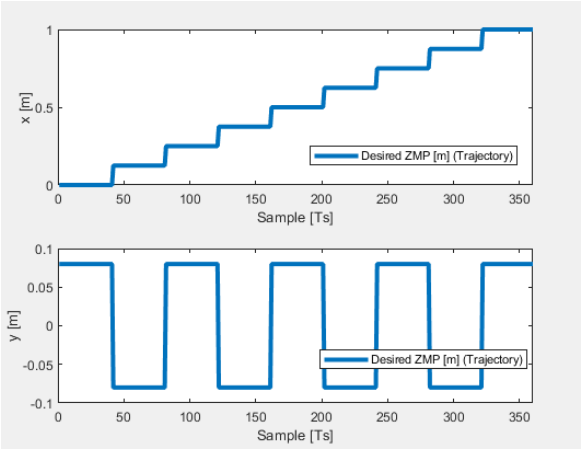

Figure 52: ZMP reference trajectory

A MPC controller utilises future information of the ZMP reference to ensure the CoM begins to move prior to the swing foot reaching the next foot step position. This is such that when the foot reaches the next foot step position, the ZMP will reside in the new single support phase polygon. A simple block diagram of the control architecture is shown in Figure 53 and a MATLAB Simulink implementation is illustrated in Figure 54.

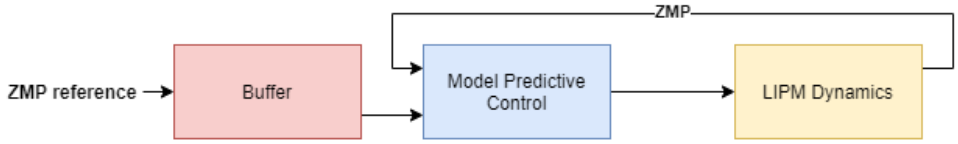

Figure 53: MPC [7]

The following Simulink file was based heavily on code sourced from [7].

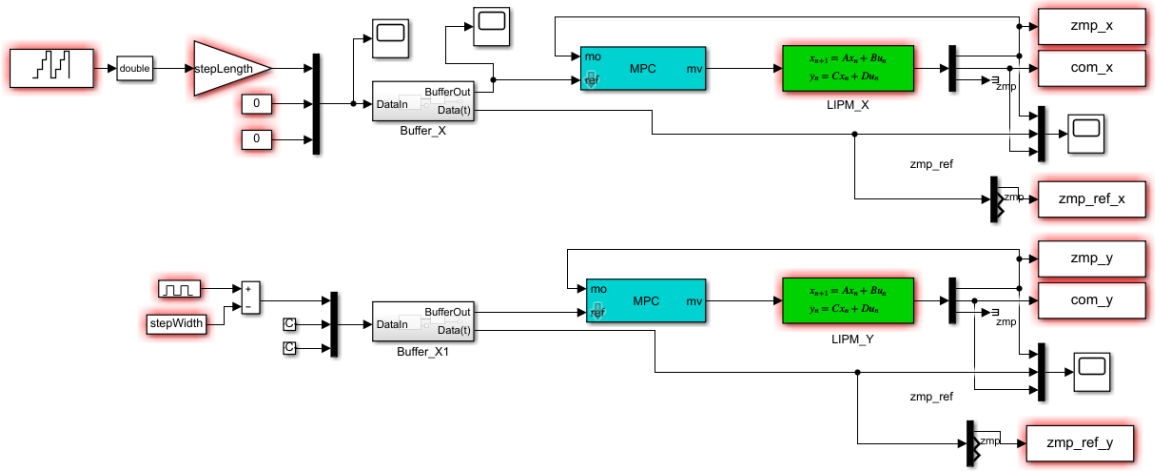

Figure 54: ZMP MPC implementation.

The MPC method successfully generating a walking pattern is illustrated in Figure 55 and 56 for gait parameters: `Step Length` = 0.075 [m], `Step Width` = 0.12 [m] and `Step Time` = 1 [s]. However, the MPC based approach was not extended to be implemented on the NUgus platform, with reasons further discussed in the following Section 6.4.

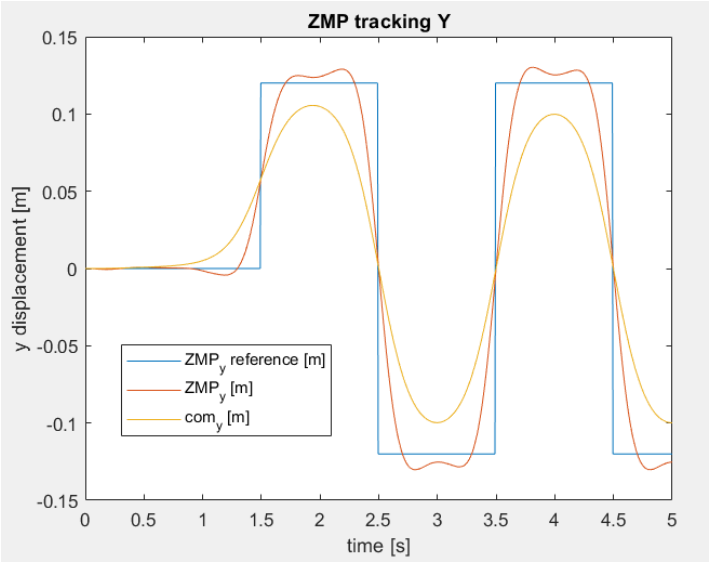

Figure 55: MPC ZMP trajectory Y.

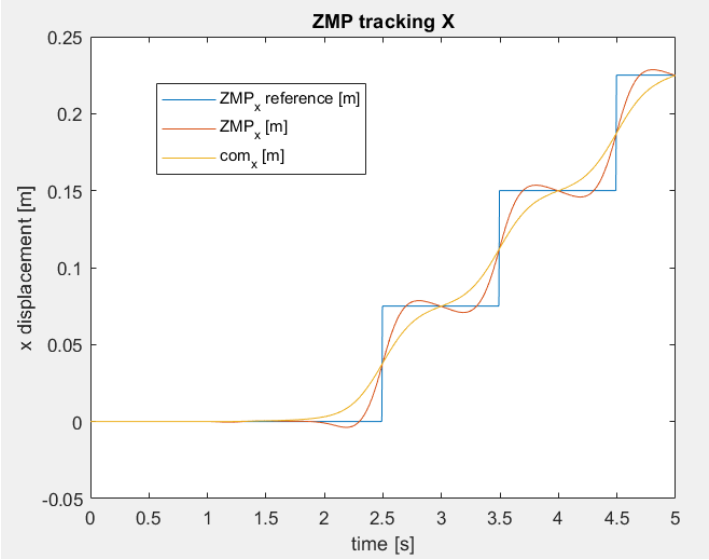

Figure 56: MPC ZMP trajectory X.

6.4. Preview Control Walking Pattern Generation

With the ultimate goal to eventually implement the developed walking generation methods within the NUbots code base, a decision was made to reduce the control design from MPC to a purely preview control based method. The rationale behind this is that the difficulty in implementing a preview control based approach when compared to a MPC solution within C++ is considerably reduced and the added benefits of using MPC are not currently being utilised.

Preview control is quite similar to MPC and exploits knowledge of the future information of the reference signals and/or the disturbances to greatly improve the performance of transient responses [32]. To design a preview control walking pattern generator, the method proposed by Katayama et al. [17] can be followed. Utilizing the discretised system listed in Equation 6.3 and provided with a ZMP reference p^{ref}, the performance index is specified as [16]:

$$J = \sum_{i=k}^{\infty} \{Q_e e(i)^2 + \Delta x^T(i) Q_x \Delta x(i) + R\Delta u^2(i)\} \tag{6.8}$$

where $e(i) = p(i) - p^{ref}(i)$ is the ZMP error, Q_e, $R > 0$ and Q_x is a 3×3 symmetric non-negative definite matrix. $\Delta x(k) = x(k) - x(k-1)$ is the incremental state vector and $\Delta u(k) = u(k) - u(k-1)$ is the incremental input [16]. Then, provided the ZMP reference can be previewed for N_L steps in the future, the following optimal controller can be implemented, which minimises the performance index 6.8.

$$u(k) = -G_i \sum_{i=0}^{k} e(k) - G_x x(k) - \sum_{j=1}^{N_L} G_p(j) p^{ref}(k+j) \tag{6.9}$$

where G_i, G_x and G_p(j) are the gains calculated from the weights Q_e, Q_x, R and the system parameters of Equation 6.3.

The preview controller listed in Equation 6.11 consists of three major components, the integral gain G_i on the tracking error, the preview gain $G_p(j)$ utilising the future reference and the state feedback gain G_x.

To compute the optimal gain, we first solve the following Riccati equation [20]:

$$\tilde{P} = \tilde{A}^T \tilde{P} \tilde{A} - \tilde{A}^T \tilde{P} \tilde{B} (R + \tilde{B}^T \tilde{P} \tilde{B})^{-1} B^T \tilde{P} \tilde{A} + \tilde{Q} \tag{6.10}$$

Then, utilising the $\tilde{P}$ solution to the Riccati equation, the optimal $\tilde{K}$ gains can be calculated via:

$$\tilde{K} = (R + \tilde{B}^T \tilde{P} \tilde{A})^{-1} \tilde{B}^T \tilde{P} \tilde{A} \tag{6.11}$$

Finally, we calculate the optimal preview gain using the following recursive algorithm:

Algorithm 3 Optimal preview gain calculation

1: **for** $i = 2$ to N **do**
2: $\quad G(i) = (R + \tilde{B}^T \tilde{P} \tilde{A})^{-1} \tilde{B}^T \tilde{X}(i-1)$
3: $\quad \tilde{X}(i) = {\tilde{A}_c}^T \tilde{X}(i-1)$
4: **end for**

where the initial conditions are $G(1) = -K_e$, $X(1) = -\tilde{A}_e \tilde{P} \begin{bmatrix} I \\ 0 \end{bmatrix}$ and N is the preview horizon.

The initial results of the preview control method tracking a Y ZMP reference are shown in Figure 57 for tuning values of $Q_e = 1$ and $R = 1e{-}3$. As observed, there is a considerable amount of error between the desired and achieved ZMP, and large amount of undershoot at the beginning. Since the ZMP is simplistically modelled based on a pendulum which is non-minimum phase, some amount of undershoot is necessary, however, the amount seen was considerably larger then results seen in literature [16].

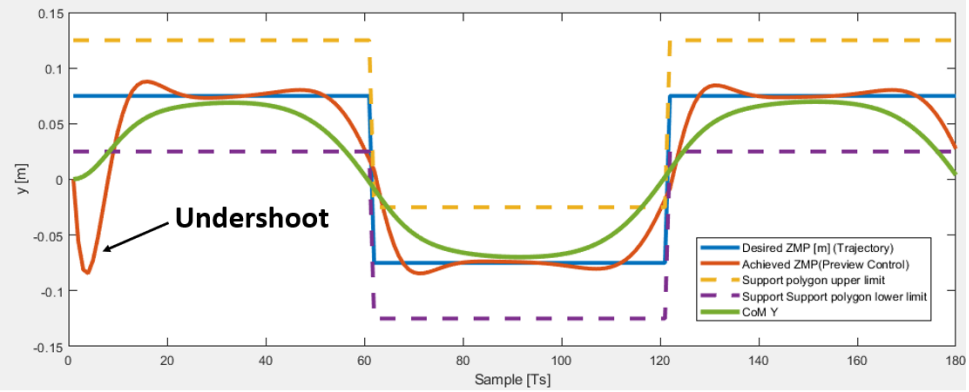

Figure 57: ZMP preview control undershoot.

To reduce the undershoot and tracking error, a 1.5s ZMP reference of 0 was added to the beginning, based purely on the results observed in [16]. The inclusion of this 0 reference had a major impact on the performance, greatly reducing the tracking error and undershoot, as depicted in Figure 58. The performance increase can be attributed to the 0 reference allowing the internal states to settle before moving. The preview controller is able to optimise the control input such that the `CoM Y` gradually starts moving prior to the reference switch, resulting in a much smoother response. Ultimately, this result further highlights the benefit of utilising a preview horizon within the control design.

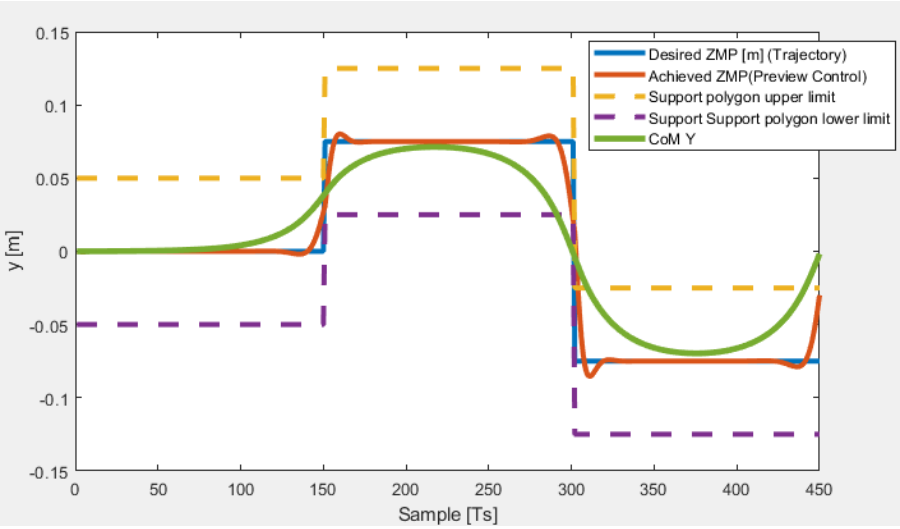

Figure 58: ZMP preview control with initial zero reference.

Furthermore, the inclusion of this 0 reference allowed for a far smaller R value, leading to tuning values of $R = 1e{-}6$ and $Q_e = 1$. With this smaller R value, the controller can take far more aggressive control action, further reducing the tracking error, leading to the results shown in Figure 59.

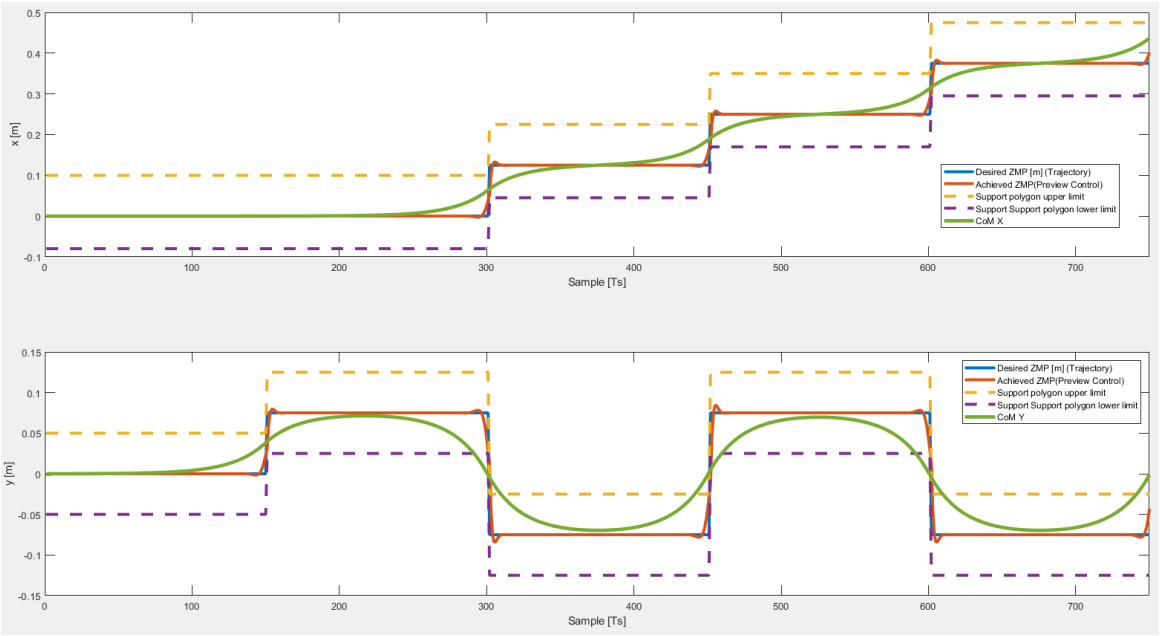

Figure 59: ZMP preview control results. `Ts = 0.001 [s]`

6.5. Implementing ZMP walking pattern on NUgus

To implement the ZMP walking pattern generated on the NUgus platform we must first define a coordinate system such that we can map the world space CoM trajectory $r^W_{C/W}*$, into the support foot reference frame $r^P_{C/P}*$, illustrated in Figure 60 [31]. The trajectory must be mapped into the support foot frame so that it can be incorporated into the inverse kinematics solver to allow joint space motions to be solved that result in the world space trajectory being tracked. Notably, the difficulty involved in this transformation involves the fact that the support foot is continuously changing position relative to the world frame, requiring the transform H^W_P to be dynamically updated after each step.

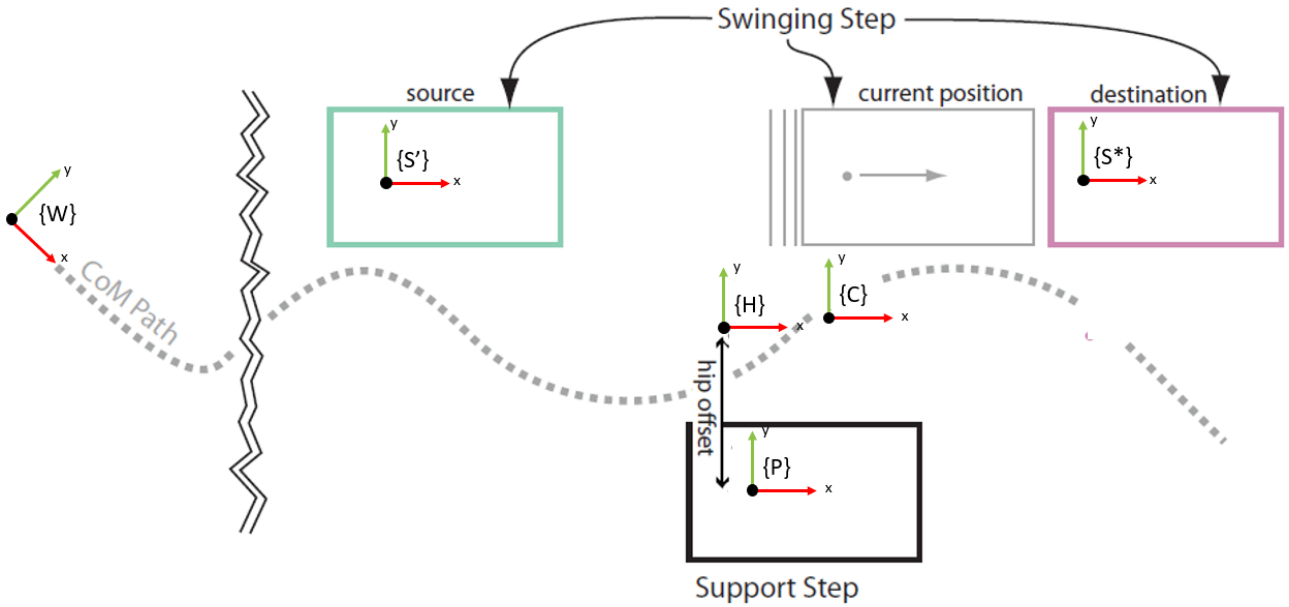

Figure 60: ZMP reference frames [31].

First, provided with a matrix $\mathbf{F}$ of footstep positions generated by the footstep planner discussed in Section 4.1 and the current footstep index i, the H^W_P transform is calculated as:

$$H^W_P = \begin{bmatrix} 1 & 0 & 0 & \mathbf{F}(i,1) \\ 0 & 1 & 0 & \mathbf{F}(i,2) \\ 0 & 0 & 1 & -h \\ 0 & 0 & 0 & 1 \end{bmatrix} R_y(\frac{\pi}{2}) \tag{6.12}$$

where h is the assumed constant height of the torso frame {H} above the ground plane and $\mathbf{R_y}$ is a 4×4 rotation matrix around the y axis of {H}.

Next, the desired world space CoM $r^W_{C/W}*$ trajectory can be mapped into the support foot space via the following:

$$r^P_{C/P}* = (H^W_P)^{-1} \begin{bmatrix} r^W_{C/W} \\ 1 \end{bmatrix} \tag{6.13}$$

By tracking a ZMP trajectory which passes through the footstep positions/support polygons, a CoM trajectory is naturally generated as an output of the system which ensures stability due to the linear relationship between the ZMP and CoM. Thus, to achieve a stable walking gait, both a CoM and swing foot trajectory must be tracked simultaneously, as illustrated in Figure 61. This can be achieved using numerical optimisation based inverse kinematics, with both desired trajectories incorporated in the cost function, leading to the following cost function:

$$q^* = \underset{q}{\operatorname{argmin}} \, ||(q - q_0)|| + ||(r^p_{S/P}(q) - r^p_{S/P}*)|| + ||(r^p_{C/P}(q) - r^p_{C/P}*)|| + \dots \\ \frac{1}{2}Tr(I - R_{S/P}) + \frac{1}{2}Tr(I - R_{6/0}) + \frac{1}{2}Tr(I - R_{12/0}) \tag{6.14}$$

where $r^p_{S/P}(q)$ is the swing-foot position, $r^p_{S/P}*$ is the desired swing-foot position, $r^p_{C/P}(q)$ is the bipeds CoM position, $r^p_{C/P}*$ is the desired CoM position, q is a vector of joint angles and q_0 is a vector of the previous solutions joint angles.

The term $\frac{1}{2}Tr(I - R)$ is equivalent to $(1 - cos(\theta))$ where θ is the angle associated with the rotation between frames, which drives θ to 0 [21]. $R_{12/0}$ and $R_{6/0}$ are rotations between torso frame and ankle pitch and $R_{S/P}$ is the rotation between support and swing foot. These terms are included in the cost function to orientate the base of the swing foot such that it is parallel with the floor when taking a step.

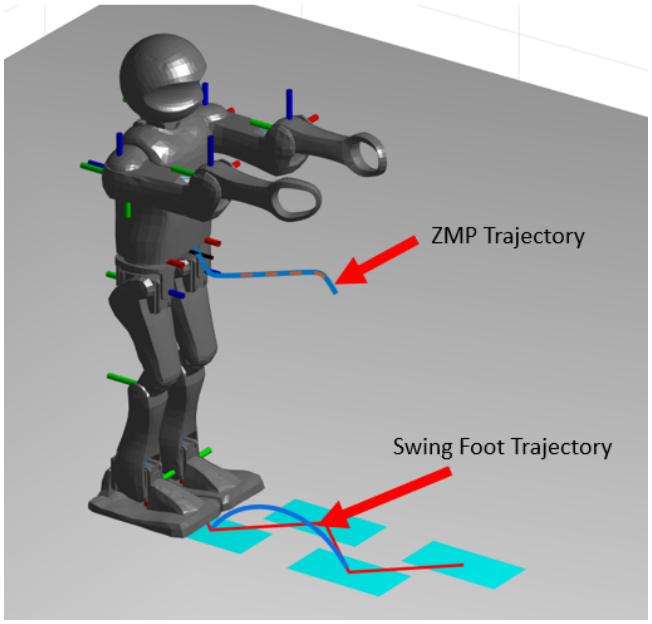

Figure 61: ZMP reference trajectories.

7. Simulation Environment

The following chapter aims to outline the components developed for validating walking routines in the WeBots simulation environment. Due to the global travel restrictions from COVID-19, Robocup 2021 was converted into a solely virtual competition. The official league simulator chosen was WeBots, shown in Figure 62, an open-source and multi-platform desktop application used to simulate robots developed by Cyberbotics Ltd. This led to choosing WeBots as the simulation environment to validate the walking routines, as work completed for this project could directly assist the team in qualifying for the 2021 competition. The competition still aims at providing an environment for transfer learning through sim2real experiments. Therefore, the robot model used in simulation must aim to resemble the robot used in real competition. The WeBots simulation environment relies on ODE for physics simulation, an open source, high performance library for simulating rigid body dynamics. It is has advanced joint types and integrated collision detection and friction. ODE is widely used in video games, simulation and 3D authoring tools, for simulating vehicles and objects in virtual reality environments [35].

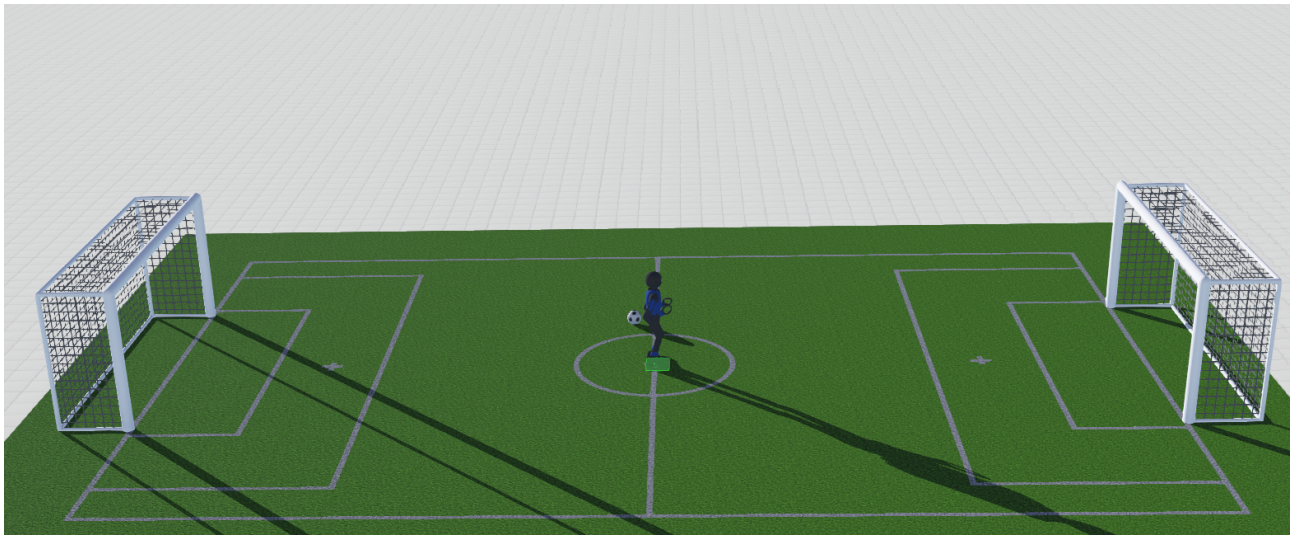

Figure 62: WeBots.

7.1. Robot Model

Working closely with the NUbots team, a robot model of the NUgus platform was created for use in the virtual competition and for testing walking gaits for this project. Robot models within WeBots are built using a .proto file which is a text file-based description of a robot, similar to commonly used URDF (Unified Robot Description Format) files. A URDF file for the original Igus robot was sourced from the Nimbro team and utilising a urdf2webots python script, a .proto file was generated. This .proto file provided a good starting point for creating a model of the NUgus platform, however, numerous updates were required to realistically model the robot and match the kinematics derived in Section 19. The WeBots model created is shown in Figure 63 which is comprised of simplified versions of the NUgus mesh files in order to improve the simulators performance.

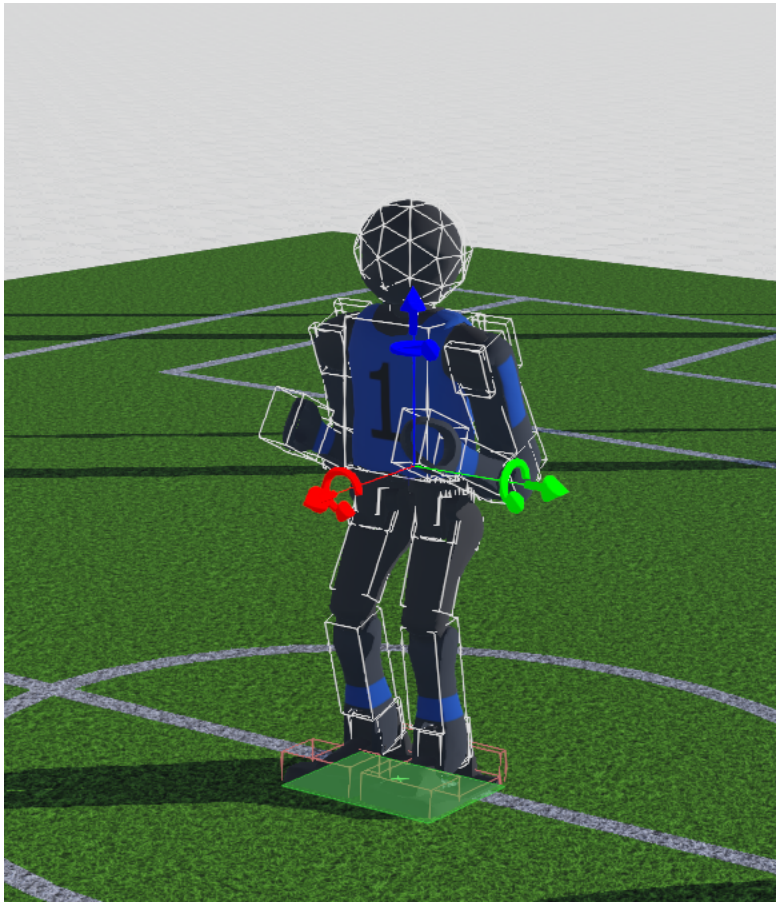

Figure 63: WeBots Model.

The servo motors used in the NUgus robot are Dynamixel servos, shown in Figure 64. The upper body are Dynamixel MX64AR servos, with two in each shoulder, one in each elbow, and two in the neck. The legs use Dynamixel MX106 servos, with two in each ankle, three in each hip, and one in each knee. The WeBots models `HingeJoint` and `RotationalMotor` nodes were modelled with maximum torque, maximum velocity, damping constants, and static friction parameters. Values for these parameters were sourced from the datasheets for each MX-106 and MX-64 Dynamixel servo and are listed in Table 6. Additionally, the Dynamixel motors position sensors use a 12 bit rotary encoder, thus, for each motor a `PositionSensor` a resolution of $2\pi/2^{12} \approx 0.0015$ was used.

Figure 64: Dynamixel MX-64 and MX-106 [26].

Table 6: WeBots RotationalMotor parameters

Motor Type	**Parameter**	**Variable Name**	**Value**
MX106	maxTorque	MX106-torque	10.00
MX106	maxVelocity	MX106-vel	5.76
MX106	dampingConstant	MX106-damping	1.23
MX106	staticFriction	MX106-friction	2.55
MX64	maxTorque	MX64-torque	7.30
MX64	maxVelocity	MX64-vel	8.17
MX64	dampingConstant	MX64-damping	0.65
MX64	staticFriction	MX64-friction	1.73

7.2. Simulation Controller and Interface

The WeBots simulator provides an API for creating controllers that can interface with WeBots models to control servos and retrieve sensor measurements. WeBots provides support for writing these controllers C++, Java, Python and MATLAB. A controller for the NUgus WeBots model was developed using MATLAB to execute the walking gaits generated provided with a set of desired joint angles. The software architecture for this is illustrated in Figure 65.

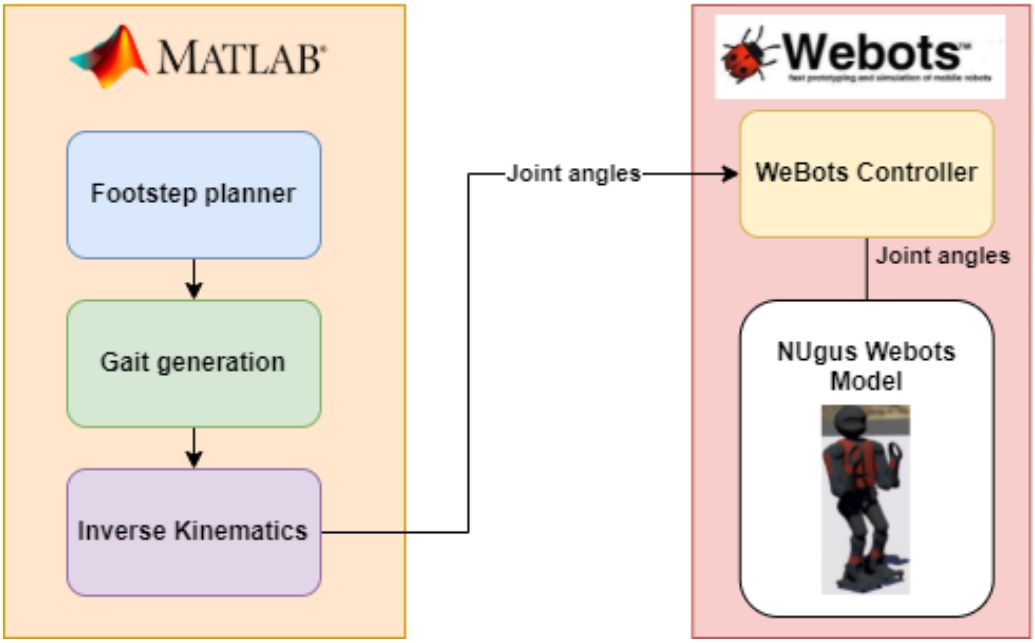

Figure 65: WeBots Controller.

Additionally, the implementation of the walking routines on the simulated model was a considerably more difficult task then first anticipated. The process required a large amount of time hand-tuning the following parameters:

- Motor PID Gain - gains of the individual motor controllers
- Velocity [rad/s] - the rotational velocity of the servos
- Step Height [m] - Peak height of step
- Step Length [m] - Length of full step
- Step Time [s] - Time for a full step
- Ts - sampling rate

To reduce the time taken to hand tune these parameters, an interface was developed between MATLAB and WeBots which allowed hundreds on varying gait parameters to be trialled and evaluated, illustrated in Figure 66. The 'fitness score' of a walking gait was determined based on the distance travelled in meters after a specified amount of time. This software lays the ground work for an optimisation routine to be developed which could optimise the gait parameters to increase performance specifications such as speed, stability and energy use. Only a broad search approach was implemented in this project, which simply trialled a vast number of predefined gait parameter values and stored the results.

Nonetheless, this provided the ability to run hundreds of walking routines autonomously and identify good parameters. The routine could be left to run for large amounts of time overnight preventing the need for human tuning.

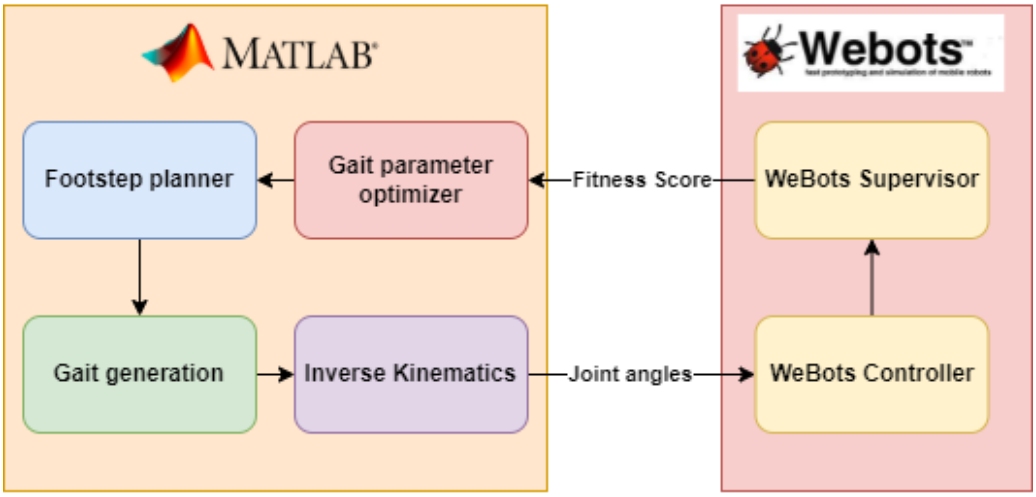

Figure 66: WeBots Optimization.

8. Results

The following chapter includes the results of the static and dynamic walking routines in the WeBots simulation environment.

8.1. Quasi-static Method Simulation Results

The quasi-static method was tested and evaluated using the Webots interface discussed in Section 7.2. To successfully implement the quasi-static method, numerous iterations of the cost function and constraints of the inverse kinematics solver were needed. Notably, the inclusion of the terms of the form, $\frac{1}{2}Tr(I - R)$, which ensure the feet are correctly orientated to the ground was critical for performing a stable support foot switch. Without this term, many of the first walking attempts resulted in the robot falling over due to the feet clipping the ground undesirably after taking a step. The static walk was successfully implemented on the NUgus platform within simulation as depicted in Figure 67.

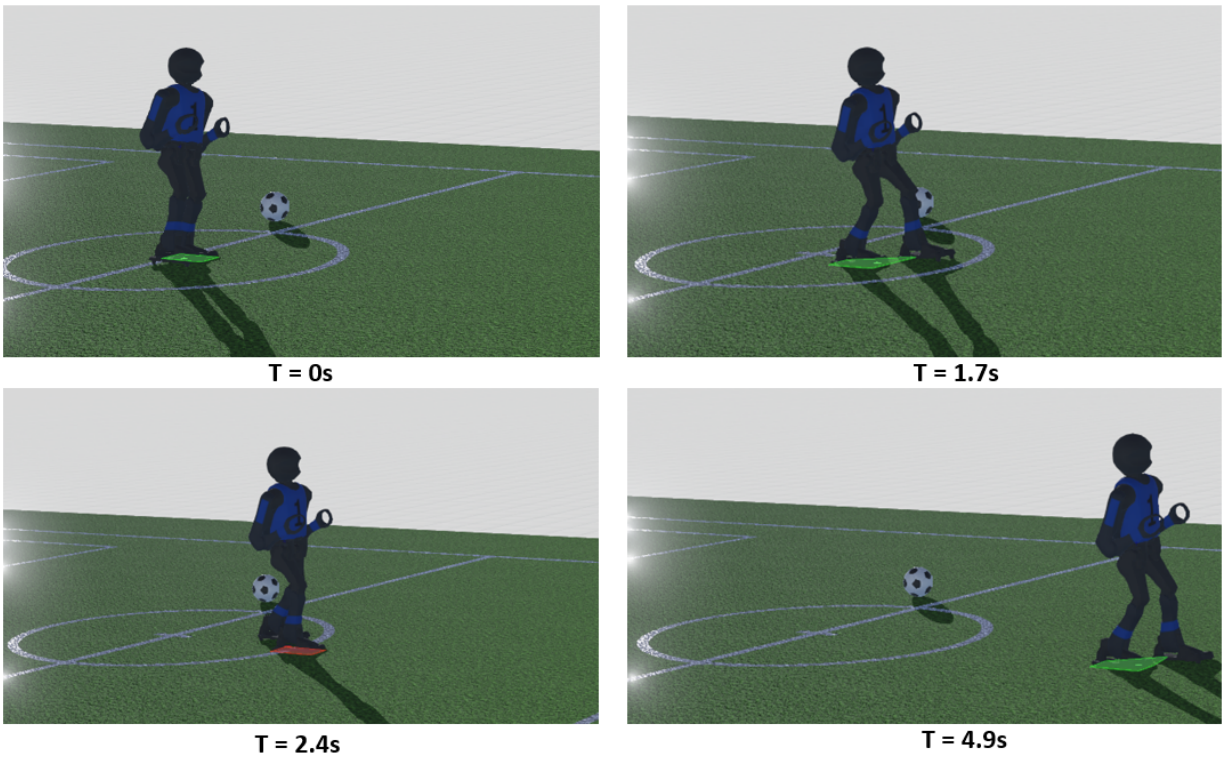

Figure 67: WeBots Static Walking.

To validate the static walking routine, the CoM position of the robot during the walking cycle in simulation was recorded and plotted against the reference trajectory. As observed in Figure 68, during a walking routine with gait parameters `Sampling Time Ts` = 0.05 [s], `step length x` = 0.2 [m] and `step time` = 4 [s], the reference CoM trajectory generated by the static routine is tracked within simulation. It is worth noting that there is considerable error when the support foot switch occurs, which can be attributed to the assumption made that the CoM switch between the support feet is instantaneous, which is, in reality, not feasibly possible. To improve the tracking error and ensure the quasi-static constraint is met during the whole walking cycle an extension to the gait generation method should be included which shifts the torso over the next support foot during the double support phase. Nonetheless, the period of time the static constraint is violated was found to be negligible and a stable walk of 0.05 [m/s] was achieved.

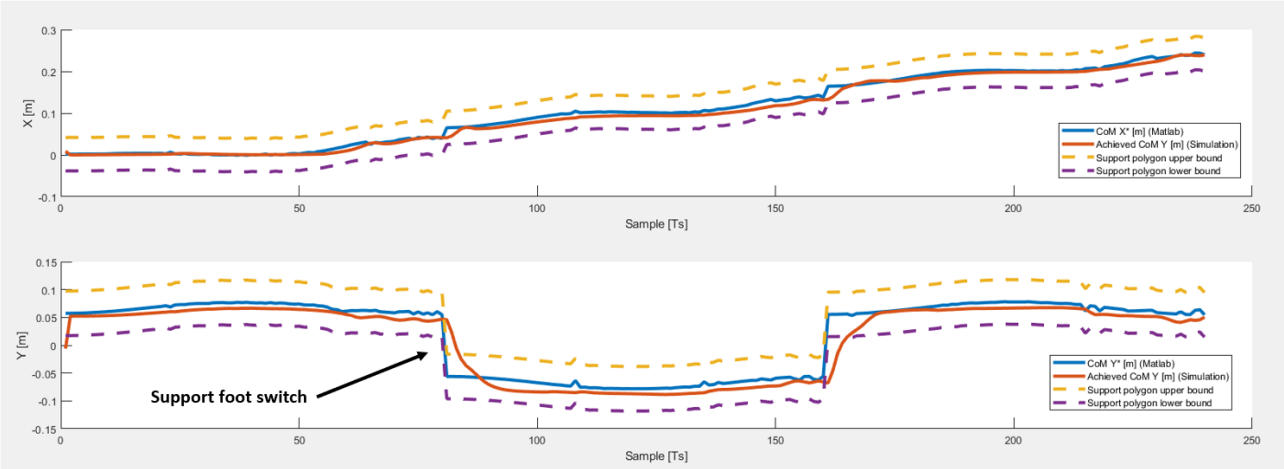

Figure 68: Static simulation results for `Step time [s] = 2`.

To increase the speed of the walking routine, the `step time` was reduced to 1 [s]. As observed, the error between the reference trajectory increases considerably. This increase in error can also be attributed to the neglected inertia of robot coming into effect when shifting between support feet.

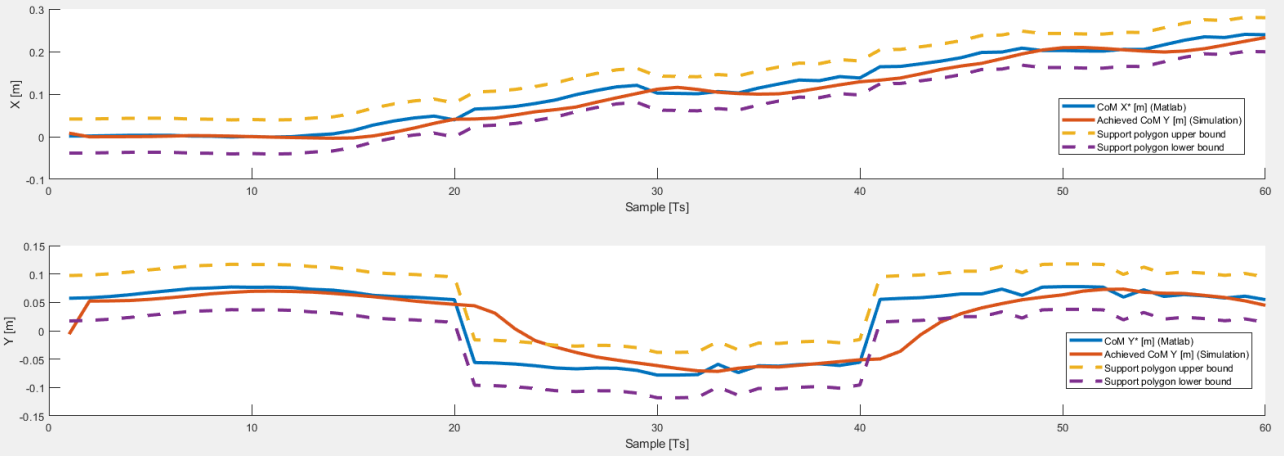

Figure 69: Static simulation results for `Step time [s] = 1`.

To find gait parameters which optimised the maximum achievable forward speed of the quasi-static routine, the generation and simulation pipeline shown in Figure 66 was utilised. This automated the process of trialling varying walking routines with different gait parameters: `step length x` and `step time` within WeBots and the results for this optimisation process are depicted in Figure 70. Each of the trials ran for a total walk time of 14 [s]. Notably, for both `step time` of 0.25 and 0.5 [s], the `distance travelled` vs `step length x` relationship is non-linear, which is due to the static routine being unstable for longer strides with short step periods. However, for `step time` of 1 and 2 [s], the assumptions made within the static routine are more reasonable, resulting in a linear relationship. From these results, the fastest walking routine achieved was 0.35 m/s with the following gait parameters: `step length x` = 0.25 [m], `step length y` = 0 [m], `step height` = 0.06 [m], `step time` = 0.25 [s] and `Ts` = 0.05 [s].

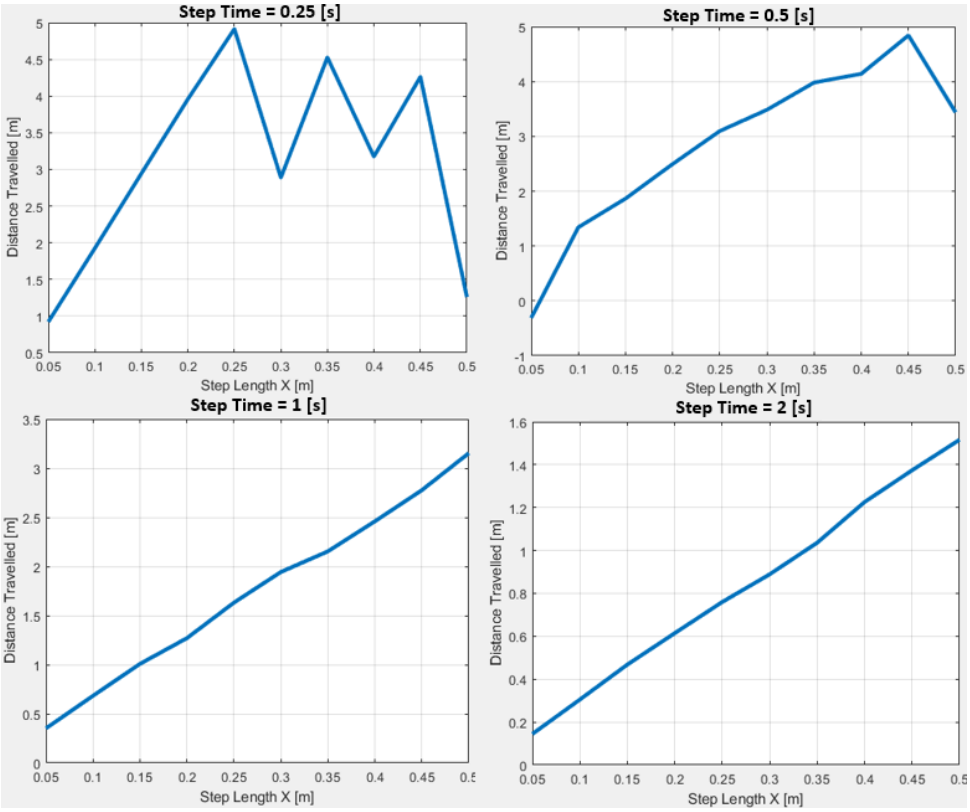

Figure 70: Static simulation results for `Step time [s] = 0.5`.

8.2. ZMP Simulation Results

The ZMP method was tested and evaluated using the Webots interface discussed in Section 7.2. It was shown that using the ZMP stability criterion, a dynamically stable gait could be achieved in simulation, illustrated in Figure 71.

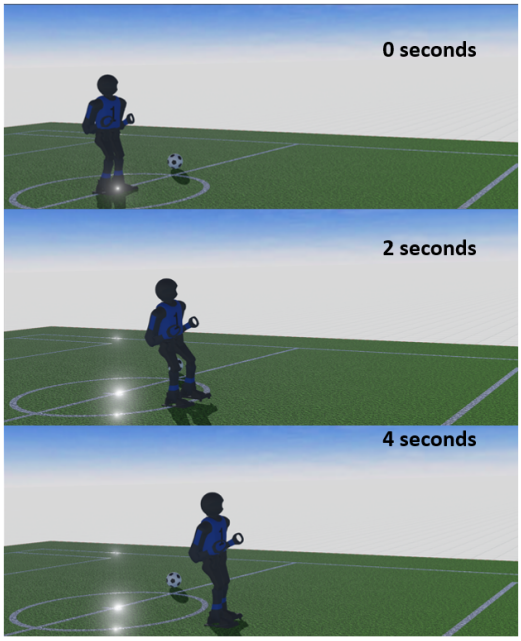

Figure 71: ZMP Walking WeBots.

To validate that the ZMP walking gait was being successfully tracked, the CoM of the robot model was retrieved from the WeBots simulation environment during the walking cycle. The results of the CoM trajectory tracking in WeBots are depicted in Figure 72. As observed, the trajectory is tracked considerably well in simulation, which results in a stable walking gait.

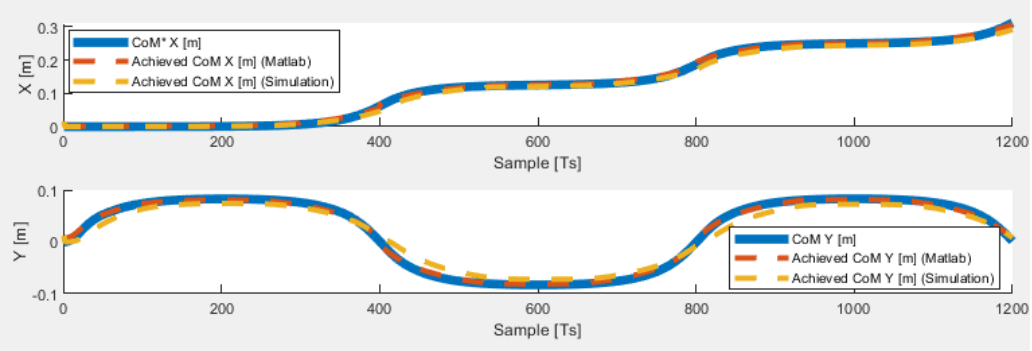

Figure 72: ZMP Method CoM tracking WeBots.

Additionally, it was discovered that the dynamic approach was far more reliable for diagonal walk commands. For the static method, the range of motion is greatly restricted as the centre of mass is required to remain above the support foot at all times, however, it was observed that for the ZMP walking gaits, considerably less torso sway is required, allowing the robot to more reliably track diagonal walking paths.

8.3. Discussion

Both the static and dynamic methods were shown to be practical methods for achieving stable walking gaits for forward motion. However, interestingly, it was observed for the algorithms developed, that the dynamic approach was considerably better at strafing and walking along diagonal walking paths. This observation was attributed to the fact that the dynamic method exhibited less upper body sway while walking, leading to a far more stable walking motion. Robotic soccer is a fast-paced dynamic environment which requires highly responsive and agile motion. For this reason, it is sensible to favour the ZMP approach for implementation in the Robocup competition.

Within MATLAB implementations for gait identical parameters and sampling time of T = 0.05 [s], the total time to generate a walking pattern for the static method was 120.352 [s] while the dynamic method was 94.436 [s]. The most computationally expensive component of the algorithms developed is the non-linear optimisation based inverse kinematics, and, since the static method has additional inequality constraints, the computation time required is higher than the ZMP method. However, while the ZMP method is faster for a sampling time of 0.05s, the results of the preview controller were found to be unusable for achieving a stable walk within simulation. To achieve good ZMP tracking performance, a sampling time of 0.01 [s] was required, drastically increasing the total computation time to 498.510 [s]. In conclusion, the walking patterns generated are open loop, thus, they can be pre-calculated offline and later implemented on the NUgus platform which means the total time taken is not a critical factor for performance. However, if the methods are to implemented on the real platform in an online setting, further investigation is required on the computational requirements for both methods in a C++ based implementation.

9. Conclusion

In this project, both a quasi-static and dynamic walking routine were successfully implemented on the Nugus platform. The walking routines were validated and compared within the simulation environment WeBots. Static walking gaits were generated via the inclusion of non-linear constraints in a numerical optimisation based inverse kinematics solver. Dynamic walking gaits were generated via the use of a simple linear inverted pendulum model (LIPM) and a preview controller, which outputs dynamically balanced center of mass trajectories. Simulation results show that both the proposed methods are capable of providing stable and reliable walking gaits. To generate the walking gaits, a completely modular MATLAB software suite was developed which decoupled components of the gait generation process, providing a blue print for transfer to a C++ implementation within the main code base.

System identification was performed to determine the length, mass and effective CoM of each of the limbs of the NUgus robot platform. Both physical and CAD based system identification methods were implemented to obtain accurate parameters. A software pipeline was setup to aid the NUbots team in maintaining and updating the models parameters. While there is expected to be a considerable simulation to reality gap, large emphasis was placed on accurately modelling the real hardware such that the work completed can eventually be transferred to the real robot.

While the walking routines were not implemented on the real hardware, the work presented provides a good starting point for developing a complete walk engine within the NUbots main code base, and provides numerous tool for the NUbots team. For example, the quasi-static inverse kinematics solution proposed was utilised for generating a statically stable kicking pattern which allowed the NUbots team to qualify for the 2021 virtual Robocup competition. Additionally, the C++ benchmarks and implementation of forward and inverse kinematics modules have laid the ground work for future motion related developments for the team going forward.

10. Future work

The results of the static and dynamic walking gait implementations indicate that either method is a suitable approach for achieving stable walking locomotion for use in the Robocup competition. There are many areas of potential research and improvements that were identified in completing this project which include:

- Implementation of the walking generation methods in C++ for integration with the main NUbots codebase
- While the walking gaits generated were stable in simulation, testing and evaluation of the walking gaits on the real hardware is crucial for validating if an open-loop solution is robust enough against disturbances introduced in a real world environment
- The step planner implemented provides a good 'starting point', however, lacks the ability to rotate the robot in the yaw direction. Extension of the step planner to generate curved walking paths will be critical to extend and implement the walking generation method for use in the Robocup competition
- An assumption is made within the static method that the support foot switch is instantaneous, however, this assumption results in instability for faster static walking gaits. To improve the performance of the static method, a double support phase CoM shift could be explored to produce a more reliable gait.

References

[1] , 2020. World Robotics 2020 Report. Tech. rep., International Federation of Robotics.

[2] Amini, A., Soleimany, A., Karaman, S., Rus, D., 05 2018. Spatial uncertainty sampling for end-to-end control.

[3] Billard, A., Kragic, D., 06 2019. Trends and challenges in robot manipulation. Science 364, eaat8414.

[4] BostonDynamics, 2021. Inside the lab: How does atlas work?
URL https://www.youtube.com/watch?v=EezdinoG4mk

[5] Buschmann, 2010. Simulation and control of biped walking robots. Ph.D. thesis.

[6] Caron, S., 2021. How do biped robots walk?
URL https://scaron.info/teaching/how-do-biped-robots-walk.html

[7] Castro, S., 2019. Walking robot control: From pid to reinforcement learning. MATLAB Blog.
URL https://blogs.mathworks.com/racing-lounge/2019/04/24/walking-robot-control/

[8] Chalup, S., 2020. Nubots.
URL https://www.newcastle.edu.au/research/centre/newcastle-robotics-laboratory/nubots

[9] Coldewey, 2018. Honda reportedly retires the iconic asimo.
URL https://techcrunch.com/2018/06/28/honda-reportedly-retires-the-iconic-asimo/

[10] Collins, S., Ruina, A., Tedrake, R., Wisse, M., 2005. Efficient bipedal robots based on passive-dynamic walkers. Science 307 (5712), 1082–1085.
URL https://science.sciencemag.org/content/307/5712/1082

[11] Dekker, M., 2009. Zero-moment point methodforstable biped walking. Master's thesis, Eindhoven, University of Technology.

[12] Dynamics, B., 2021. Atlas.
URL https://www.bostondynamics.com/atlas

[13] Flemming, T., 2019. Evaluating and minimizing the reality gap in the domain of robocup humanoid soccer.

[14] Honda, 2021. Asimo: Frequently asked questions.
URL https://asimo.honda.com/downloads/pdf/honda-asimo-robot-fact-sheet.pdf

[15] Honda, 2021. Honda robotics.
URL https://global.honda/innovation/robotics/ASIMO.html

[16] Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H., 2003. Biped walking pattern generation by using preview control of zero-moment point. In: 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422). Vol. 2. pp. 1620–1626 vol.2.

[17] KATAYAMA, T., OHKI, T., INOUE, T., KATO, T., 1985. Design of an optimal controller for a discrete-time system subject to previewable demand. International Journal of Control 41 (3), 677–699.
URL https://doi.org/10.1080/00207185089611 56

[18] Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., Matsubara, H., Mar. 1997. Robocup: A challenge problem for ai. AI Magazine 18 (1), 73.
URL https://ojs.aaai.org/index.php/aimagazine/article/view/1276

[19] Kolawole, 2013. What if this atlas shrugged? — darpa unveils new humanoid robot.
URL https://www.washingtonpost.com/news/innovations/wp/2013/07/12/what-if-this-atlas-shrugged-darpa-unveils-new-humanoid-robot/

[20] Liu, R. E., 2015. Walking pattern generation for humanoid robots.

[21] Mahony, R., Hamel, T., Pflimlin, J.-M., 07 2008. Nonlinear complementary filters on the special orthogonal group. Automatic Control, IEEE Transactions on 53, 1203 – 1218.

[22] NUbots, 2021. Overview and specifications.
URL https://nubook.nubots.net/system/hardware/overview

[23] Pfeifer, R., Iida, F., Bongard, J., 12 2005. New robotics: Design principles for intelligent systems. Artificial life 11, 99–120.

[24] Razlanyusoff, A., Zalani, M., Suffian, M., Taib, M., 12 2011. Literature review of optimization technique for chatter suppression in machining. JOURNAL OF MECHANICAL ENGINEERING AND SCIENCES 1.

[25] Rhoban, 2021. onshape-to-robot.
URL https://github.com/Rhoban/onshape-to-robot

[26] robot-r us, 2021. Dynamixel mx-106.
URL https://robot-r-us.com.sg/p/dynamixel-mx-106

[27] Ruggiero, F., Petit, A., Serra, D., Satici, A., Cacace, J., Donaire, A., Ficuciello, F., Buonocore, L., Fontanelli, G., Lippiello, V., Villani, L., Siciliano, B., 03 2018. Nonprehensile manipulation of deformable objects: Achievements and perspectives from the robdyman project. IEEE Robotics and Automation Magazine PP, 1–1.

[28] Sherikov, 2012. Model predictive control of a walking bipedal robot using online optimization. Master's thesis.

[29] Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G., 2009. Modelling, planning and control. Advanced Textbooks in Control and Signal Processing. Springer,.

[30] SooHoo, 2016. Atlas.
URL https://robots.ieee.org/robots/atlas2016/

[31] Strom, J., Slavov, G., Chown, E., 06 2009. Omnidirectional walking using zmp and preview control for the nao humanoid robot. pp. 378–389.

[32] Takaba, K., 09 2003. A tutorial on preview control systems. pp. 1388 – 1393 Vol.2.

[33] Vukobratović, M., Borovac, B., Potkonjak, V., 2007. Towards a unified understanding of basic notions and terms in humanoid robotics. Robotica 25 (1), 87–101.

[34] Vukobratovic, M., Borovac, B., 03 2004. Zero-moment point - thirty five years of its life. I. J. Humanoid Robotics 1, 157–173.

[35] WeBots, 2021. Ode: Open dynamics engine.
URL https://cyberbotics.com/doc/reference/ode-open-dynamics-engine

A. Time Log

A time log of the time spent on the project over the first and second semester is listed in Figure 73 and Figure 74. A total of 387 hrs was put into the project.

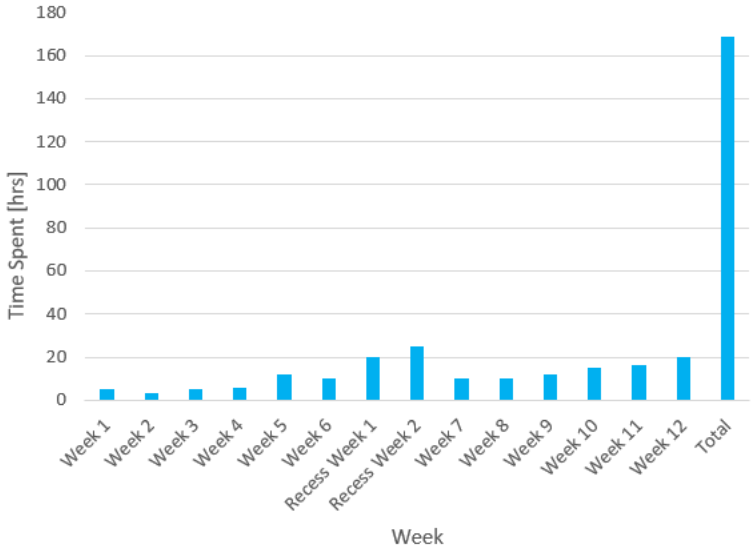

Figure 73: Time Log Semester 1.

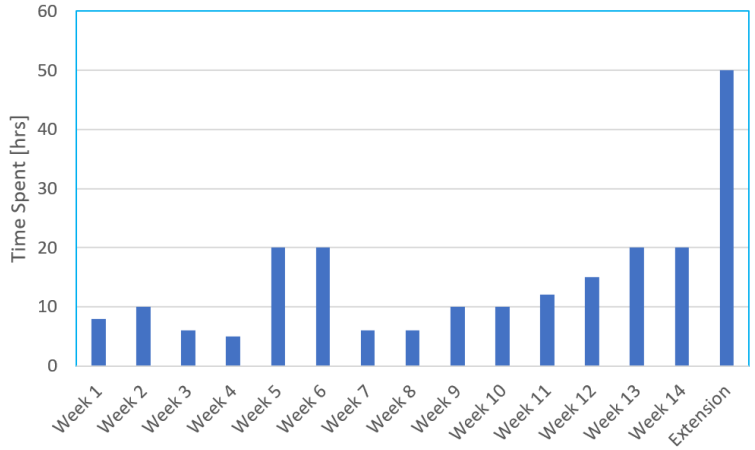

Figure 74: Time Log Semester 2.

B. Code

MATLAB Gait Generation Code: https://github.com/TheCoolKidsTable/Walk

C++ Inverse Kinematics Code: https://github.com/TheCoolKidsTable/InverseKinematics

WeBots model: https://github.com/TheCoolKidsTable/Walk/tree/main/protos/robot/NUgus

C. NUgus Onshape Assembly

Onshape Assembly: https://cad.onshape.com/documents/7a93aa6f68a144f3337682ca/w/21ea40de0160de/beac31a00af2912b29fa71ac?renderMode=0&uiState=61b1433d30f89817088608db

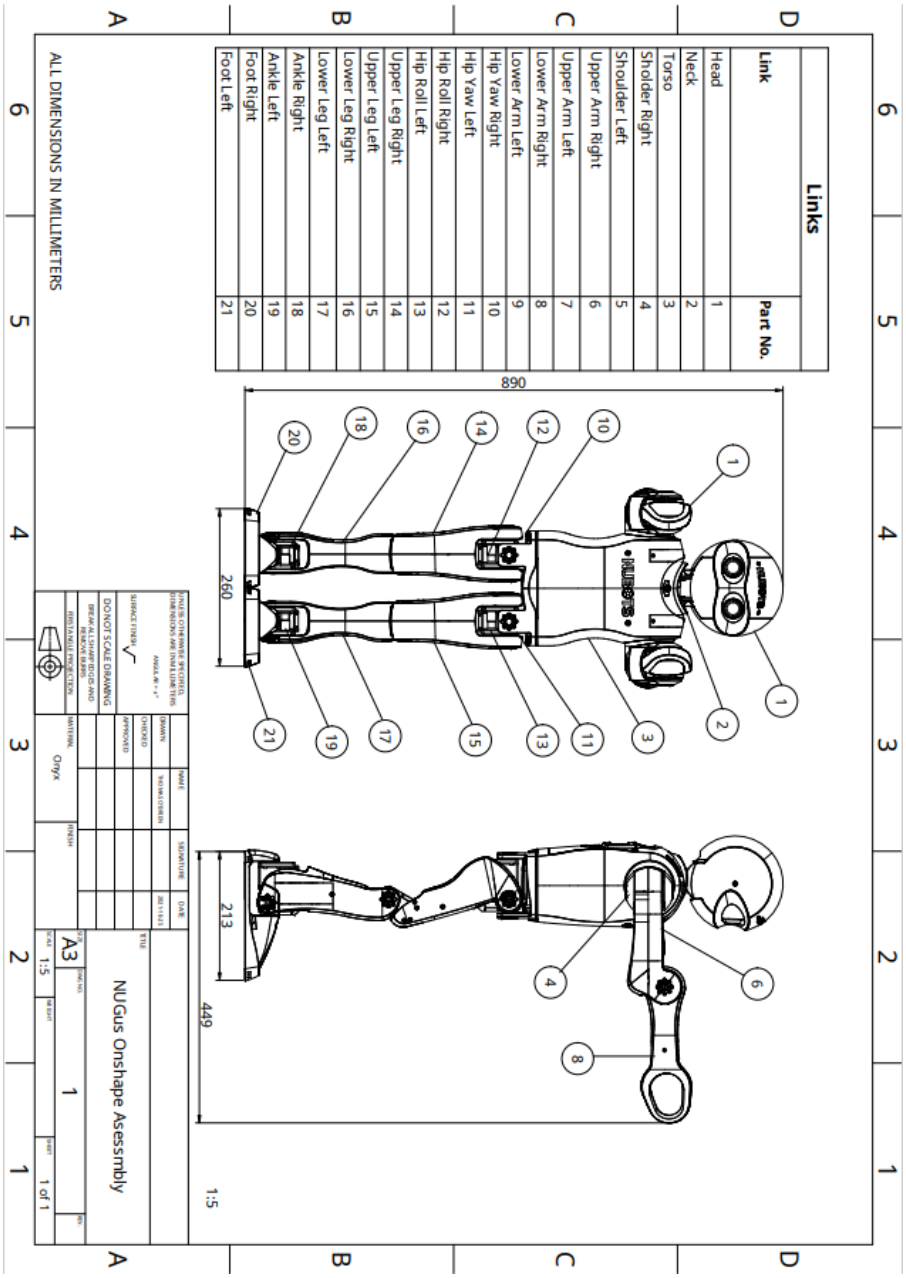

Figure 75: NUgus assembly

D. C++ Forward and Inverse Kinematics Benchmarking

The following appendix covers the performance and evaluation of numerous implementations of forward and inverse kinematics investigated in MATLAB and C++. The code for C++ libraries pinnochio, RBDL and NLopt benchmarks can be found here : https://github.com/TheCoolKidsTable/InverseKinematics. Both Pinnochio and RBDL are light weight C++ libraries for efficient rigid body dynamics utilised for implementing forward kinematic models. NLopt is an open-source library for nonlinear optimisation which was utilised to develop custom inverse kinematics solutions with non-linear constraints.

To evaluate forward kinematics, a set of all 0 joint angles were passed to each of the FK implementations listed in Table 7. Each FK implementation was called 100,000 times and the total time was recorded. As observed, the C++ library Pinocchio's FK module is the quickest of all four implementations.

Table 7: Forward Kinematics Benchmark

Case	**Total time (100,000 calls) [s]**	**Average time/call [μs]**
MATLAB Kinematics Class	40.95	409.5
MATLAB Robotic Systems Toolbox	229.40	2294.0
Pinocchio	0.082981	0.829881
RBDL	0.371116	3.71116

To evaluate the inverse kinematics implementations, an arbitrarily chosen feasible desired pose was chosen and passed to each of the IK implementations listed in Table 7. Each IK implementation was called 1,000 times and the total time was recorded. As observed, the C++ library RBDL's IK module is the quickest of all four implementations, which utilises the Levenberg-Marquardt optimisation algorithm to find a solution to the inverse kinematics problem.

Table 8: Inverse Kinematics Benchmark

Case	**Algorithm**	**Total time (1000 calls) [s]**	**Average time/call [s]**
MATLAB fmincon	sqp	1127.0	1.127
MATLAB fmincon	interior-point	2295.8	2.30
MATLAB Robotic Systems Toolbox	BFGSGradientProjection	59.6	0.05963
pinnochio & NLopt	COBLYA	8.95	0.00895
pinnochio	CLIK	4.15	0.00415
RBDL	Levenberg-Marquardt	0.0319228	0.0000319
RBDL with constraints	Levenberg-Marquardt	2.4204	0.0024
RBDL & NLopt	COBLYA	5.80412	0.00580

E. URDF Generation Instructions

The following instructions outline how to generate a URDF file from the Onshape model.

First, install the `onshape-to-robot` api by following this documentation: https://onshape-to-robot.readthedocs.io/en/latest/installation.html

1. Clone the code:

```
nerd@basement $ git clone https://github.com/TheCoolKidsTable/Walk
```

2. Navigate to the URDF folder:

```
nerd@basement $ cd Walk/urdf
```

3. Run the following script to generate the URDF file:

```
nerd@basement $ onshape-to-robot NUgus
```

4. A URDF will be generated from the Onshape model

F. Homogeneous Transformations

The following appendix lists some commonly used homogeneous transformation matrices utilised throughout the report.

Translations

$$Tran_x(x) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.1}$$

$$Tran_y(y) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.2}$$

$$Tran_z(z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.3}$$

Rotations

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) & 0 \\ 0 & sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.4}$$

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.5}$$

$$R_z(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 & 0 \\ sin(\psi) & cos(\psi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.6}$$

G. Forward Kinematics Parameters

Table 9: FKM details

Idx	Link Name	Joint Name	Position (x,y,z)	Orientation (r,p,y)	Parent link	Child link
1	*right_hip_yaw*	*right_hip_yaw*	(-1.16226e-16 -0.055 0.000499999)	(0 0 0)	*torso*	*right_hip_roll*
2	*right_hip_roll*	*right_hip_roll*	(-0.04625 6.93889e-18 -0.0545)	(-2.62033e-30 1.5708 0)	*right_hip_yaw*	*right_upper_leg*
3	*right_upper_leg*	*right_hip_pitch*	(-2.20657e-15 2.9564e-05 0.06895)	(-1.5708 -4.44818e-05 0)	*right_hip_roll*	*right_lower_leg*
4	*right_lower_leg*	*right_knee_pitch*	(0.199541 -2.33261e-05 -0.0015)	(0 0 -0.00523599)	*right_upper_leg*	*right_ankle*
5	*right_ankle*	*right_ankle_pitch*	(0.199419 -0.00566882 0.0015)	(0 0 0)	*right_lower_leg*	*right_foot*
6	*right_foot*	*right_ankle_roll*	(0 -6.94565e-06 -0.000982442)	(1.5708 0 0)	*right_ankle*	
7	*left_hip_yaw*	*left_hip_yaw*	(0 0.055 0.000499999)	(0 0 0)	*torso*	*left_hip_roll*
8	*left_hip_roll*	*left_hip_roll*	(-0.04625 6.93889e-18 -0.0545)	(0 1.5708 0)	*left_hip_yaw*	*left_upper_leg*
9	*left_upper_leg*	*left_hip_pitch*	(05 2.9564e-05 0.06895)	(-1.5708 0 0)	*left_hip_roll*	*left_lower_leg*
10	*left_lower_leg*	*left_knee_pitch*	(0.199541 -2.33261e-05 -0.0015)	(0 0 -0.00523599)	*left_upper_leg*	*left_ankle*
11	*left_ankle*	*left_ankle_pitch*	(0.199419 -0.00566882 -0.0002)	(0 0 0)	*left_lower_leg*	*left_foot*
12	*left_foot*	*left_ankle_roll*	(0 -6.94565e-06 -0.000982442)	(1.5708 03 0)	*left_ankle*	
13	*left_shoulder*	*left_shoulder_pitch*	(3.99598e-16 0.077 0.17775)	(-1.5708 06 0)	*torso*	*left_upper_arm*
14	*left_upper_arm*	*left_shoulder_roll*	(0.0100012 -0.0226 0.0479124)	(1.5708 0 0)	*left_shoulder*	*left_lower_arm*
15	*left_lower_arm*	*left_elbow_pitch*	(0.1575 -0.0235 0.0033)	(-1.5708 0 0)	*left_upper_arm*	
16	*right_shoulder*	*right_shoulder_pitch*	(0 -0.077 0.17775)	(-1.5708 0 0)	*torso*	*right_upper_arm*
17	*right_upper_arm*	*right_shoulder_roll*	(0.0100012 -0.0226 -0.0479124)	(1.5708 0 0)	*right_shoulder*	*right_lower_arm*
18	*right_lower_arm*	*right_elbow_pitch*	(0.155337 0.023814 0.00263175)	(-1.5708 1.5708 0)	*right_upper_arm*	
19	*neck*	*neck_yaw*	(-0.008 0 0.20875)	(-3.14159 0 3.14159)	*torso*	*head*
20	*head*	*head_pitch*	(0 -0.000244706 -0.0425)	(-1.5708 1.5708 0)		*neck*