UCLouvain

epl

École polytechnique de Louvain

Design and Control of a Bipedal Robot

Author: **Xing CHEN**
Supervisors: **Raphaël JUNGERS, François WIELANT**
Readers: **Renaud RONSSE, Paul FISETTE, Julien HENDRICKX**
Academic year 2022–2023
Master [120] in Electro-mechanical Engineering

Acknowledgments

I would like to express my sincere gratitude and appreciation to all those who have contributed to the completion of this master's thesis.

Foremost, I am deeply grateful to my supervisors, Raphaël Jungers and François Wielant, for their invaluable guidance, support, and expertise throughout this research journey. Their unwavering commitment, insightful feedback, and encouragement have been instrumental in shaping the direction of this thesis.

I would also like to extend my thanks to the faculty members of the Control Department and the Department of Applied Mathematics at Université Catholique de Louvain for their exceptional teaching and mentorship. Their dedication to academic excellence has been a constant source of inspiration.

Lastly, I want to express my heartfelt appreciation to all the individuals who may not be mentioned here but have played a part, no matter how small, in shaping my academic and personal growth.

Thank you all for your support and contributions to this thesis.

Abstract

This study investigates the design and control procedures for bipedal walking, with a specific focus on the application of Zero Moment Point (ZMP) based control. A thorough review of control strategies in bipedal robots reveals the significance of ZMP-based control as a primal method. The chosen algorithm serves as a complement to another project that focuses on the development of a new control technique called 'abstraction'.

The main objective of this research is to develop an effective ZMP-based controller that ensures stable and robust locomotion for a bipedal robot within a controlled environment. This controller will later serve as a valuable benchmark for comparison and evaluation. The research primarily focuses on the design and implementation of the controller, accompanied by the creation of a new simulation environment specifically tailored for testing and benchmarking the newly proposed controller.

Drawing upon existing literature on control methods in bipedal robots, a novel control algorithm is proposed by integrating various techniques available, including foot planner, swing foot, reference ZMP calculation, Centre of Mass (CoM) trajectory generation, inverse kinematics, and feedback control. Extensive simulations are conducted using a dynamic bipedal robot model to validate the effectiveness of the proposed controller. The simulations encompass diverse walking scenarios, such as straight walking and circular motion. The performance evaluation of the controller considers factors such as walking speed, energy efficiency, and trajectory tracking.

The simulation results demonstrate that the developed ZMP-based controller achieves stable walking at a maximal speed of 0.235 m/s for straight motion and 0.163 m/s for circular trajectory within a controlled environment and without ZMP feedback. However, the inclusion of ZMP feedback to enhance performance and position tracking reveals unanticipated control issues that remain unresolved at this stage.

This research significantly contributes to the field of bipedal robotics by offering a comprehensive analysis and implementation of a ZMP-based controller for an eight-degree-of-freedom bipedal robot. The result provides valuable insights into the design considerations and challenges associated with developing effective control strategies for bipedal locomotion.

Contents

Introduction **1**

1 State of the Art **3**
1.1 Biped Robots . . . 3
1.2 Walking pattern generator . . . 4
1.2.1 Stability based criteria . . . 5
1.2.1.1 Zero-moment point . . . 5
1.2.1.2 The 2D Inverted Pendulum model . . . 5
1.2.1.3 Cart-Table Model . . . 7
1.2.2 Bio-inspired methods . . . 8
1.2.3 Passive Walk . . . 9
1.2.4 Intuitive control . . . 10
1.2.5 Hybrid-zero dynamics . . . 11
1.3 Conclusion . . . 11

2 Modelling of the Robot **13**
2.1 Presentation of the Robot . . . 13
2.2 Notation and Robot Model . . . 13
2.3 Robot Kinematics . . . 15
2.3.1 Direct Kinematics . . . 15
2.3.2 Inverse Kinematics . . . 17
2.4 Robot Dynamics . . . 19
2.4.1 Forces and Torques . . . 19
2.4.2 Contact models . . . 20
2.4.3 Robot Dynamics Equations . . . 22
2.5 Conclusion . . . 22

3 Bipedal Walking **23**
3.1 Achieving Stable Walking . . . 23
3.2 Cart-Table Model . . . 24
3.3 Preview Control . . . 25
3.3.1 Introduction . . . 25
3.3.2 Problem Formulation for Servomechanism Design . . . 26
3.4 Preview Control for the Cart-Table Model . . . 29
3.5 Conclusion . . . 32

4 Development of a ZMP based Controller for our Robot **33**
4.1 Presentation of the controller . . . 33
4.2 Foot Planner . . . 34
4.2.1 Working Principle . . . 34
4.2.2 Implementation of the Foot Planner . . . 35
4.3 Swing Foot Trajectory . . . 36
4.3.1 Working principle . . . 36
4.3.2 Implementation of the Swing Foot Trajectory . . . 38
4.4 ZMP Trajectory Generator . . . 38

4.4.1 Working principle . . . 38
4.4.2 Implementation of the ZMP Trajectory Generator . . . 39
4.5 CoM Trajectory Generator . . . 39
4.5.1 Working Principle . . . 39
4.5.2 Implementation of the CoM Trajectory Generator . . . 41
4.6 Inverse Kinematics . . . 41
4.6.1 Working principle . . . 41
4.6.2 Implementation of the Inverse Kinematics . . . 42
4.7 Numerical results of the ZMP based controller . . . 43
4.8 Conclusion . . . 44

5 Simulation using `RigidBodyDynamics.jl` 47
5.1 Introduction . . . 47
5.2 Robot Model . . . 47
5.2.1 The URDF file . . . 48
5.2.2 Contact Model . . . 49
5.3 Simulation environment . . . 50
5.3.1 Position Control . . . 50
5.3.2 Estimation of the ZMP . . . 52
5.4 Optimisation of the ZMP Based Controller . . . 54
5.4.1 Problem Formulation . . . 54
5.4.2 Constraints formulation . . . 55
5.4.2.1 Constraints . . . 55
5.4.2.2 Non-Independent Input Parameters . . . 57
5.4.3 Optimisation for a discrete input-space . . . 57
5.5 Results . . . 58
5.5.1 Optimisation result . . . 58
5.5.2 Testing the Optimised Controller . . . 60
5.6 Conclusion . . . 61

6 Closed-loop form of our ZMP based controller 65
6.1 Introduction . . . 65
6.2 Closed loop system . . . 65
6.2.1 Dynamics Filter based on Preview Control . . . 66
6.2.2 Integration in the simulation environment . . . 66
6.3 Futures perspectives . . . 67
6.4 Conclusion . . . 68

7 Conclusion 69

Acronyms

CoG Centre of Gravity

CoM Centre of Mass

LQI Linear Quadratic Integral

ZMP Zero Moment Point

List of Figures

1.1 WABOT-1 (1973) [12] . . . 3
1.2 Honda ASIMO (2000) [3] . . . 3
1.3 2D AMBER Robot (2014) [1] . . . 4
1.4 AMBER2 Robot (2014) [25] . . . 4
1.5 2D inverted pendulum . . . 7
1.6 2D linear inverted pendulum . . . 7
1.7 Cart-table model . . . 8
1.8 Tiles of the walking gait for a human subject . . . 9
1.9 The mean human output data computed over one step . . . 9
1.10 Bipedal Toy from McMahon . . . 10
1.11 4-link passive robot . . . 10

2.1 Experimental setup of the robot . . . 14
2.2 Kinematic diagram . . . 14
2.3 Robot joints definitions . . . 15
2.4 Robot torques and ground reaction forces/torques . . . 16
2.5 Robot local frames according to Denavit-Hartenberg convention . . . 17
2.6 Basic Concept Behind Numerical Approach to Inverse Kinematics . . . 18
2.7 Analytic solution to the Inverse Kinematics problem . . . 20
2.8 Compliant Contact Model . . . 21

3.1 Definition of the ZMP . . . 23
3.2 Support Polygon . . . 24
3.3 Servo-controllers to track the target ZMP . . . 25
3.4 ZMP and CoM trajectory for one step . . . 26
3.5 Block diagram of the Preview controller . . . 28
3.6 Reference ZMP trajectory . . . 31
3.7 Preview Gains for a preview time of 2 seconds . . . 32
3.8 CoM and ZMP trajectories from the closed-loop system of the cart-table model . . . 32

4.1 Our version of the ZMP based controller basic diagram . . . 34
4.2 Foot Planner implementation . . . 36
4.3 Foot Planner Flowchart: Foot Planner and Next Reference Point . . . 37
4.4 Swing foot trajectory in time and spacial domain . . . 38
4.5 Flowchart: ZMP Trajectory and Swing Foot Trajectory . . . 40
4.6 Difference between hip height z_h and CoM height z_c . . . 41
4.7 CoM Trajectory Flowchart: 3D and 2D CoM Trajectory Generator . . . 42
4.8 Inverse Kinematics Flowchart: Inverse Kinematics and Global to Local . . . 43
4.9 Result from the ZMP based controller for a straight path . . . 45
4.10 Result from the ZMP based controller for a quadrant path . . . 45

5.1 Real Biped robot and simulated model . . . 49
5.2 URDF model of the experimental setup . . . 50
5.3 Distribution of the contact point on each foot . . . 51
5.4 Definition of each variable for the ZMP measurement . . . 52

5.5 Schematic of the step length $L_{\max}$ constraint 56
5.6 Flowchart: Optimisation algorithm 58
5.7 Optimisation result 59
5.8 Optimisation result where objectives value are normalised and sorted by $f_1(x)$ 61
5.9 Simulation result for a straight path 62
5.10 Simulation results for circular motion 63

6.1 Closed loop form of our ZMP based controller 65
6.2 Dynamic filter based on Preview Control 66
6.3 Dynamic filter based on Preview Control 67

List of Tables

2.1 Denavit-Hartenberg parameters for the biped robot 18

5.1 Parameters of each body in the biped robot 48
5.2 Inertia of each body in the biped robot. 48
5.3 Parameters of each body in the boom 48
5.4 Description of the generalised coordinate for the real setup 50
5.5 Description of the URDF file 51
5.6 Maximal characteristic for each Dynamixel at an input voltage of 12 V 56
5.7 Discritised input space for the Optimisation Algorithm 58
5.8 Performance achieved of the ZMP based controller in open-loop 60

7.1 Performance achieved of by the chosen ZMP based controller in open-loop. 69

Introduction

Motivations

This thesis is an integral part of a broader project that seeks to tackle diverse theoretical control problems through the utilisation of an approach known as 'abstraction'. The primary objective is to validate this approach by applying it to a range of control problems, including the control of bipedal robots and self-driving cars, among others. The current work serves as a foundational component, offering an initial understanding of the performance of a well-established controller on an experimental biped robot. Moreover, it establishes a valuable benchmark for evaluating and comparing the efficacy of the newly proposed technique within the context of the broader project.

Bipedal robots have long been a subject of fascination and research, as they aim to replicate the complex locomotion capabilities of humans. Achieving stable and natural walking motion is a fundamental challenge in robotics. In recent years, numerous complex control algorithms have been developed, but in this thesis, we focus on studying a basic four-degree-of-freedom bipedal robot. The simplicity of the robot calls for a simple control strategy to avoid unnecessary complexity. Therefore, we investigate the use of a traditional control technique for bipedal walking known as Zero Moment Point (ZMP) based control.

The objective of this thesis is to redesign the robot, if necessary, and implement a ZMP-based controller that can achieve stable and dynamic locomotion in a controlled environment. The newly designed controller will utilise the concept of ZMP to generate walking trajectories and employ feedback control to ensure stability and robustness. To validate the effectiveness of the controller, we will test it in various scenarios, including straight and circular motions, using a simulation environment developed from scratch.

In summary, this thesis aims to explore and harness the power of ZMP-based control in the context of bipedal locomotion. By developing an innovative controller and conducting comprehensive simulations, this research endeavours to contribute to the field of robotics and inspire further advancements in the design and control of bipedal robots.

Thesis organisation

This thesis is about 6 chapters if we exclude the chapter 7 which is about the general conclusion of this work.

Chapter 1 (State of the Art) provides a comprehensive review of the existing literature on different control strategies and their applications in bipedal walking. This state-of-the-art analysis offers valuable insights into the advantages and limitations of various control strategies, guiding the development of a suitable controller for our bipedal robot.

Chapter 2 (Modelling of the Robot) presents a concise description of the studied robot, including the notation and assumptions used throughout the thesis. The chapter also discusses the kinematic and dynamic models of the robot.

Chapter 3 (Bipedal walking) introduces the ZMP concept and the ZMP stability criterion. It

also presents a linearised and simplified model of the robot using a cart-table model. This chapter establishes the foundation of stable walking for bipedal beings, which is crucial for the subsequent study.

Chapter 4 (Development of a ZMP based controller for our Robot) explores the insights gained from the literature review, with a focus on ZMP-based controllers. A novel open-loop control algorithm is proposed, integrating a foot planner, a swing foot trajectory, a reference ZMP calculation, a Centre of Mass (CoM) trajectory generation, and an inverse kinematics process. The controller aims to generate smooth and natural walking patterns while adhering to the ZMP criterion.

Chapter 5 (Simulation using `RigidBodyDynamics.jl`) aims to validate the controller's design through extensive simulations using a Unified Robot Description Format (URDF) as a dynamic bipedal robot model. The performance evaluation of the ZMP-based controller considers key metrics such as walking speed, energy efficiency, and mechanical energy conversion. The selection of the controller will be based on optimising these metrics.

Chapter 6 (Closed-loop form of our ZMP based controller) introduces the concept of ZMP feedback through a novel closed-loop system called the "Dynamic Filter." This control structure ensures stability by dynamically adjusting the robot's posture and gait parameters to adapt to varying walking conditions. The chapter also discusses the attempted implementation of this structure in our simulation environment and provides insights into future perspectives for the controller.

State of the Art

1

1.1 Biped Robots

Bipedal robots, also known as humanoid robots, are advanced machines designed to walk on two legs and imitate human movement and gait. These robots find applications in various fields, including search and rescue, entertainment, and assistive robotics. The development of bipedal robots has garnered significant attention from scientists, engineers, and the general public, leading to extensive research and development efforts over the years.

In the 1970s, the first humanoid robot, WABOT-1 (Figure 1.1), was created [12]. This robot demonstrated remarkable capabilities such as object recognition, language understanding and speech, object manipulation, and bipedal walking. Honda's development of humanoid robots in the 1980s, including the Px series robot, paved the way for the renowned ASIMO robot [3] (Figure 1.2). More recently, simpler and more affordable robots like the NAO robot by Aldebaran Robotic have been developed [19]. Additionally, research robots such as AMBER (Figure 1.3) and AMBER2 (Figure 1.4), which are simple planar 2D robots, have contributed to the field [1, 25].

One of the primary challenges in developing biped robots is maintaining stability and balance during walking, especially on uneven or unfamiliar terrain. Replicating human behaviour on a simpler, rigid mechanical structure presents difficulties due to the complexity of the human body. Researchers have explored various control strategies and mechanical designs to tackle this challenge and achieve stable and balanced walking.

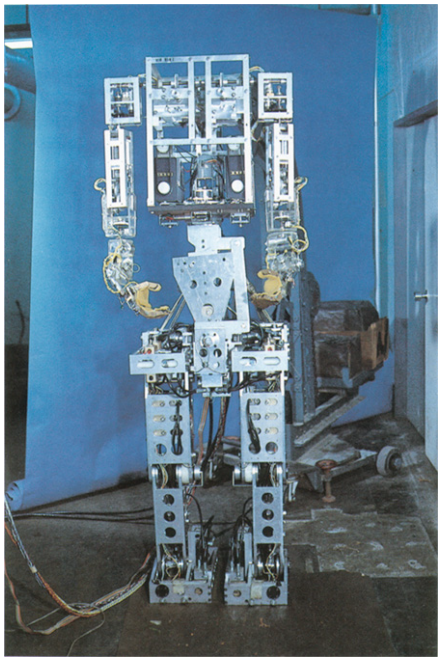


Figure 1.1: WABOT-1 (1973) [12]

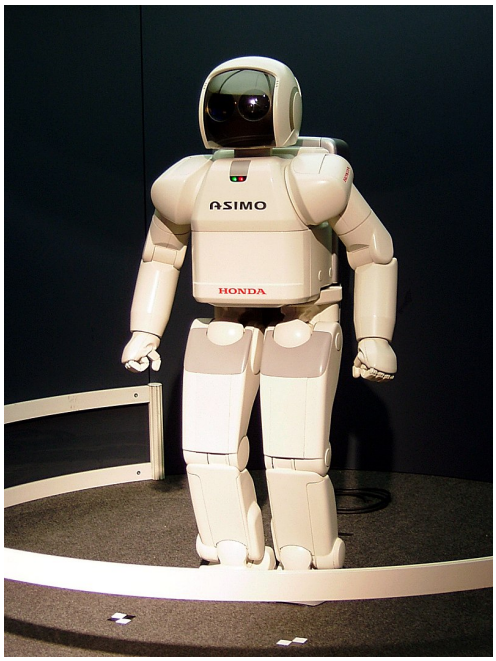


Figure 1.2: Honda ASIMO (2000) [3]

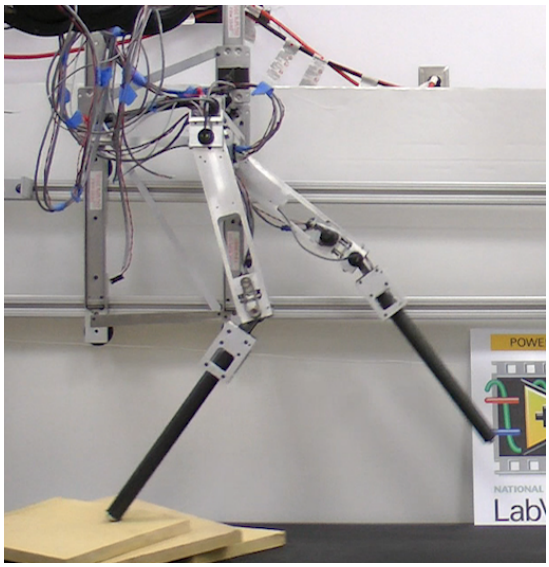


Figure 1.3: 2D AMBER Robot (2014) [1]

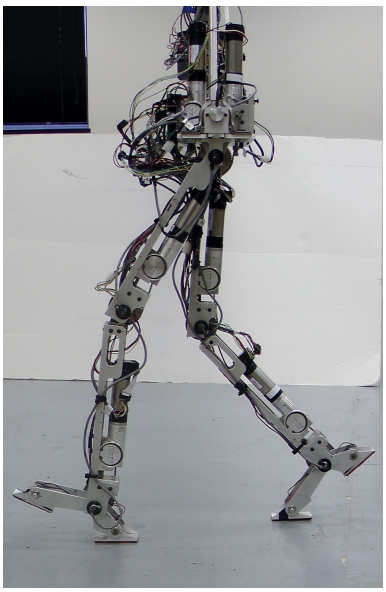


Figure 1.4: AMBER2 Robot (2014) [25]

1.2 Walking pattern generator

Since the 1970s, researchers and engineers have developed various control schemes to replicate human walking behaviour, leading to the emergence of walking pattern generators for biped robots. Walking pattern generators play a crucial role in bipedal robots by providing the necessary control signals to generate stable and efficient walking gaits. These generators utilise mathematical models and control algorithms to generate desired joint trajectories and torques required for walking motions.

Achieving stability and balance in the robot's gait is one of the primary challenges in developing walking pattern generators, especially for bipedal robots that need to maintain balance on two legs while walking on even or uneven terrain.

Another important area of research in walking pattern generators is the development of adaptable and responsive control strategies. This involves designing generators that can adapt to changes in the environment or unexpected disturbances such as slips or impacts. Various algorithms have been explored to generate stable walking gaits that address these challenges, including:

- Stability based criteria ;
- Bio-inspired methods ;
- Passive walk ;
- Intuitive control ;
- Hybrid-zero dynamics ;

Each of these approaches offers unique advantages and trade-offs in terms of stability, energy efficiency, and robustness. Researchers continue to explore and refine these algorithms to improve the performance of walking pattern generators and enhance the capabilities of biped robots in terms of locomotion.

1.2.1 Stability based criteria

1.2.1.1 Zero-moment point

Definition

Stability and balance are fundamental aspects of humanoid robots. In the 1970s, Vukobratovic *et al.* [24] introduced the concept of the Zero-Moment Point (Zero Moment Point (ZMP)), which is a famous stability criterion in robotics.

Vukobratovic defined the ZMP as the point on the ground where the net moment of inertial forces and gravity forces has no components along the horizontal axes.

According to this definition, the stability criteria define a point p where $\tau_x = 0$ and $\tau_y = 0$, with τ_x and τ_y representing the moments around the x-axis and y-axis generated by all reaction forces/torques. Therefore, the point p is referred to as the zero-moment point (or ZMP), and when the ZMP exists within the support polygon, the contact between the ground and the support leg is stable.

Historical Views

The concept of ZMP was first introduced in 1972 by Vukobratovic *et al.* [24] to achieve stable walking for humanoid robots. Since then, researchers have developed numerous algorithms to maintain the ZMP within the base support surface of the robot by measuring the ZMP through force/torque sensors and employing feedback loops to adjust the ZMP in real-time.

These control algorithms have been incorporated and implemented in the design and control of various bipedal robots, including ASIMO, ARCHIE, NAO, and more. The popularity of this algorithm is mainly due to the simplicity of the control scheme, which finds application in a wide range of scenarios.

However, the concept of ZMP has its limitations, primarily the need for accurate measurement of the centre of mass or ZMP to ensure a correct feedback loop and reject disturbances. Furthermore, the ZMP is defined only when the foot is in contact with the ground, limiting its application when the robot is running.

1.2.1.2 The 2D Inverted Pendulum model

Definition

The 2D Inverted Pendulum model represents a bipedal robot where the Centre of Mass (CoM) is located beyond the pivot point, mimicking an inverted pendulum. This model is commonly employed to describe the dynamics of a bipedal robot during the single support phase, with the fixed pivot representing the foot placement point and the mass representing the centre of mass of the robot's leg.

To simplify the analysis of a robot with multiple parts, each with its own mass, inertia, and length, Kajita *et al.* [11] introduced a more straightforward yet powerful model. The fundamental idea behind the 2D Inverted Pendulum Model is based on three assumptions:

1. The entire mass of the robot is concentrated at a single point known as the CoM;
2. The robot's legs are considered massless, and their end-effectors make contact with the ground at a single point;
3. The robot's motion is limited to the forward/backward direction in the 2D plane, neglecting lateral motion.

These assumptions allow us to represent the robot as a 2D inverted pendulum, as depicted in Figure 1.5.

Dynamics of the 2D Linear Inverted Pendulum model

The inputs of the 2D Inverted Pendulum model (2D-IPM) are the torque τ and the kick force f. It is important to note that the torque τ needs to be limited due to the small size of the bipedal foot. The dynamics of the pendulum are described using Lagrange's method, resulting in the following equations:

$$\begin{aligned} r^2\ddot{\theta} + 2r\dot{r}\dot{\theta} - gr\sin\theta &= \frac{\tau}{M} \\ \ddot{r} - r\dot{\theta}^2 + g\cos\theta &= \frac{f}{M} \end{aligned} \tag{1.1}$$

where r and θ are the polar coordinates of the CoM, g is the acceleration due to gravity, and M is the mass of the robot. Although this system is difficult to solve analytically, we can simplify these equations.

Considering hypothesis 2, we set τ to zero. This is necessary to ensure stability, as the ZMP should remain within the support polygon (which is reduced to a point in this case). The only way to achieve the ZMP at the contact point with the ground is by imposing $\tau = 0$. By solving these equations, we obtain various falling patterns, but the most interesting case arises when $f = \frac{Mg}{\cos\theta}$. In this scenario, the motion of the CoM is constrained to horizontal movement, where the effect of gravity is cancelled by the kick force f, allowing the CoM to maintain a constant height z_c.

Now, the system is simply described by (as shown in fig. 1.6) :

$$M\ddot{x} = f\sin\theta = \frac{Mg}{\cos\theta}\sin\theta = Mg\frac{x}{z_c} \tag{1.2}$$

This equation represents what is known as the 2D Linear Inverted Pendulum Model (2D-LIPM) and can be solved analytically. The solutions to these ordinary differential equations are given by:

$$x(t) = x(0)\cosh(\frac{t}{T_c}) + T_c\dot{x}(0)\sinh(\frac{t}{T_c}) \tag{1.3}$$

$$\dot{x}(t) = \frac{x(0)}{T_c}\sinh(\frac{t}{T_c}) + \dot{x}(0)\cosh(\frac{t}{T_c}) \tag{1.4}$$

Here, $x(0)$ and $\dot{x}(0)$ represent the initial position and velocity of the CoM, and $T_c = \sqrt{\frac{z_c}{g}}$ is the time constant of the system. By appropriately selecting the initial conditions, these equations provide a possible trajectory of the robot's centre of mass during one step. To extend this model to a 3D robot, Equations 1.2, 1.3, and 1.4 can be extended with a new axis y, resulting in three additional equations.

Once the trajectory of the centre of mass is computed, the joint angles can be determined using a swing foot trajectory and inverse kinematics.

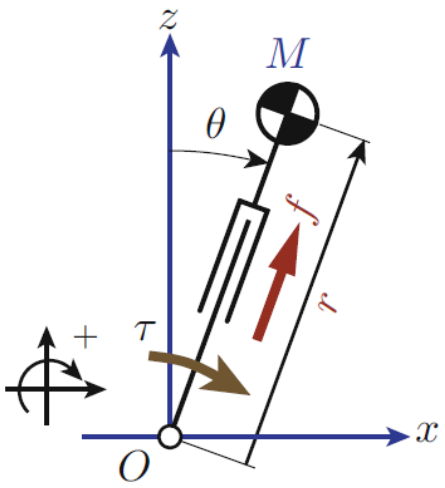


Figure 1.5: 2D inverted pendulum [12]

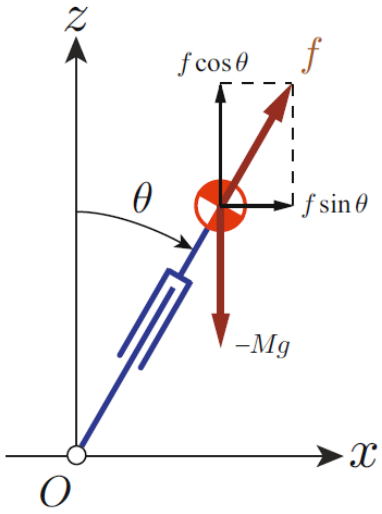


Figure 1.6: 2D linear inverted pendulum [12]

1.2.1.3 Cart-Table Model

The Cart-Table model, proposed by Kajita *et al.* [10, 12], is a simplified representation of a bipedal robot's dynamics. It approximates the complex dynamics of the robot by modelling it as a cart with mass M that moves on a frictionless table along the x and y axes, as depicted in Figure 1.7. This model has been widely used by researchers in the field of bipedal robotics [14, 23, 7].

The position of the cart, or the centre of mass of the robot, determines the centre of pressure (or ZMP) acting on the floor. The ZMP can be computed as a function of the cart's state using the following equation:

$$p_x = x - \frac{z_c}{g}\ddot{x} \tag{1.5}$$

It is interesting to note that Equation 1.5 bears resemblance to Equation 1.2 if we consider the base of the linear inverted pendulum to be located at $x_0 = p$:

$$\ddot{x} = \frac{g}{z_c}(x - p_x) \tag{1.6}$$

Thus, the cart-table model is equivalent to the linear inverted pendulum model when we treat the point p as the ZMP point rather than a foot placement.

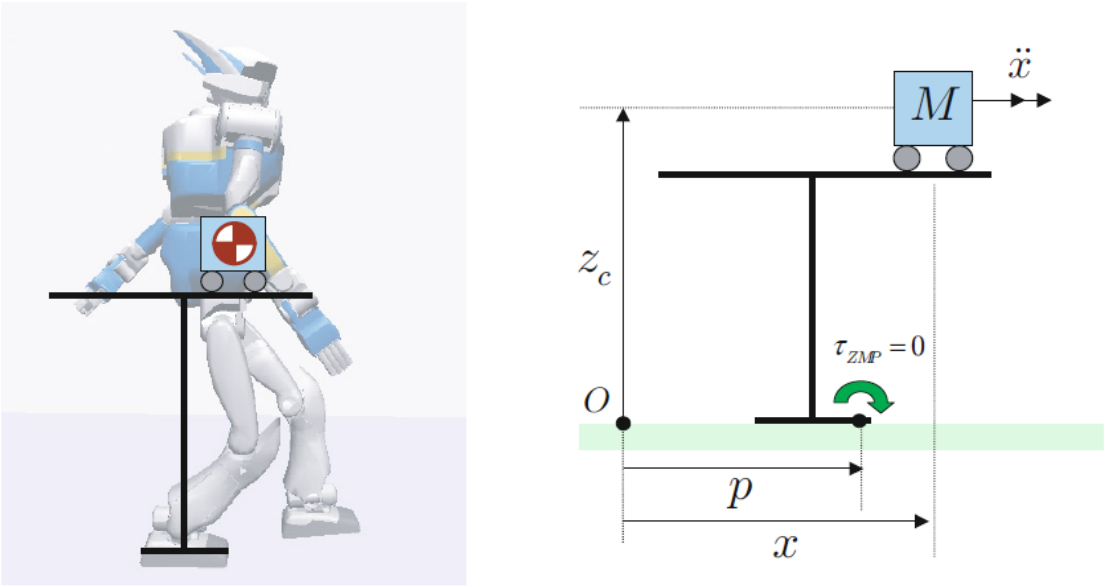
The main difference between Equation 1.5 and Equation 1.6 is that in the cart-table model, we generate the ZMP based on the motion of the centre of mass, whereas in the 2D linear inverted pendulum model, we generate the centre of the mass trajectory based on the ZMP. This approach is more convenient as it allows us to specify a target ZMP trajectory and the cart-table model will calculate an appropriate centre of mass motion to achieve that trajectory.

Once the centre of the mass trajectory is generated, the joint angles can be computed using an appropriate swing foot trajectory and an inverse kinematics model of the biped robot.

For a more physical interpretation, Figure 1.7b demonstrates that the table's foot is too small to support the cart on the edge. However, by selecting an appropriate cart acceleration, the table can be stabilised. The definition of the ZMP implies that:

$$\tau_{x,\,\mathrm{ZMP}} = Mg(x - p_x) - M\ddot{x}z_c = 0 \tag{1.7}$$

$$\tau_{y,\,\mathrm{ZMP}} = Mg(y - p_y) - M\ddot{y}z_c = 0 \tag{1.8}$$



(a) A real robot and a cart-table in the sagittal plane **(b) Detailed modelling of the cart-table model**

Figure 1.7: Cart-table model [12]

If p_x and p_y remain within the support polygon during walking, the motion is considered stable.

1.2.2 Bio-inspired methods

Bio-inspired methods for bipedal walking involve the utilisation of approaches and techniques inspired by biological organisms, particularly humans and animals, to develop robotic systems capable of walking on two legs. These methods aim to gain a deeper understanding of and replicate complex mechanisms, control strategies, and biological behaviours that are highly inspired by humans or animals.

Furthermore, bio-inspired methods draw upon principles from biomechanics, neuroscience, and control theory to create robotic systems that can exhibit human-like walking patterns. By studying and mimicking the structure and function of biological systems, researchers strive to enhance the efficiency, stability, and adaptability of bipedal robots. For instance, Ames [1] developed a **human-inspired control** approach, which was implemented in a 2D robot named AMBER, to achieve bipedal walking. Another notable example is Zhao *et al.* [25], who developed a different type of human-inspired controller and demonstrated its effectiveness using the robot AMBER2.

Let's briefly explore how these researchers developed their human-inspired control approaches. Ames views the walking system as a 'black box', where specific walking behaviours serve as inputs, and the output reflects the desired human-like walking behaviour. To derive a suitable human-inspired controller, Ames proposes designing a controller that takes signals generated by the 'black box,' reflecting human walking behaviour, as inputs. The resulting output is then applied to the robot to exhibit human-like walking behaviour.

Upon closer examination of human walking data in his paper [1], Ames identifies two types of functions that characterise human walking:

- Linear functions of time, which describe the forward position of the hip.
- Solutions to spring-damper systems, which describe the remaining outputs. These functions are commonly referred to as canonical walking functions.

Figure 1.8 illustrates an example of human walking data. Using this data, Ames derives a low-dimensional representation of human walking, referred to as *human output*, from the experimental data for the AMBER robot with five degrees of freedom. Figure 1.9 presents the mean human output computed over one step (heel strike to heel strike) from data collected during walking experiments involving nine subjects. The figure also includes error bands indicating one standard deviation from the mean, along with fits of the canonical walking functions (shown in red). The data for δp_{hip} was shifted to be zero at time zero.

The walking behaviour is estimated using the following equations:

$$y_1(t, v_{\text{hip}}) = v_{\text{hip}} t^H \qquad y_2(t, \alpha) = e^{-\alpha_4} \left(\alpha_1 \cos(\alpha_2 t^H) + \alpha_3 \sin(\alpha_2 t^H) \right) + \alpha_5 \tag{1.9}$$

Here, v_{hip} and α are parameters determined through fitting the human output data. Once these functions are evaluated, Ames proposes a human-inspired controller that takes these functions as inputs.

The human-inspired controller utilised these functions as inputs to generate control signals for the bipedal robot, allowing it to exhibit human-like walking patterns. By leveraging the insights from human walking behaviour, this bio-inspired approach aimed to improve the realism and naturalness of bipedal robot locomotion.

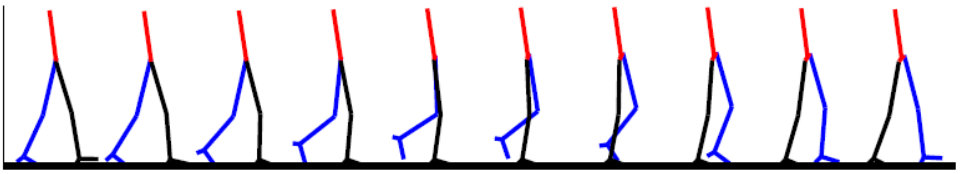
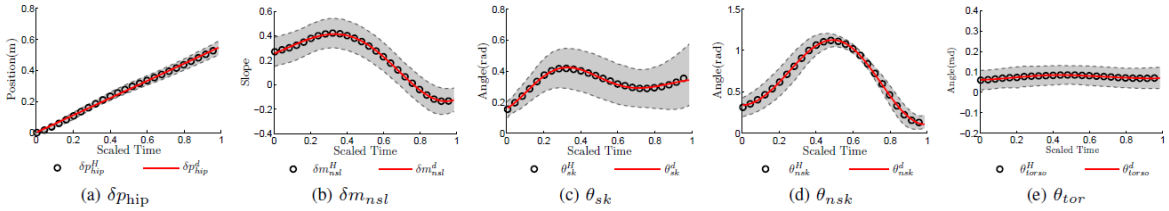


Figure 1.8: Tiles of the walking gait for a human subject, plotted directly from the human walking data from [1]



(a) δp_{hip} (b) δm_{nsl} (c) θ_{sk} (d) θ_{nsk} (e) θ_{tor}

Figure 1.9: The mean human output data computed over one step (heel strike to heel strike) from the data of nine subjects walking, with error bands showing one standard deviation from the mean, and the canonical walking function fits (red). The data for the δp_{hip} was shifted to be zero at time zero. from [1]

1.2.3 Passive Walk

The concept of passive dynamic walking was first explored in the early 1990s by McGeer in [17]. This research was inspired by bipedal toys, as demonstrated by McMahon in [18], such as the one depicted in Figure 1.10. These toys can walk downhill using only their potential energy and a clever balancing mechanism. The energy required to compensate for friction losses within the internal mechanism is achieved by lowering the toy's height as it walks downhill.

McGeer demonstrated in [17] that a 4-link mechanism, as shown in Figure 1.11, can achieve self-powered walking on an inclined surface without the need for actuators. When the inclination is

properly chosen within a certain range, the system exhibits stable cyclic walking. In this case, the energy lost by the biped is equal to the work done by gravity.

By carefully designing the mass and length of the passive walker, it is possible to achieve a smooth and natural walking motion with high efficiency.

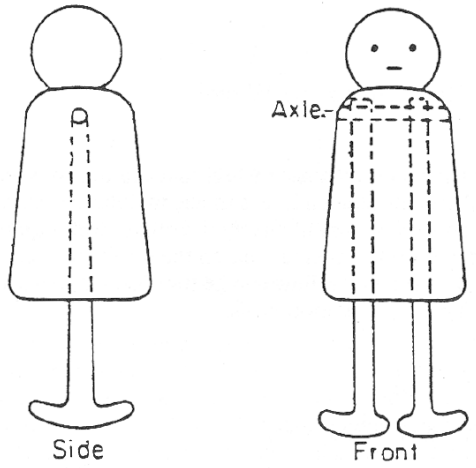


Figure 1.10: Bipedal Toy from McMahon [18]

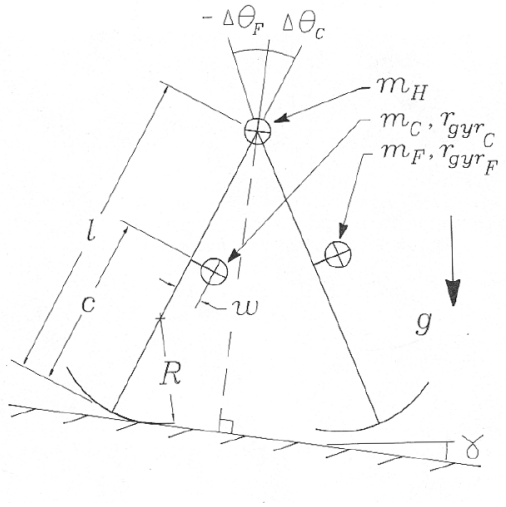


Figure 1.11: 4-link passive robot [17]

1.2.4 Intuitive control

In many cases, analysing bipedal robots mathematically can be challenging. However, successful control strategies can be developed using simple physical intuition and described in straightforward terms. Intuitive control refers to a type of controller that is based on human intuition about the system and an understanding of its behaviour. In essence, an intuitive controller acts as a coach, providing instructions to the robot. An analogy can be made to a PD controller, which can be considered a type of intuitive controller.

Pratt *et al.* proposed in 1997 [21] that a simple 2D bipedal robot will walk only if it achieves the following five objectives:

1. Stabilise the robot's height.
2. Stabilise the robot's pitch.
3. Stabilise the robot's speed.
4. Move the swing leg in a way that allows for stability of height, pitch, and speed.
5. Perform transitions from one supporting leg to the other at the appropriate time.

Pratt's idea was to develop a control strategy that ensures the fulfilment of these objectives, thereby enabling the robot to walk. This approach involves using an appropriate intuitive control strategy that addresses each objective individually. For example, three distinct intuitive controllers could be utilised for height stabilisation, pitch stabilisation, and speed stabilisation. The strategy might include the following:

1. Height Stabilisation:
    - Maintain a constant height above the ground.

- Keep the stance leg length constant.

2. Pitch Stabilisation:
   - Maintain a desired pitch level.
   - Follow a predetermined pitch trajectory.
3. Speed Stabilisation:
   - Adjust the location of the body's centre of mass (CoM) by pitching forward or backward.
   - Change transition events based on the desired speed.

These are just a few examples of intuitive control strategies. There are numerous possibilities, but defining an effective control strategy can be challenging and requires a deep understanding of the system being studied.

1.2.5 Hybrid-zero dynamics

In a hybrid-zero dynamic systems, two main components are involved: the hybrid mechanism and zero dynamics.

A hybrid system is a type of system that consists of different subsystems, and based on certain triggers or conditions, the system transitions from one subsystem to another. For example, a bipedal robot is a hybrid system because it exhibits both a swing phase and an impact phase.

Inverse Dynamics is a well-known approach for controlling biped robots. It involves defining a set of outputs (e.g., torques) that correspond to the desired inputs and designing a control law that drives these outputs to zero. The concept of zero dynamics assumes that the robot's task is encoded in the set of outputs in such a way that nullifying these outputs is equivalent to achieving the desired task. The computed torque method, which asymptotically drives the outputs to zero, indirectly designs the zero dynamics for the robot.

In a hybrid-zero dynamic systems, these two concepts are combined to generate joint trajectory angles. This approach asymptotically constrains the dynamics of the closed-loop hybrid model to a lower-dimensional, attractive, and invariant subset of the state space. As a result, it produces stable periodic solutions of the dynamics within this subset, which encode the desired task.

1.3 Conclusion

In this chapter, we have explored various walking pattern generators for bipedal robots. The walking pattern generator plays a crucial role in generating stable and efficient walking gaits.

We have discovered that stability-based criteria, particularly the ZMP criteria, are derived from a fundamental understanding of robot models. By modelling the robot as a Linear Inverted Pendulum Model, it is possible to generate walking patterns using certain assumptions.

Bio-inspired methods are widely used to generate human-like motion for humanoid robots. By studying human data and walking behaviour, it is possible to generate appropriate human-like walking patterns.

Passive walkers are intriguing because they utilise the dynamics of the robot's mechanism to walk passively or with minimal energy. This type of control is highly efficient, although it is not very mobile,

limiting its range of applications.

Intuitive control provides a simple approach to controlling bipedal robots. It relies on the designer's physical intuition and requires a deep understanding of the robot's mechanisms.

Hybrid-zero dynamics offer a more mathematical approach to controlling bipedal robots. This method seeks a stable periodic solution where the zero dynamics come into play.

Recalling the initial goal of this thesis, it is to provide a preliminary assessment of the performance of the chosen controller for our biped robot, which will serve as a benchmark for the newly developed technique 'abstraction'. As a result, our priorities will revolve around ensuring the reproducibility of the algorithm and selecting a technique that is both simple and widely adopted within the field. By prioritizing these aspects, we aim to contribute to the evaluation and the development of the method.

Therefore, we will focus exclusively on stability-based walking patterns because they are simple, efficient, fast, and easy to replicate. The cart-table model shows promise, as it enables online pattern generation, which is highly convenient for future applications.

Modelling of the Robot

2

A robot model is a mathematical representation of the behaviour, capabilities, and limitations of an actual physical robot. Having a clear and precise model is crucial to redesign, control, and understanding of the walking process, among other purposes.

The underlying philosophy of this project is to construct a simple bipedal robot model with four degrees of freedom capable of walking. Developing a simple prototype robot enables us to test the hardware and make any necessary redesign.

Firstly, in Section 2.1, we will introduce the robot under study. Then, Section 2.2 will present the notations and the planar model employed. Finally, Sections 2.3 and 2.4 will delve into the mathematical representation of the robot's kinematics and dynamics, respectively.

2.1 Presentation of the Robot

The bipedal robot being studied is an anthropomorphic robot designed for analysing walking behaviour. In this work, the robot's original design has been slightly modified to accommodate a ZMP-based walking gait generators, requiring a flat foot for the robot.

In this new design, the robot is considered planar, enabling it to walk in the sagittal plane. The hip is connected to the reference frame through a 3-axis boom, allowing for a circular trajectory with a radius defined by the length of the boom link. It can be assumed that the radius is sufficiently large to simplify the walking gait into planar motion at any given instant t.

The experimental setup is illustrated in Figure 2.1, which shows the bipedal robot and its connection to the reference frame. The kinematic diagram is also depicted in Figure 2.2. The robot consists of seven bodies: a hip connected via the boom, right and left thighs, right and left legs, and right and left feet. The articulations between these bodies are revolute and perpendicular to the planar motion. Only four actuators control the right and left hip joints and the right and left knees. Therefore, the hip orientation and ankles are not actuated. For the sake of simplicity in the study, we will set the hip orientation to zero, effectively fixing the y-revolute joint from the boom to the robot.

During the single support phase, we have three degrees of freedom from the position of the swinging foot and two degrees of freedom from the body position. Thus, the system is under-actuated. In the double support phase, the system is no longer under-actuated if each foot is in contact with the ground. An under-actuated system implies that the system cannot follow any arbitrary trajectory in configuration space. In simpler terms, the robot has more degrees of freedom than actuators.

2.2 Notation and Robot Model

The notation used for each joint in the bipedal robot is shown in Figure 2.3. The index starts at 2 because we consider the 2 degrees of freedom from the boom, since we have block one degree of freedom. The boom's joint, q_1 and q_2, define the position $P_h = \begin{bmatrix} x_h & z_h \end{bmatrix}^T$ of the hip in the plane. Note that α is set to a constant value as the hip orientation is fixed. The relative angles q_i are

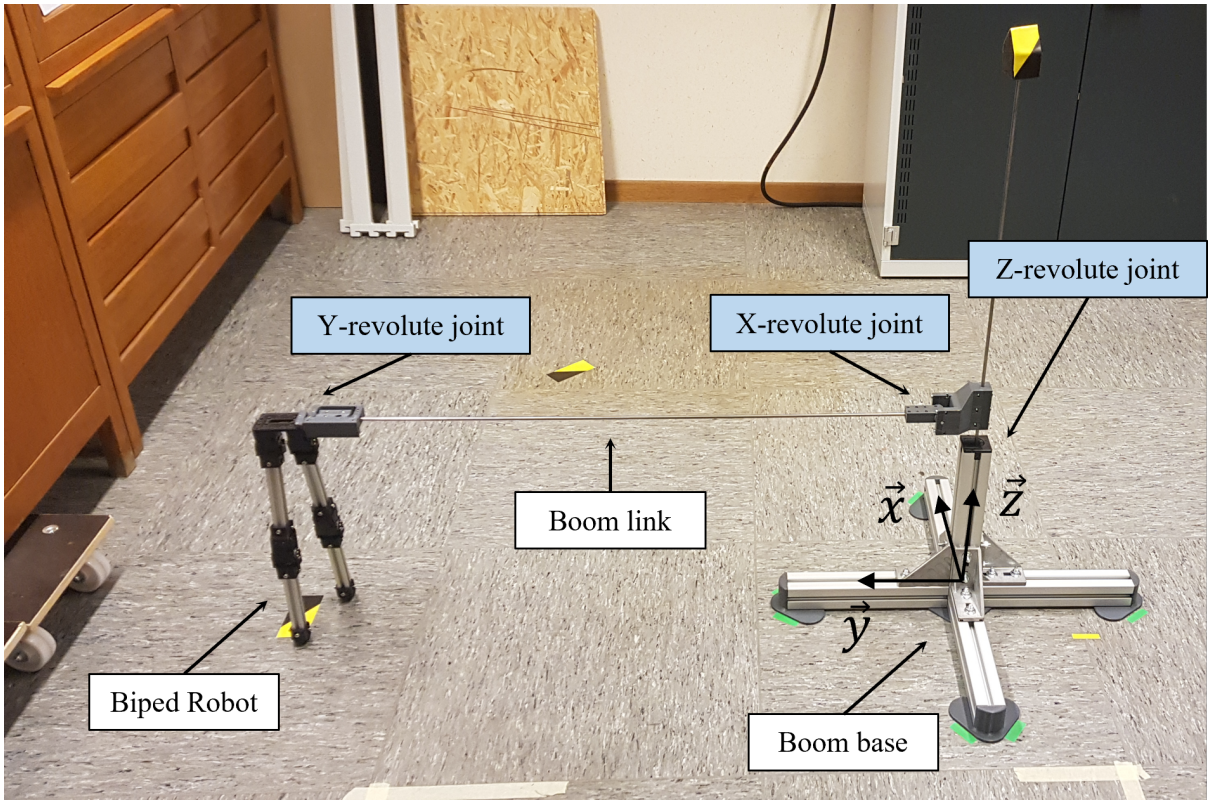


Figure 2.1: Experimental setup of the robot. This figure depicts the unconstrained y-revolute joint, which is not the case in this work.

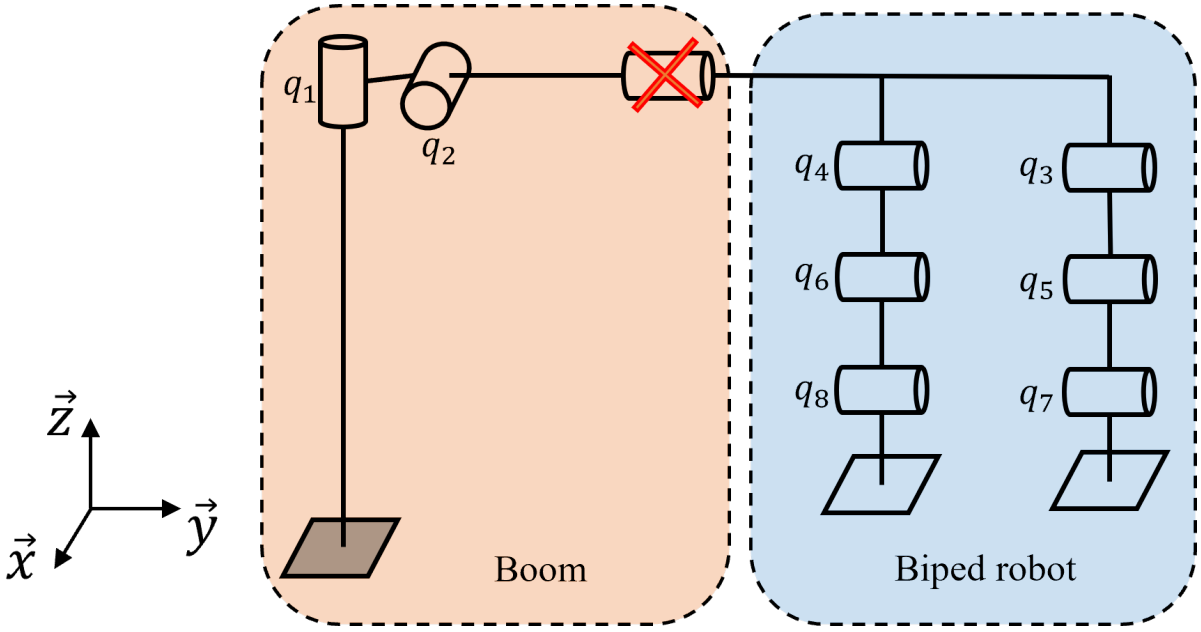


Figure 2.2: Kinematic diagram. This kinematic diagram is derived from the real robot, but the y-revolute joint for the boom is omitted here, as it compensates for the torque produced by each actuator

.

positive in the counterclockwise direction and are defined as the angles between body i and its successor.

We can represent the joint configuration vector as $q = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 \end{bmatrix}^T$, which defines the configuration of the robot. Let $q_u = \begin{bmatrix} q_3 & q_4 & q_5 & q_6 \end{bmatrix}^T$ denote the vector containing the positions of each actuated joint. The position of the left ankle is denoted as $P_L = \begin{bmatrix} x_L & z_L \end{bmatrix}^T$, and the position of the right ankle is denoted as $P_R = \begin{bmatrix} x_R & z_R \end{bmatrix}^T$.

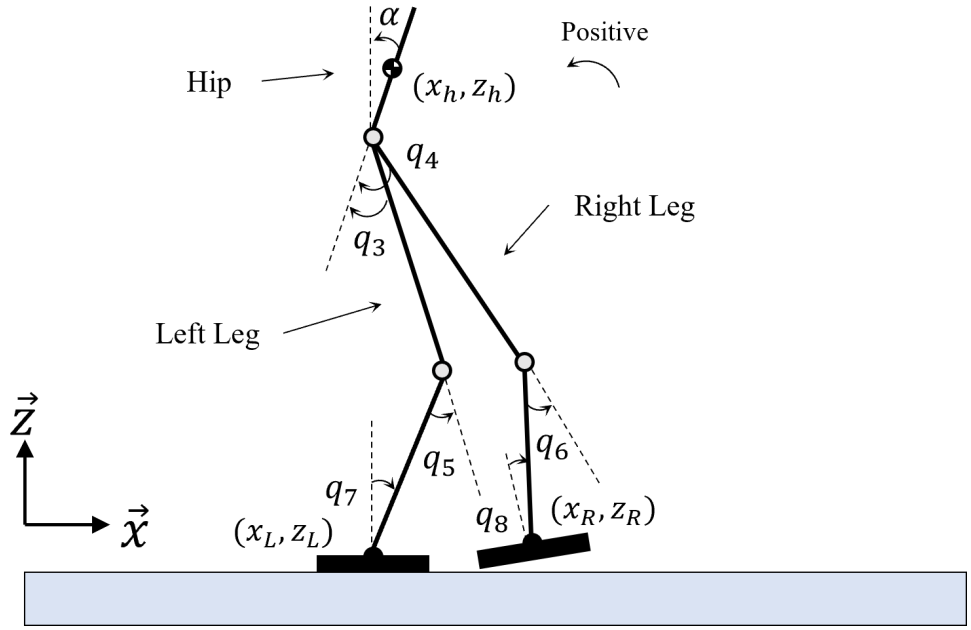


Figure 2.3: Robot joint definitions. Here, (x_h, z_h) represent the location of the CoM of the hip body with respect to he fixed frame. (x_L, z_L) and (x_R, z_R) refer to the location of the centre of the right/left foot sole.

Figure 2.4 illustrates the notation used to represent the joint torque provided by the actuators and the ground reaction forces/torque. τ_3 (τ_4) represents the torque applied from the hip to the left (right) thigh, while τ_5 (τ_6) represents the torque applied from the left (right) thigh to the left (right) leg. The torque are considered positive if they allow a counterclockwise motion of the successor link.

Let $\vec{R}_L$ and $\vec{R}_R$ represent the resultant ground reaction forces on each foot with respect to the reference frame. We also consider τ_L and τ_R as the ground reaction torque, as the contact of the foot with the ground may not be even.

2.3 Robot Kinematics

2.3.1 Direct Kinematics

Direct kinematics is a fundamental concept in robotics that involves determining the position and orientation of a robot's end-effector based on the configuration of its joints. In our case, we need to find the positions P_L and P_R as functions of q_u. Various techniques exist for solving the direct kinematics problem, including analytic solutions, numerical methods, and closed-form solutions. In this section, we will focus only on closed-form solutions, specifically the use of the Denavit-Hartenberg (DH) convention, which provides a concise mathematical representation of the direct kinematics solution for a given robot.

We define the homogeneous transformations from the global coordinate (GC) to the foot (f) and

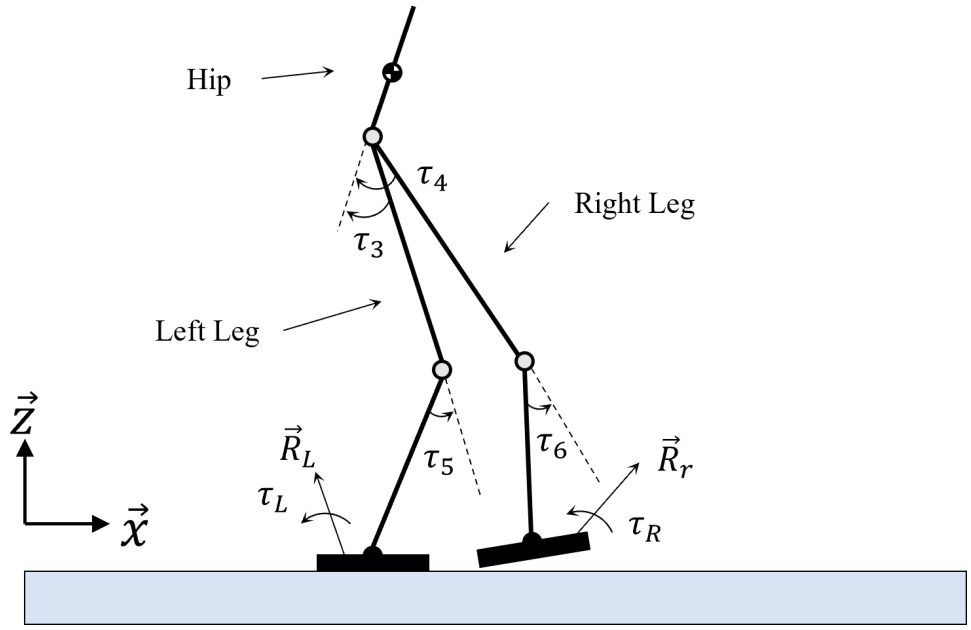


Figure 2.4: Robot torques and ground reaction forces/torques. Here, $\vec{R}$ and $\vec{\tau}$ represent the reaction force/torques between the support foot and the ground.

hip (h), denoted as H^f_{GC} and H^h_{GC}, respectively. The homogeneous transformations for the left foot and the hip can be computed as:

$$H_f^{GC} = \begin{bmatrix} R & P_L \\ 0 & 1 \end{bmatrix} \qquad H_h^{GC} = \begin{bmatrix} R & P_h \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

which can be written as:

$$\mathbf{o}_f^{GC} = H_f^{GC}\mathbf{o}_f^f \qquad \mathbf{o}_h^{GC} = H_h^{GC}\mathbf{o}_h^h \tag{2.2}$$

where $\mathbf{o}_j^i$ denotes the origin of frame j in frame i. Using this notation, we can express a vector p^i with respect to frame i as:

$$p^{GC} = H_f^{GC}p^f = H_h^{GC}p^h \tag{2.3}$$

Therefore, the homogeneous transformation from the foot frame to the hip frame is given by:

$$p^h = (H_h^{GC})^{-1}H_f^{GC}p^f \tag{2.4}$$

Let's define $H_f^h = (H_h^{GC})^{-1}H_f^{GC}$ as the transformation matrix from the foot frame to the hip frame. This transformation can also be computed as:

$$H_f^h = H_t^h(q_3)H_l^t(q_5)H_f^l(q_3, q_5) \tag{2.5}$$

Where $H_t^h(q_3)$ is the transformation from the thigh frame to the hip frame, $H_l^t(q_5)$ is the transformation from the leg frame to the thigh frame and $H_f^l(q_3, q_5)$ is the transformation from the foot frame to the leg frame. It should be noted that the last transformation is only applicable when the foot is on the sole, which constrains the orientation of the foot. Otherwise, the transformation assumes that the foot remain parallel to the ground.

These transformations are defined by the Denavit-Hartenberg convention, where the homogeneous transformation matrix is defined as:

$$H_i^{i-1} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

The DH parameters are defined in Table table 2.1 using the reference frame defined in Figure fig. 2.5. By utilising these parameters, we can derive the direct kinematic model of the biped robot. However, it is necessary to incorporate a small offset into the result to take into account the height of the foot, denoted as d_4, which has not been taken into consideration. To address this issue, we can compute the point p^h with respect to the hip frame by adjusting it with an offset of d_4 relative to the point p^f in the foot frame:

$$p^h = H_f^h(q_3, q_5) \begin{bmatrix} p_x^f \\ p_y^f - d_4 \\ p_z^f \end{bmatrix} \tag{2.7}$$

We have demonstrated that using the DH convention, we are able to develop a direct kinematic model for the left leg. The direct kinematic model for the right leg would be identical, with the only difference being the replacement of q_3 with q_4 and q_5 with q_6. This similarity is expected since the robot is planar and symmetric.

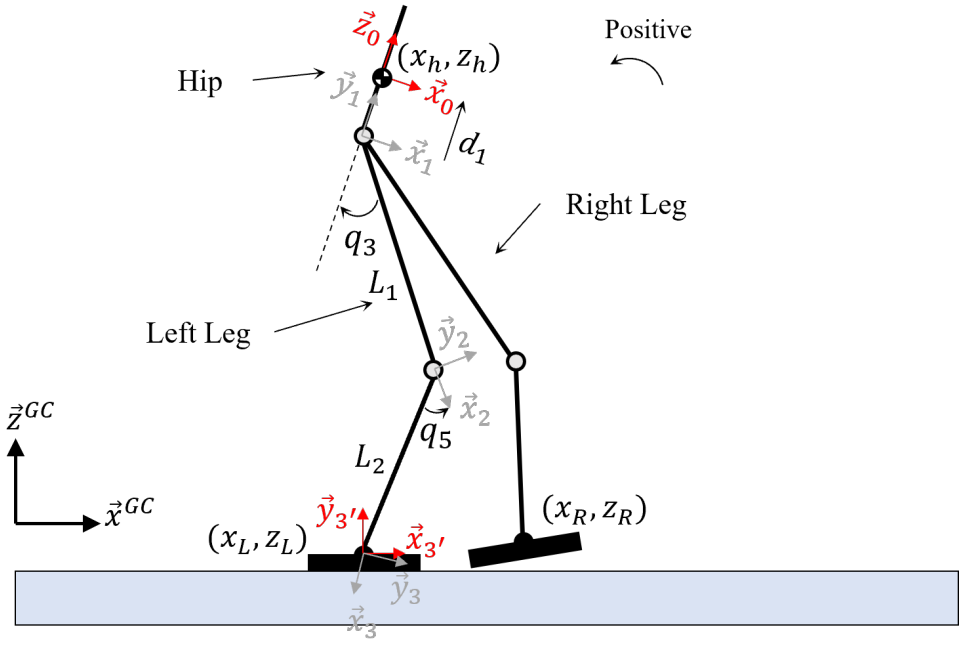


Figure 2.5: Robot local frames according to Denavit-Hartenberg convention. The red frame shows the initial (0) and final frame (3') for the direct kinematics process. It should be noted that a point in the foot frame is defined by its xy coordinates, while a point in the hip frame is defined by its xz coordinates.

2.3.2 Inverse Kinematics

The direct kinematics allows the conversion of joint angles into position of the end-effector in the reference frame. This result is not really interesting in our case the walking gait will be defined by 2 trajectories (CoM and foot position) with respect to the global coordinate. Therefore, we already know the position of the end-effector and we need to compute the joints angles which is done using the

Table 2.1: Denavit-Hartenberg parameters for the biped robot

Joint i	a_i	α_i	d_i	θ_i
1	0	$\frac{\pi}{2}$	$-d_1$	0
2	L_1	0	0	$q_3 - \frac{\pi}{2}$
3	L_2	0	0	$-q_5$
3'	0	0	0	$-q_5 + q_5 + \frac{\pi}{2}$

inverse kinematics of the biped robot.

There are several methods available to obtain the inverse kinematics of a mechanical system, primarily numerical and analytic approaches.

The numerical solution involves computing the joint angles through trial and error, leveraging the known forward kinematics model. This computation is relatively efficient, involving matrix multiplication with a vector. Kajita *et al.* [12] explains this methodology in his book. Figure 2.6 illustrates a scheme that employs this basic concept to compute the appropriate joint angle.

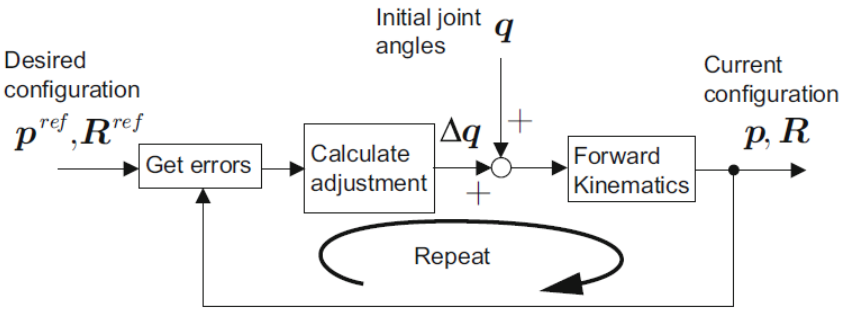


Figure 2.6: Basic Concept Behind Numerical Approach to Inverse Kinematics [12]

The analytical solution provides more accurate results compared to the numerical approach. However, deriving an analytic solution might be challenging or even impossible due to the highly nonlinear nature of the system. Nonetheless, we can take advantage of the geometric shape of the robot and solve the inverse kinematics problem using trigonometry and geometric properties.

To simplify our problem, let's focus on finding the joint angles between the hip articulation and the ankle. We can observe that we have an inverse kinematics problem similar to a double pendulum. Therefore, we can solve it easily. Let's consider only the right leg, as shown in Figure 3. We define R as the length between the origin of the hip-joint frame and the ankle-joint frame, where $\begin{bmatrix} x & z \end{bmatrix}_h^T$ represents the position of the end-effector (ankle joint) with respect to the hip joint frame.

By employing the generalised Pythagorean theorem in the red triangle shown in Figure 3, we can express it as:

$$R^2 = L_1^2 + L_2^2 - 2L_1L_2\cos(\theta) \tag{2.8}$$

Where θ denotes the angle between the thigh link and the leg link. We can also observe that θ is related to q_6 as follows:

$$q_6 = \pi - \theta \tag{2.9}$$

Thus, utilising Equations 2.8 and 2.9, we can write:

$$\cos(q_6) = \frac{R^2 - L_1^2 - L_2^2}{2L_1L_2} = \frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} \tag{2.10}$$

Consequently, the joint angle q_6 can be calculated as:

$$q_6 = \arccos\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2}\right) \tag{2.11}$$

To determine the last joint angle, we will utilise the blue rectangle and the fact that $\sin(q_6) = \pm\sqrt{1 - \cos^2(q_6)}$. Let β represent the absolute angle of the hip joint with respect to the z-axis, such that $\tan(\beta) = \frac{-x}{-z}$, and γ be the angle shown in Figure 2.7. We can then write:

$$\tan(\gamma) = \frac{L_2 \sin(q_6)}{L_1 + L_2 \cos(q_6)} \tag{2.12}$$

Finally, the hip-joint angle is computed as:

$$q_4 = -\alpha + \beta + \gamma = -\alpha + \text{atan2}\left(\frac{L_2 \sin(q_6)}{L_1 + L_2 \cos(q_6)}\right) + \text{atan2}(-x, -y) \tag{2.13}$$

This provides two sets of solutions. As we aim to mimic human walking, we choose the solution where the knee joint is positioned toward the forward step motion.

To verify if the position of the ankle-joint lies outside the robot's workspace, which would result in a singular configuration, the square of Equation 2.10 should be greater than one. Thus, we need to avoid this case.

In summary, we have computed the inverse kinematics problem for the right leg by solving a geometric problem. To obtain the results for the left leg, we can use the same equations since our biped robot is considered planar. Thus, we obtain q_3 and q_5 instead of q_4 and q_6. Therefore, by obtaining the trajectories of the right and left ankle joints (with respect to the hip joint), the inverse kinematics will yield the corresponding hip and knee joint trajectories.

2.4 Robot Dynamics

2.4.1 Forces and Torques

To accurately simulate the biped robot, it is crucial to develop a correct multi-body dynamic system model. Initially, we simplify the simulation by considering only the main 'relevant' forces and torques. Here, 'relevant' implies that we exclude parasitic forces such as joint friction forces, drag forces, and so on. Furthermore, we neglect the impact of the boom on the biped, assuming zero mass and inertia for the boom. Finally, we consider only rigid bodies. Hence, we can enumerate all the forces and torques acting on the robot:

- Inertial forces;
- Inputs torques;
- Gravity force;
- Contact forces.

Inertial forces arise due to the consideration of rigid bodies with distributed masses for each link. Input torques originates from the motors, which are responsible for controlling the biped. The gravity force is taken into account by applying a constant downward acceleration of 9.81 m/s. Contact forces are essential to model the behaviour of the foot whenever it makes contact with the ground (solid-to-solid contact).

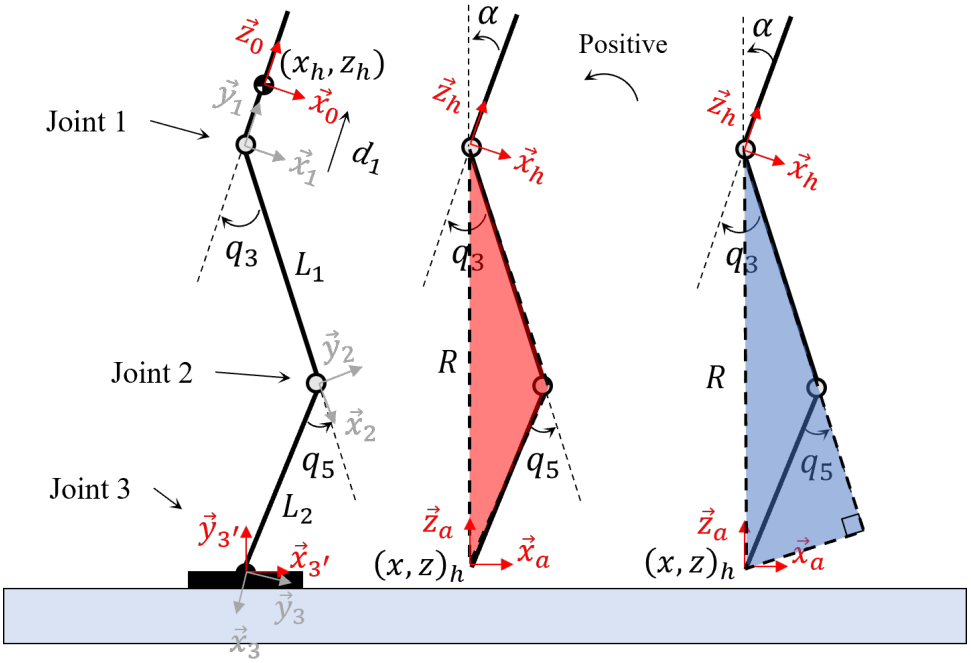


Figure 2.7: Analytic solution to the Inverse Kinematics problem. The left side of the figure shows local frames defined in the Direct Kinematics model. The last two schemes show specific triangles used to solve the inverse kinematic problem, where the red frame represents the local hip and foot frame.

2.4.2 Contact models

In this work, we adopt the Hunt and Crossley model [16] for the normal contact force and the viscoelastic Coulomb model [6] for the tangential contact force.

Normal contact model

The Hunt and Crossley contact model consists of a spring in parallel with a nonlinear damper. The contact force, f_n, is governed by the first equation in eq. (2.14), where z represents the penetration, $\dot{z}$ is the penetration velocity, λ_n is the damping constant, k_n is the spring constant, and n is a parameter dependent on the contact surface. The second equation describes the linearised coefficient of restitution e, which models the impact. Here, v_o denotes the normal relative velocity after the collision, v_i represents the normal relative velocity before the collision, and α is a coefficient with the unit s/m.

$$f_n = -(\lambda_n z^n)\dot{z} - k_n z^n \qquad e = -\frac{v_0}{v_i} = 1 - \alpha v_i \tag{2.14}$$

In this model, the damping term is nonlinear and dependent on the penetration depth. While a more comprehensive description can be found in the paper [16], we will summarise by stating that for consistency with Hertzian theory for contact spheres, we choose $n = \frac{3}{2}$, and $\lambda_n = \frac{3}{2}\alpha k_n$ to maintain an approximately linear relationship between e and v_i for small α. By appropriately selecting the parameters α and k_n, which define the contact surface, we can compute the corresponding normal force between a sphere and the contact surface.

Tangent Contact Model

For directions other than the normal direction, we extend the model by incorporating an additional compliance system, consisting of a linear spring and damper. This extension is depicted in fig. 2.8 and is used to implement a Coulomb friction model, where the friction force is a function of the position x and velocity $\dot{x}$.

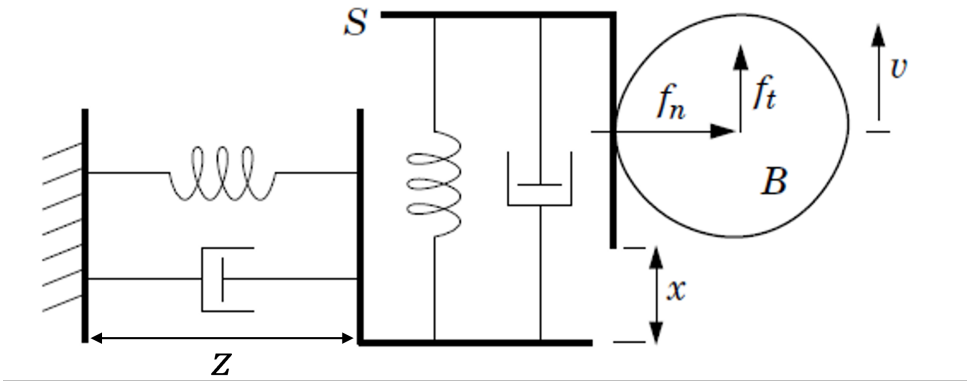


Figure 2.8: Compliant Contact Model [6]. The contact model assumes a spring-damper system to compute the forces in both the normal and tangential directions.

Let x represent the displacement from the equilibrium position of the contact point in the tangential direction, and $\dot{x}$ denote the velocity of the contact point. It can be observed that x and $\dot{x}$ are the tangential counterparts of z and $\dot{z}$, respectively. Consequently, the tangential force is given by:

$$f_t = -k_t x - \lambda_t \dot{x} \tag{2.15}$$

Here, k_t and λ_t are constant coefficients representing the spring and damper in the tangential direction, respectively. However, the objective is to compute both the normal force f_n and the tangential force f_t, or friction force. To accomplish this, we first evaluate the normal force using eq. (2.14) by obtaining the penetration z and its velocity $\dot{z}$:

$$f_n = \begin{cases} 0 & \text{if no contact} \\ \max(0, -(\lambda^n z)\dot{z} - k_n z^n) & \text{else} \end{cases} \tag{2.16}$$

The dynamics are governed by the equation:

$$\dot{z} = -\frac{k_n z^n + f_n}{\lambda_n z^n} \tag{2.17}$$

Next, to compute the tangential force, we need to determine the friction force necessary to prevent slipping:

$$f_{\text{stick}} = -k_t x - \lambda_t \dot{x} \tag{2.18}$$

The tangential force is calculated based on the stick force, f_{stick}, and is constrained within the friction cone:

$$f_t = \begin{cases} f_{\text{stick}} & \text{if } ||f_{\text{stick}}||^2 > (\mu||f_n||)^2 \\ f_{\text{stick}} \frac{\mu||f_n||}{||f_{\text{stick}}||} & \text{else} \end{cases} \tag{2.19}$$

By appropriately selecting the contact surface parameters, we can simulate the contact between the foot and the ground.

In summary, we have presented the normal and tangential contact models to simulate the interaction between the foot and the ground. The normal contact model utilises the Hunt and Crossley model [16], while the tangential model employs a Coulomb model [6].

2.4.3 Robot Dynamics Equations

This section aims to present a typical numerical evaluation of the equations of motion for a biped robot. The robot is treated as a collection of rigid bodies interconnected by various joints and subjected to different forces and torques.

The equation of motion for a general rigid-body system in an unconstrained space can generally be expressed in the following canonical form:

$$M(q)\ddot{q} + C(q, \dot{q}) = Q \tag{2.20}$$

In the above equation, q, $\dot{q}$, and $\ddot{q}$ are vectors representing the generalised position, velocity, and acceleration, respectively. Q is a vector of generalised forces, $M(q)$ is the mass matrix, and $C(q, \dot{q})$ is the generalised bias force that incorporates Coriolis and centrifugal forces, gravity, and external forces acting on the system.

Considering the constraints imposed by the contact with the floor, we need to incorporate them into the equation of motion, resulting in the following form:

$$M(q)\ddot{q} + C(q, \dot{q}) = Q + Q_c \tag{2.21}$$

Where Q_c represents the constraint force, which is typically unknown. However, it can be shown that the constraint force takes the following form:

$$Q_c = J(q)^T \lambda \tag{2.22}$$

In this equation, λ is a vector of unknown force variables, also known as Lagrange multipliers and $J(q)$ is the Jacobian of the system.

Consequently, we can simulate the multi-body dynamic system numerically by using the recursive Newton-Euler algorithm to estimate the forward and backward dynamics of the system.

2.5 Conclusion

In conclusion, this chapter has presented the real robot, from which a simple kinematic model can be derived. Based on this model, we have explained the methodology for computing the direct and inverse kinematic models. The final section has covered the dynamic aspects of the robot, particularly the contact model used in the simulation environment.
Next chapter will focus on ensuring a stable walking gait based on the well-known criterion: the **Zero Moment Point**.

Bipedal Walking

3

How can we define the walking process? Walking can be described as a repetitive action of lifting and placing each foot on the ground in turn, with the performer maintaining balance throughout the process to avoid falling.

It is crucial to note the key terms in this basic definition. The term "**in turn**" implies that at least one foot should be in contact with the ground at any given moment. This excludes the consideration of the running process, which presents a distinct set of challenges. Another significant term is "**not fall**', which is self-explanatory in its meaning of maintaining stability and preventing any loss of balance.

The chapter begins by outlining a general strategy, referred to as a *walking pattern*, to ensure a stable walking process. This is discussed in section 3.1. Subsequently, in section 3.2, a simplified robot model known as the *cart-table model* is presented. Moving on to section 3.3, a preview-control structure is introduced. Finally, section 3.4 explains the computation of the cart-table motion using a preview controller.

3.1 Achieving Stable Walking

One of the most challenging aspects of bipedal locomotion is maintaining contact between the sole of a bipedal robot and an even ground. In order to determine the stability of a walking motion, a criterion is needed. One widely used criterion is the concept of the Zero Moment Point (ZMP).

In 1972, Vukobratovic and Stepanenko [24] defined the ZMP as the point on the sole where the resultant force R passes through the surface of the foot. Figure 3.1 illustrates the definition of the ZMP in two dimensions for a given force distribution across the sole. This concept can be generalised to three dimensions; where, the ZMP is defined as the point where the horizontal component of the moment of the ground reaction force becomes zero [12].

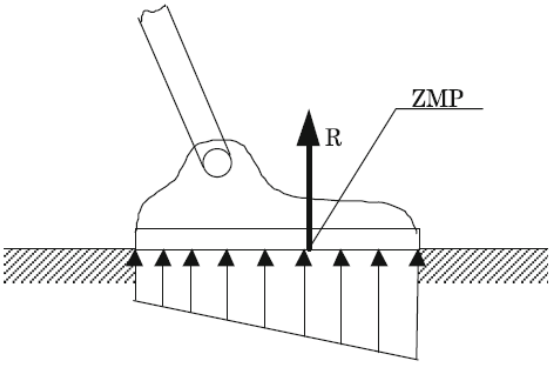


Figure 3.1: Definition of the ZMP [24]

A stable walk occurs when the ZMP remains within the support polygon throughout the motion. The support polygon is defined as the region formed by enclosing all contact points between the robot and the ground with an elastic cord, as depicted in Figure 3.2. In the subsequent section, we will utilise

a simplified robot model to compute the motion of the CoM (Centre of Mass) and ensure the stability of the walking motion.

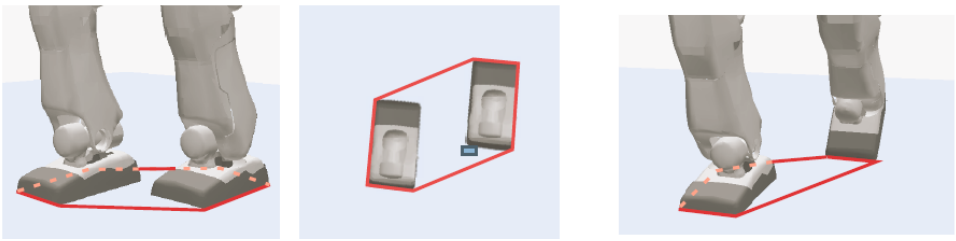


Figure 3.2: Support Polygon [12]. The support polygon is defined as the convex hull for the support foot area

.

3.2 Cart-Table Model

Recall that in section 1.2.1.3, we have presented the cart-table model, initially introduce by Kajita *et al.* [10] in 2003. He proposed this model as a means to simplify the dynamics of a bipedal robot where the primary objective was to transform the nonlinear system into a linear system, enabling the utilisation of classical linear control theory. This model calculates the ZMP using the following equations:

$$p_x = x - \frac{z_c}{g}\ddot{x} \tag{3.1}$$

$$p_y = y - \frac{z_c}{g}\ddot{y} \tag{3.2}$$

Here, p_i represents the ZMP along the i-axis, while x, $\dot{x}$, and $\ddot{x}$ denote the position, velocity, and acceleration of the cart or the CoM trajectory, respectively. z_c corresponds to the height of the table, and g represents the gravitational constant. Since eq. (3.1) is similar to eq. (3.2), we can solve the system considering only the x-direction, for example.

The proposed eq. (3.1) is a simplified version, assuming that the CoM or the cart lies on a flat plane at a constant height, $z = z_c$. A more complex model could incorporate the dynamics of the mass in the z-axis, as expressed by the following equation derived from the inverse pendulum model:

$$p = x - \frac{z - z_c}{g + \ddot{z}}\ddot{x} \tag{3.3}$$

In this work, we focus solely on the simplified equation, neglecting the variation in the height of the CoM.

To solve the ordinary differential equation in eq. (3.1), Kajita *et al.* [10] proposed a closed-loop system with a designated controller. The closed-loop system aims to track the ZMP in real-time, facilitating online walking pattern generations.

In this paper, we employ the same methodology to compute the trajectory of the CoM but for an *offline* walking pattern. The current version of the robot lacks appropriate sensors to measure the CoM or the ZMP. Consequently, we compute joint trajectories for an offline walking gait.

Regarding the cart-table model as a dynamic system, Kajita employed a servo system with feedback control to track the target ZMP reference p_{ref}, as illustrated in Figure 3.3.

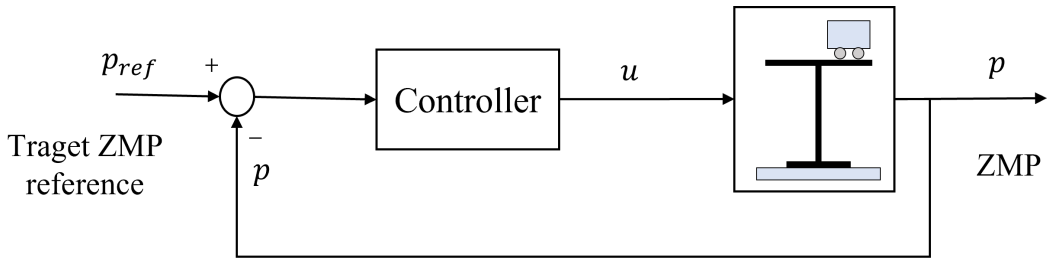


Figure 3.3: Servo-controller to track the target ZMP. The idea here is to use a closed-loop system instead of directly solving the cart-table equation to obtain the cart's motion.

We can introduce a dummy control variable, denoted as $u = \dddot{x}$, to facilitate the representation of the model in state-space form:

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \\ p_x &= \begin{bmatrix} 1 & 0 & -\frac{z_c}{g} \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \end{aligned} \tag{3.4}$$

This system allows us to utilise modern control theory to design a controller. However, when employing standard control methods, the provided walking pattern proves to be inadequate. For instance, in the case of forward step walking, classical control theory using a Linear Quadratic Integral (LQI) regulator yields a change in the CoM after a change in the reference ZMP. Upon closer examination, it becomes apparent that the torso (approximately representing the CoM) lags behind when the foot takes a forward step (resulting in a change in the reference ZMP). This poses a problem as the robot is more susceptible to falling, according to the ZMP criterion.

Figure 3.4 demonstrates this issue, where the LQI regulator (fig. 3.4b) fails to properly track the ZMP reference and exhibits a negative spike that could cause the robot to fall if it exceeds the stability polygon. Additionally, the CoM starts to move after the reference signal. On the other hand, the preview controller (fig. 3.4a) can perfectly track the ZMP reference, and the CoM starts to move **before** the reference signal.

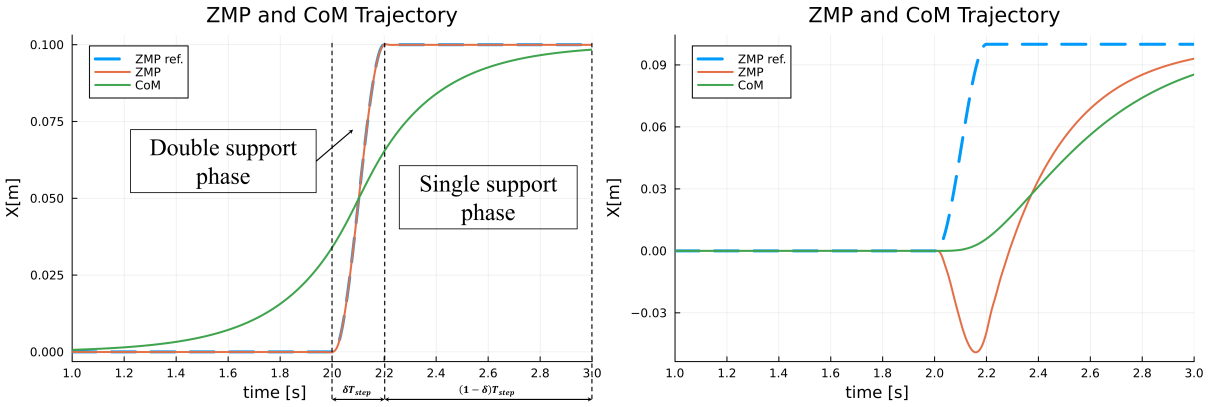
In conclusion, it is evident that any controller cannot be used to solve the cart-table system due to the delay in the output motion after a reference change. In the next section, we will introduce a type of controller called *Preview Control*, initially proposed by Kajita, which employs future information to compensate for this delay.

3.3 Preview Control

3.3.1 Introduction

Preview Control, also known as preview servo control or preview feedback control, is a control theory concept that utilises future information, such as future reference signals or disturbances, to improve the output signal of a system. It was initially introduced by Katayama *et al.* in 1985 [13] and has since been widely used in various control applications.

To illustrate the need for preview control, let's consider the example of driving a car along an unknown road. If the driver can only see a short distance ahead, they can only react to immediate changes in the road, such as sharp curves. This limited sight range can lead to large-control efforts, as the driver needs to make sudden adjustments or even break the car. On the other hand, if the driver



(a) With preview control **(b) With LQI control**

Figure 3.4: ZMP and CoM trajectory for one step. fig. 3.4a utilises the Preview control method where we define the single phase as the duration when the foot swings and the double support phase as the duration when both feet lay on the ground. As a comparison, fig. 3.4a reach perfect tracking (it can generate a CoM trajectory which can ensure the ZMP ref). contrary to fig. 3.4b which utilises a classical LQI regulator

can see farther ahead and anticipate the shape of the road, they can smoothly steer the car without overshooting or requiring drastic control actions.

In the context of bipedal walking control, preview control can be applied to generate smooth and stable motions of the CoM by using future information about the desired ZMP trajectory. By considering future reference signals, the control system can compensate for delays and better track the desired motion.

3.3.2 Problem Formulation for Servomechanism Design

The goal of servomechanism design is to design a control strategy that determines the input $u(k)$ based on the current state $x(k)$ and the desired output $y_d(k)$ to achieve a desired system response. Preview control extends this concept by considering future reference signals $y_d(k+i)$, where i represents the preview horizon. By utilising future information, the control system can improve its performance and response time.

Now, let's formulate the problem for servomechanism designs using a linear time-invariant (LTI) discrete-time system representation, as described by Katayama *et al.* [13]:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{3.5}$$

where:

- $x(k) \in \mathbb{R}^n$ is the state vector at time k.
- $u(k) \in \mathbb{R}^m$ is the input vector at time k.
- $y(k) \in \mathbb{R}^p$ is the output vector at time k.
- $A \in \mathbb{R}^{n \times n}$ is the state matrix.
- $B \in \mathbb{R}^{n \times m}$ is the input matrix.
- $C \in \mathbb{R}^{p \times n}$ is the output matrix.

Then, we will suppose:

- Reference values $r(k), ..., r(k+h)$ are available, where h is the preview able time;

- Futures values of the reference signal $r(k)$ that exceed the time $k+h$ are approximated by:

$$r(k+h+i) = r(k+h) \qquad \forall i = 1, \dots \tag{3.6}$$

  In other words, if we can't see h time steps ahead, we will suppose that the next reference is equal to the last preview able reference.

- No disturbance in the system.

The basic design of a controller need to satisfy theses criteria:

1. No steady-state error;
2. The closed-loop system is asymptotically stable and exhibits acceptable transient responses.

The classical LQI technique will fulfil these requirements. The idea here is to extend the LQI methodology with a preview action which is a function of future values of the reference values up to h time step ahead. First, let's extend the LTI discrete system (3.5) using state augmentation technique:

$$\Delta x(k) = x(k) - x(k-1) \tag{3.7}$$
$$e(k) = y(k) - r(k) \tag{3.8}$$
$$x_d(k) = \begin{bmatrix} r(k+1) & \dots & r(k+h) \end{bmatrix}^T \tag{3.9}$$
$$\xi(k) = \begin{bmatrix} \Delta x(k) & e(k) & x_d(k) \end{bmatrix}^T \tag{3.10}$$

Where $\Delta x(k) \in \mathbb{R}^{n\times 1}$ is the incremental state, $e(k) \in \mathbb{R}^{p\times 1}$ is the actual tracking error, $x_d(k) \in \mathbb{R}^{h\times 1}$ is the preview signal and $\xi(k) \in \mathbb{R}^{(n+p+ph)\times 1}$ is the new state vector.

The new system can be rewritten using the incremental state vector eq. (3.7):

$$\Delta x(k+1) = A\Delta x(k) + B\Delta u(k) \tag{3.11}$$
$$\Delta y(k) = C\Delta x(k) \tag{3.12}$$

The actual error gives a new equation:

$$e(k+1) = e(k) + \Delta y(k+1) - \Delta r(k+1) \tag{3.13}$$
$$= e(k) + CA\Delta x(k) + CB\Delta u(k) - \Delta r(k+1) \tag{3.14}$$

Since the assumption made in eq. (3.6), the preview signal gives another equation:

$$x_d(k+1) = A_d x_d(k) \tag{3.15}$$

Where,

$$A_d = \begin{bmatrix} 0 & I_p & & \mathbf{0} \\ & \ddots & \ddots & \\ & & 0 & I_p \\ \mathbf{0} & & & I_p \end{bmatrix} \tag{3.16}$$

With I_p the identity matrix of dimensions $(p \times p)$

Combining eq. (3.11), eq. (3.14) and eq. (3.16) with the new state vector define in eq. (3.10), we obtain;

$$\xi(k+1) = \begin{bmatrix} A & 0 & 0 & 0 & \dots & 0 \\ CA & I_p & -I_p & 0 & \dots & 0 \\ & 0 & & A_d & & \end{bmatrix} \xi(k) + \begin{bmatrix} B \\ CB \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Delta u(k) \tag{3.17}$$

which is an augmented state-space representation of the initial system. We can perform a classical state-feedback of the new augmented state-space by minimising a performance index. The performance index J of the system 3.17 is defined as:

$$J = \sum_{i=k}^{\infty} \left\{ \xi^T(i) Q \xi(i) + \Delta u^T(i) R \Delta u(i) \right\} \tag{3.18}$$

Where Q and R are definite positive matrix. Since $x_d(k)$ represent future values of $r(k)$, the state $x_d(k)$ should not be included in the performance index J because J should be minimised for any reference signal r. Therefore, J can also be defined as:

$$J = \sum_{i=k}^{\infty} \left\{ \Delta x^T(i) Q_x \Delta x(i) + e^T(i) Q_e e(i) + \Delta u^T(i) R \Delta u(i) \right\} \tag{3.19}$$

Where Q_x, Q_e and R are $(n \times n)$, $(p \times p)$ and $(m \times m)$ symmetric definite positive matrix respectively. One should note that the term $\Delta x^T(i) Q_x \Delta x(i)$ denote the loss due to incremental state, $e^T(i) Q_e e(i)$ means the loss due to the tracking error and $\Delta u^T(i) R \Delta u(i)$ represents the loss due to the control vector.

The optimal incremental control to the performance index eq. (3.19) can be computed using the theory of optimal controller:

$$\Delta u(k) = -G_I e(k) - G_x \Delta x(k) - \sum_{l=1}^{h} G_d(l) \Delta r(k+l) \tag{3.20}$$

Which results from a state-feedback of the augmented system (3.17).

And finally, the optimal controller is given by summing up eq. (3.20) from $i = 0$ to $i = k$:

$$u(k) = -G_I \sum_{i=0}^{k} e(i) - G_x x(k) - \sum_{l=1}^{h} G_d(l) r(k+l). \tag{3.21}$$

Where G_I is the integral gain, G_x is the state-feed back gain, $G_d(l)$ is the preview gains, h is the preview length and $r(k)$ is the reference ZMP. One should notice that the term $G_I \sum_{i=0}^{k} e(i)$ represent the integral action, $G_x x(k)$ is the state-feedback action and $\sum_{l=1}^{h} G_d(l) r(k+l)$ is the preview action. A summary of the controller is sown in fig. 3.5 for a cart-table model where z denote the shift operator.

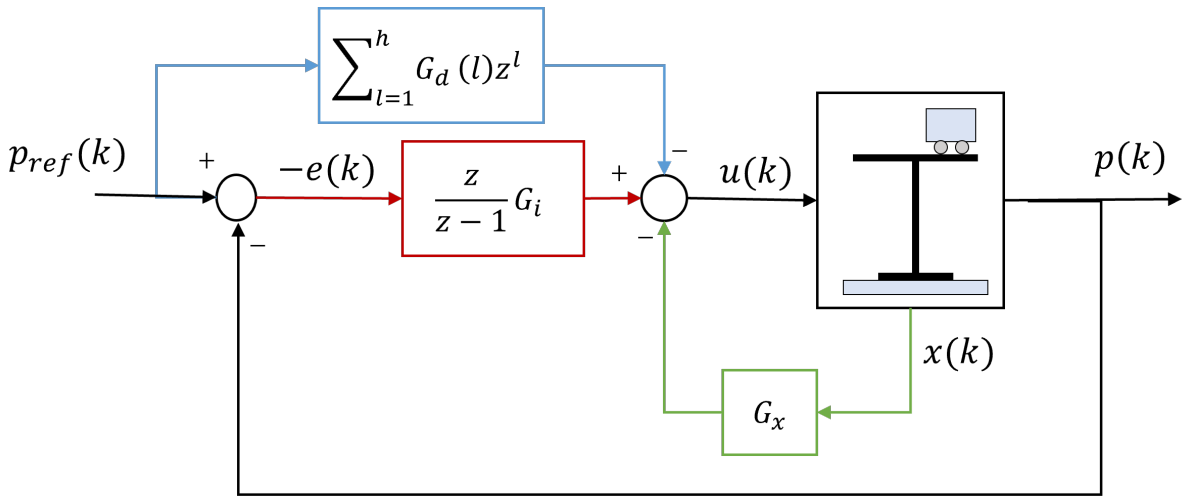


Figure 3.5: Block diagram of the Preview controller. The green box relate the state-feedback action , the red box involve the integral action and the blue box correspond to the preview action.

To compute the gains, let simplify the notation of eq. (3.17):

$$\tilde{B} = \begin{bmatrix} CB \\ B \end{bmatrix} \qquad \tilde{I} = \begin{bmatrix} I_p \\ 0 \end{bmatrix} \qquad \tilde{F} = \begin{bmatrix} CA \\ A \end{bmatrix} \qquad \tilde{Q} = \begin{bmatrix} Q_e & 0 \\ 0 & Q_x \end{bmatrix} \qquad \tilde{A} = \begin{bmatrix} \tilde{I} \\ \tilde{F} \end{bmatrix}$$

Then, the optimal control theory gives the optimal control gains for the performance index define at eq. (3.19):

$$G_I = [R + \tilde{B}^T \tilde{K} \tilde{B}]^{-1} \tilde{B}^T \tilde{K} \tilde{I} \tag{3.22}$$

$$G_x = [R + \tilde{B}^T \tilde{K} \tilde{B}]^{-1} \tilde{B}^T \tilde{K} \tilde{F} \tag{3.23}$$

$$G_d(1) = -G_I \tag{3.24}$$

$$G_d(l) = [R + \tilde{B}^T \tilde{K} \tilde{B}]^{-1} \tilde{B}^T \tilde{X}(l-1), \qquad l = 2, ..., h \tag{3.25}$$

Where $\tilde{K}$ is a $(p+n) \times (p+n)$ definite positive matrix defined by the solution of the Riccati equation:

$$\tilde{K} = \tilde{A}^T \tilde{K} \tilde{A} - \tilde{A}^T \tilde{K} \tilde{B} [R + \tilde{B}^T \tilde{K} \tilde{B}]^{-1} \tilde{B}^T \tilde{K} \tilde{A} + \tilde{Q} \tag{3.26}$$

And the $(p+n) \times p$ matrices $\tilde{X}$ are computed with:

$$\tilde{X}(1) = -\tilde{A}_c^T \tilde{K} \tilde{I} \tag{3.27}$$

$$\tilde{X}(l) = \tilde{A}_c^T \tilde{X}(l-1), \qquad l = 2, ..., h \tag{3.28}$$

Where $\tilde{A}_c$ is the closed-loop matrix given by:

$$\tilde{A}_c = \tilde{A} - \tilde{B} [R + \tilde{B}^T \tilde{K} \tilde{B}]^{-1} \tilde{B}^T \tilde{K} \tilde{A} \tag{3.29}$$

If the reader wants to see further, Katayama *et al.* [13] shows more detailed results.

To summarise, this section covers a way to design a *Preview Control*, actually proposed by Katayama *et al.* in [13], by optimising a performance index defined in eq. (3.19). Thus, we compute the gains of controller G_x, G_i and G_d, which are linked, respectively, to the state-feedback action, integral action and preview action.

3.4 Preview Control for the Cart-Table Model

In this section, we will discuss the application of preview control for the cart-table model defined in Equation 3.4 to estimate the trajectory of the CoM in accordance with the reference ZMP. To begin, we need to discretise the cart-table system at a sampling rate of T.

The discrete model of the cart-table system is given by:

$$\begin{aligned} x_x(k+1) &= \underbrace{\begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}}_{A} x_x(k) + \underbrace{\begin{bmatrix} \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \end{bmatrix}}_{B} u \\ p_x(k) &= \underbrace{\begin{bmatrix} 1 & 0 & -\frac{z_c}{g} \end{bmatrix}}_{C} x_x(k) \end{aligned} \tag{3.30}$$

Here, $x_x(k) = \begin{bmatrix} x(k) & \dot{x}(k) & \ddot{x}(k) \end{bmatrix}^T$ represents the state vector for the CoM in the x-axis, and $p_x(k)$ denote the ZMP in the x-axis.

Applying the preview control theory, we define the performance index as shown in Equation 3.19. The weight matrices Q_x, Q_e, and R need to be chosen. To preserve the natural dynamics of the system, the weight matrix on the incremental state vector should be set to:

$$Q_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.31}$$

Following Kajita *et al.* in [10], we choose the values of Q_e and R as:

$$Q_e = 1 \qquad R = 10^{-6} \tag{3.32}$$

Next, we solve the Riccati equation given by Equation 3.26. The fixed-point iteration method is commonly used to solve this equation. The equation can be written in the form of a function:

$$x_{n+1} = f(x_n) \qquad n = 0, 1, 2, \ldots \tag{3.33}$$

Here, x_n represents the solution of the Riccati equation $\tilde{K}$, and $f(x_n)$ is the Riccati equation itself. By iterating this equation with an initial condition of the identity matrix, the solution converges to a solution near the initial condition. As the number of iterations approaches infinity, the solution becomes exact.

Using eq. (3.22), eq. (3.23), eq. (3.24), and eq. (3.25), we can compute the optimal controller. Once the controller is defined, we can solve the cart-table system to obtain the state vector corresponding to the cart motion. However, we have not yet explained how to choose the preview time t_h, which determines how far into the future we can see the reference signal. The next section will present a numerical example that demonstrates a simple methodology for defining t_h.

Numerical Example

In this section, we will illustrate the use of the *Preview controller* with arbitrary numerical values. Let's assume the following:

- $z_c = 0.65$ m and $g = 9.81 \frac{\text{m}}{\text{s}}$;
- The sampling time T is set at 0.05 s and the step period T_{step} is set to 2 s;
- The path will be defined by thesis equations:

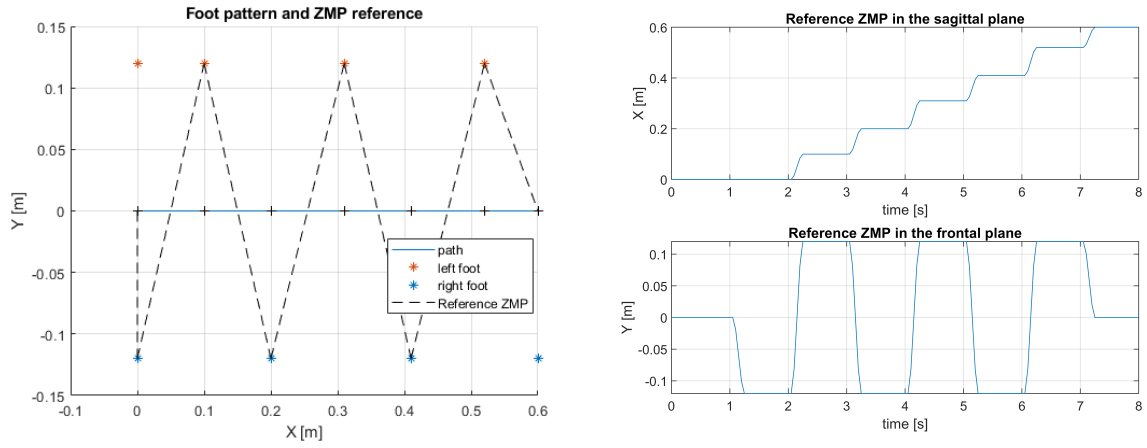
  $$x(t) = 0.01t \qquad y = 0 \qquad t = 1, \ldots, 60 \tag{3.34}$$

  Where x is the x-coordinate of the path, y is the y-coordinate of the path and t is the parameter.
- The reference ZMP shown in fig. 3.6a is defined to follow the path described in eq. (3.34). The ZMP can be considered as the position of the support foot, resulting in a step-like function where the step values represent the x or y coordinates of the support foot. However, the double support phase is not properly defined.
- To smoothen the ZMP trajectory, let's assume that during the double support phase, the reference ZMP follows a third-order polynomial defined as:

  $$p(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \qquad i = x, y \tag{3.35}$$

  Here, a_j represents constant coefficients, t represents discrete time, and p represents the x-coordinate of the ZMP. These coefficients will be determined by solving a system of equations using the following initial conditions:

  $$p(0) = p_{\text{start}} \qquad p(T_{\text{step}}) = p_{\text{end}} \qquad \frac{dp}{dt}(0) = 0 \qquad \frac{dp}{dt}(T_{\text{step}}) = 0$$



(a) ZMP in the XY-plane **(b) ZMP in sagittal and frontal plane**

Figure 3.6: Reference ZMP trajectory. These trajectories will be fetch to the Preview Control in order to generate the CoM trajectory.

By satisfying these conditions, we obtain a smoother ZMP trajectory. The resulting trajectory is shown in fig. 3.6b.

The gains for the controller are computed using eq. (3.23) and eq. (3.22) as follows:

$$G_x = \begin{bmatrix} 2372.57 & 738.03 & 36.60 \end{bmatrix}^T \qquad G_I = 207.35 \tag{3.36}$$

The preview gains can be calculated for a sufficiently large preview length h, and the values can be plotted as shown in fig. 3.7. To determine an "optimal" preview length h, one approach is to choose a preview time t_h where the gain $G_d(h) \approx 0$. Adding information close to zero to the control input $u(k)$ is unnecessary. This allows us to select a suitable candidate for the preview time t_h, which relates to the preview length h as follows:

$$t_h = T(h-1) \quad \Leftrightarrow \quad h = \frac{t_h}{T} + 1 \tag{3.37}$$

For instance, in fig. 3.7, the gains $G_d(l) \quad l = 1, ..., 41$ are shown. It can be observed that the preview gains ranging from $t_h = 1.6$ s to $t_h = 2$ s are negligible. Therefore, we can reduce the preview time to, for example, $t_h = 1.6$ s without significantly affecting the output and reducing overall computational consumption. It is worth noting that this method is not optimal since the preview time is determined iteratively. Other approaches, such as searching for the optimal preview step h, can be found in the reference [15].

Finally, using the control input $u(k)$ defined in eq. (3.21), the trajectory of the CoM can be computed as shown in fig. 3.8. Overall, the tracking of the reference ZMP is good, at least when the reference signal is available. However, when the reference signal cannot be previewed, the actual ZMP does not track properly, as seen in fig. 3.8.

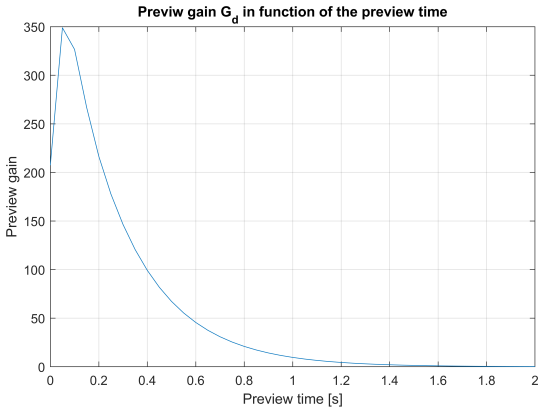


Figure 3.7: Preview Gains for a preview time of 2 seconds. Here, one can remark that the preview gain G_d is negligible for $t_h > 1.6$ s. Thus, information more than 1.6 s future ahead will not contribute significantly to the control performance.

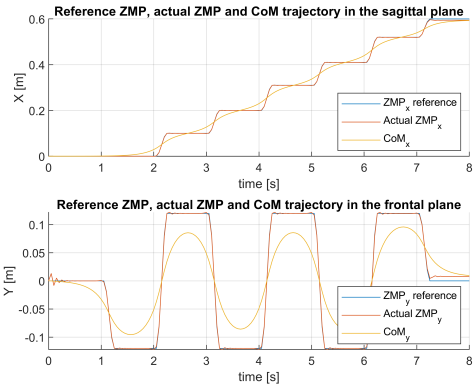


Figure 3.8: CoM and ZMP trajectories from the closed-loop system of the cart-table model

3.5 Conclusion

In this chapter, we have demonstrated the utilisation of the ZMP stability criteria with a simple robot model known as the **Cart-Table**, originally introduced by Kajita *et al.* in [10]. Our objective was to generate the motion of the cart, representing the CoM of the robot, by implementing a **Preview Control** approach based on Katayama *et al.* in [13]. Through numerical experiments, we have successfully shown that the resulting cart motion aligns with the desired ZMP trajectory.

In the next chapter, we will discuss a high-level controller that generates the joint angles for a given path, further expanding on the control of the bipedal robot.

Development of a ZMP based Controller for our Robot

4

4.1 Presentation of the controller

In Chapter 3, we discussed how to achieve stable walking motion using the ZMP concept. Now, the question arises: How do we implement this on our bipedal robot? One practical approach is to use a **ZMP-based controller** that generates the joint angles necessary for stable motion based on the ZMP criterion.

We propose our own version of the ZMP-based controller, as shown in Figure 4.1. The controller is divided into several internal blocks:

- *Foot Planner*: Evaluates the landing position of the right and left foot.
- *Swing Foot Trajectory*: Determines the 3D position of the swing foot.
- *ZMP Trajectory Generator*: Defines the reference ZMP trajectory to remain within the support polygon.
- *CoM Trajectory Generator*: Computes the CoM trajectory and the hip body position based on the reference ZMP.
- *Inverse Kinematics*: Converts the local foot position in workspace coordinates ($\begin{bmatrix} x & y & z \end{bmatrix}^T$) into joint space coordinates ($\begin{bmatrix} q^{\text{hip}} & q^{\text{knee}} \end{bmatrix}^T$).

Once the reference joint trajectories are generated, they can be fed into the simulation environment, which will be described in more detail in the next chapter (Chapter 5). Our version of the controller differs from what is available in existing literature [14, 23] in two ways:

- *Active Balance Control*: This feature allows the robot to compensate for trunk orientation and ensure that the hip remains vertical.
- *ZMP Feedback*: By measuring the actual ZMP of the robot, this feature enables tracking of the predicted ZMP. Consequently, the robot becomes more resilient to disturbances and compensates for model error.

For simplicity, we have excluded these two functional blocks from our design. The *Active Balance Control* can be neglected if we assume that the rotation of the hip along the y-axis is fixed (by blocking the y-revolute joint of the boom, as explained in Section 2.1). Additionally, incorporating ZMP feedback would complicate the controller by introducing a connection between the simulation environment and the ZMP-based controller. Therefore, we will initially neglect the feedback system in our controller. If required, we can develop a new controller to incorporate this feedback system.

In the following sections, we will explain the various blocks of the ZMP-based controller. Section 4.2 describes the *Foot Planner* block, followed by Section 4.3 which covers the *Swing Foot Trajectory* block. The generation of the reference ZMP and CoM trajectories will be discussed in Section 4.4 and Section 4.5, respectively, using the *ZMP Trajectory Generator* and *CoM Trajectory Generator* blocks. Section 4.6 focuses on the *Inverse Kinematics* block, which converts the foot position into joint angles. Finally, in Section 4.7, we present a numerical example of the ZMP-based controller.

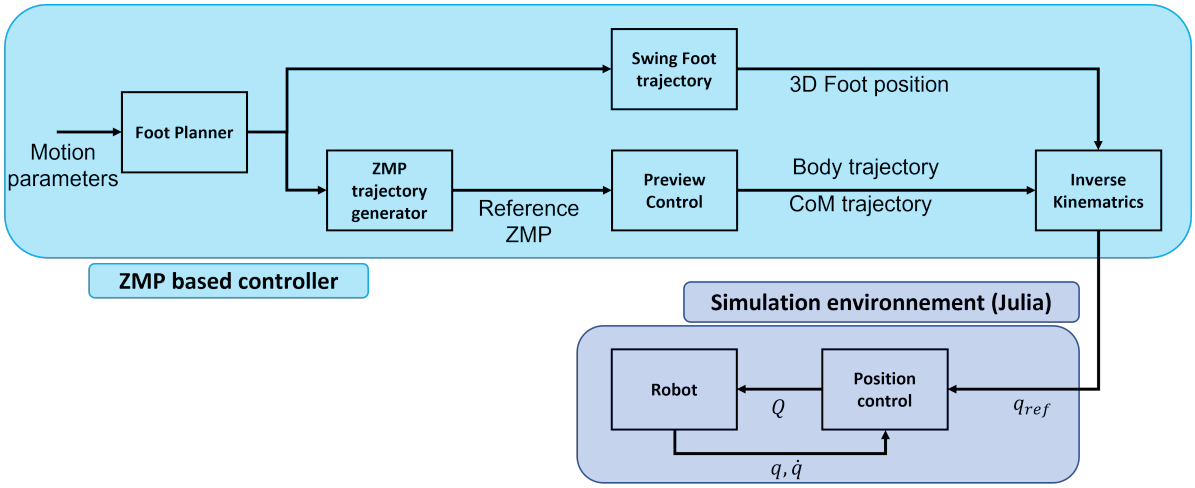


Figure 4.1: Our version of the ZMP based controller basic diagram. The ZMP based controller will generate the reference joint angles which are used in the Position Control.

4.2 Foot Planner

The objective of this block is to generate future values for the positions of the support feet of the bipedal robot. During the walking process, the robot needs to determine where to place its feet to adhere to the planned motion. There are two main types of foot planners:

- Setting future foot positions based on the robot's current state and desired velocity vector ([23], [8]).
- Setting future foot positions based on the intended path to follow ([14]).

The first approach is optimised for environments without obstacles or safe path calculations. The second approach doesn't require a safe path calculator and can follow complex paths provided to it. In this work, the second method is more suitable for path following, allowing specification of any path without a safe path calculation.

4.2.1 Working Principle

An effective foot planner should be able to compute the support feet positions for any path in an environment with any number of obstacles. The path should be feasible and avoid collisions with the robot itself or the environment.

The desired path is defined parametrically with $x(t)$ and $y(t)$ representing the path coordinates in the global frame. A typical path definition is:

$$x(t) = f(t) \qquad y(t) = g(t) \qquad t = 0, ..., N-1 \tag{4.1}$$

Here, $f(t)$ and $g(t)$ are parametric equations where t is the parameter, and N is the number of points. The path's initial position is defined when $t = 0$, which is also a suitable choice for setting the origin of the global frame with zero orientation of the robot (x-axis aligned with the sagittal plane). For example, a straight path along the x-axis can be defined as:

$$x(t) = ct \qquad\qquad y(t) = 0 \tag{4.2}$$

Where, c represents the distance between two points in metres, and for this case, we set $c = 0.01$ m.

The foot planner requires additional inputs:

1. Maximum step length $L_{\max}$

2. Maximum rotation angle for a step $\theta_{\max}$
3. Centre-to-centre distance between the right and left foot of the bipedal robot d

These inputs are defined by the kinematic constraints of the bipedal robot, where the step length $L_{\max}$ is calculated to avoid any singular configurations, and $\theta_{\max}$ is specified for a particular robot configuration.

The foot planner algorithm aims to iteratively determine the next foot position and orientation using a reference point located on the path. Let's assume that the robot is at the starting position corresponding to $t = t_0 = 0$. The algorithm computes the next kinematically viable reference point (corresponding to $t = t_1$) on the path using the previous reference point at $t = t_0$, and so on. Once the entire path has been explored, i.e., $t = N - 1$, the path will have been sampled at n steps, each representing a kinematically viable reference point corresponding to the parameter values $t_0, \ldots, t_n$.

We can denote the set of reference points as $\left\{\left[x(t_0), y(t_0), \theta(t_0)\right]; \ldots; \left[x(t_n), y(t_n), \theta(t_n)\right]\right\}$, where x and y are the coordinates of the reference point, and θ is the orientation angle defined as the angle between the tangent line at the reference point and the x-axis. For simplicity, a reference point i can be denoted as $\left[x_i, y_i, \theta_i\right]$.

A reference point is considered kinematically viable if and only if the step length L_{step} and the rotation angle $\Delta\theta$ satisfy the following conditions:

$$L_{\text{step}} = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \leq L_{\max} \tag{4.3}$$

$$\Delta\theta = \theta_i - \theta_{i-1} \leq \theta_{\max} \tag{4.4}$$

Here, $i = 0, ..., n$ represents the index of the reference points, and n is the number of steps. But how can we find reference points that satisfy these conditions for the parameters $t_0, \ldots, t_n$? In practice, we can first find a point that maximises the first condition eq. (4.3) and then check if this point satisfies the second condition eq. (4.4). If the point on the path fails to meet the second condition, we backtrack to the first condition and reduce the maximum step length to find a new reference point. We can then recheck the second condition with this new reference point and iterate if necessary. By progressively reducing the step length, we eventually obtain a point that satisfies both conditions. If the maximum step length $L_{\max}$ is decreased to zero, the robot needs to rotate to achieve the correct orientation.

Once the entire path has been explored, the algorithm provides the set of reference points $\left[x(t_0), y(t_0), \theta(t_0)\right]; \ldots; \left[x(t_n), y(t_n), \theta(t_n)\right]$, and these reference points represent the positions of each foot. The relationship between a reference point and the foot position is defined as follows:

$$x_{f,i} = x_i \pm \frac{d}{2}\cos(\theta_i) \tag{4.5}$$

$$y_{f,i} = y_i \pm \frac{d}{2}\sin(\theta_i) \tag{4.6}$$

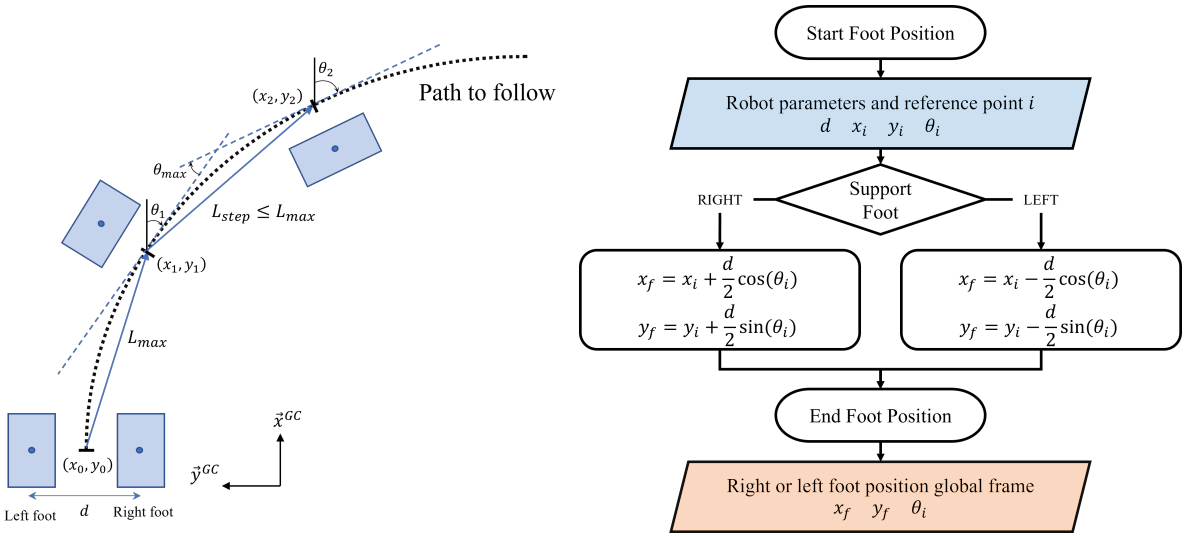
Here, $(x_{f,i}, y_{f,i})$ represents the coordinates of the foot for the reference point i, d is the centre-to-centre distance between the right and left feet, and θ_i is the global orientation of the robot relative to the global frame. The sign is determined based on whether the right foot (negative sign) or the left foot (positive sign) is in the support phase. Figure 4.2a shows a simple diagram illustrating the foot planner algorithm for the first two steps.

4.2.2 Implementation of the Foot Planner

Using the approach explained in the previous section, we can develop an algorithm to compute the i^{th} step for a defined path based on the paper [14]. The flowchart of the path planner is shown in Figures

4.3a, 4.2b, and 4.3b. The output of each function is shown in a red box, while the input is shown in a blue box.

The *Foot Planner* requires the robot, motion, and path parameters as input. It then computes the next reference point using the function described in Figure 4.3b. Based on the support foot, the algorithm calculates the position of the new foot after the step and repeats the process if the robot has not reached the end of the path.



(a) Foot Planner scheme. Only the first two step are shown.

(b) Flowchart: Foot Position function. The output is shown in a red box and the input in a blue box.

Figure 4.2: Foot Planner Implementation

The *Next Point* function (Figure 4.3b) calculates the reference point by first finding a point that maximises the step length and then iterate until the point is kinematically viable . The *Foot Position* function (Figure 4.2b) converts a reference point into a foot position.

4.3 Swing Foot Trajectory

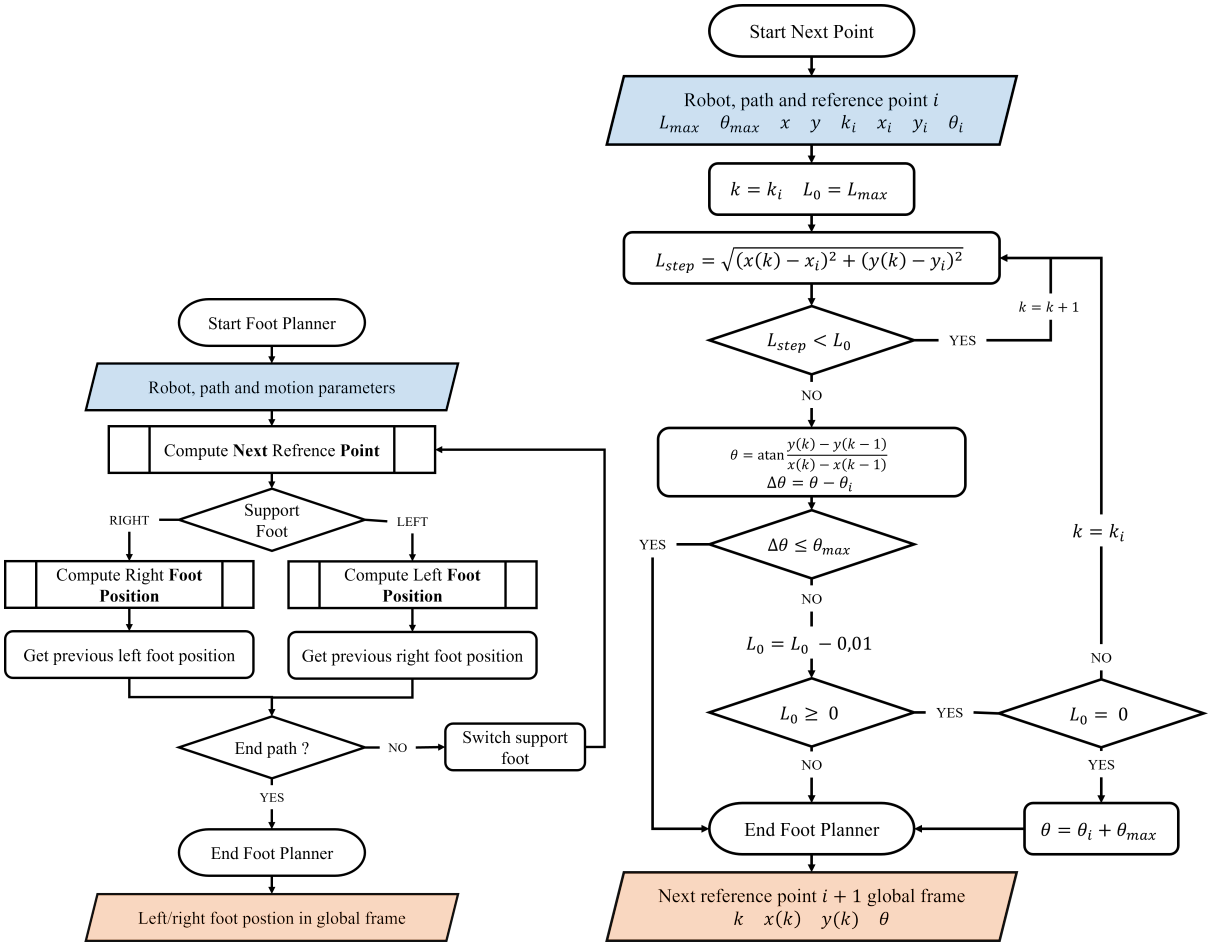
The purpose of the swing foot trajectory is to determine time-dependent functions for x, y, z, and θ that describe the position and orientation of the foot during the swing phase of the walking gait. There are various methods available to generate a smooth swing trajectory, such as:

- Polynomial interpolation, as described in [22].
- Trigonometric functions, as proposed in [14].

In this study, we will adopt the second method for constructing our trajectory profile, as it offers simplicity and ensures zero velocity at foot contact through the use of cosine functions.

4.3.1 Working principle

An effective swing foot trajectory computes an appropriate trajectory for each foot during the swing phase. The algorithm requires the initial and final foot positions as input. Let us denote the starting point coordinates as $[x_s, y_s, \theta_s]$, and the end position coordinates as $[x_f, y_f, \theta_f]$. Based on the work by [14], the following time-dependent functions are proposed to compute the position and orientation of



(a) Foot Planner (main) **(b) Next Reference Point function**

Figure 4.3: Foot Planner Flowchart: Foot Planner and Next Reference Point. The output is shown in a red box and the input in a blue box.

the foot for a step:

$$x(t) = \begin{cases} x_s & \text{if } t \leq t_{ds} \\ \frac{x_f - x_s}{2}\left(1 - \cos\left(\frac{\pi(t - t_{ds})}{T_{\text{step}} - t_{ds} - t_{\text{ver}}}\right)\right) + x_s & \text{if } t_{ds} < t \leq T_{\text{step}} - t_{\text{ver}} \\ x_f & \text{if } t > T_{\text{step}} - t_{\text{ver}} \end{cases} \tag{4.7}$$

$$y(t) = \begin{cases} y_s & \text{if } t \leq t_{ds} \\ \frac{y_f - y_s}{2}\left(1 - \cos\left(\frac{\pi(t - t_{ds})}{T_{\text{step}} - t_{ds} - t_{\text{ver}}}\right)\right) + y_s & \text{if } t_{ds} < t \leq T_{\text{step}} - t_{\text{ver}} \\ y_f & \text{if } t > T_{\text{step}} - t_{\text{ver}} \end{cases} \tag{4.8}$$

$$z(t) = \begin{cases} 0 & \text{if } t \leq t_{ds} \\ \frac{h_{\text{step}}}{2}\left(1 - \cos\left(\frac{2\pi(t - t_{ds})}{T_{\text{step}} - t_{ds}}\right)\right) & \text{if } t_{ds} < t \leq T_{\text{step}} - t_{ds} \end{cases} \tag{4.9}$$

Here, T_{step} represents the walking period, $t_{ds} = \delta T_{\text{step}}$ denotes the duration of the stance phase, where δ represents the ratio between the stance phase and the step period, t_{ver} represents the duration of the vertical motion, and h_{step} defines the maximum foot height. As the foot orientation is not actively controlled, the orientation of each foot is assumed to be the same as the robot's orientation, denoted by θ.

Figure 4.4 illustrates the behaviour of the swing foot for a single step. The swing foot trajectory

generator primarily relies on three parameters (δ, t_{ver}, and h_{step}) that can be adjusted to modify the swing foot behaviour:

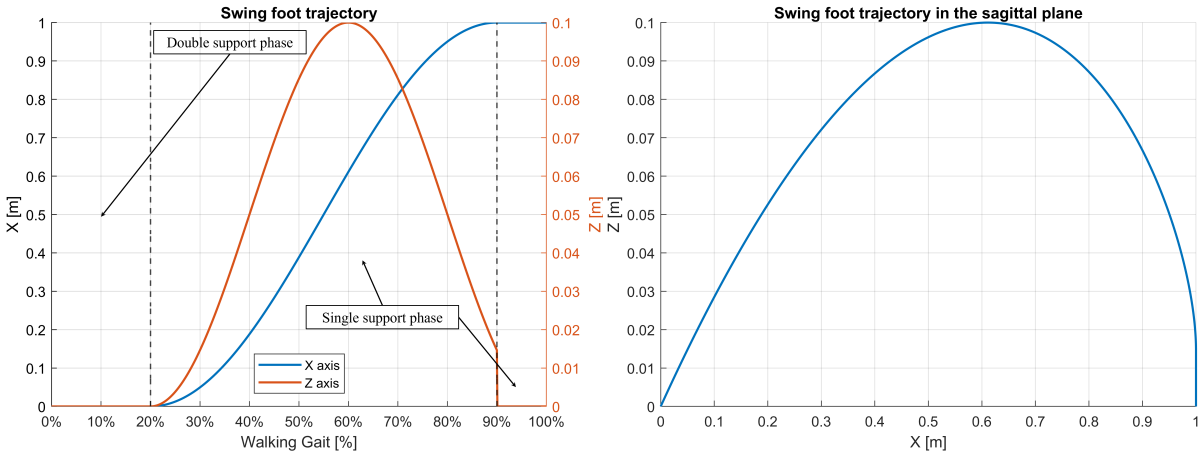
- Increasing δ: This will result in faster motion of the swing foot.
- Increasing t_{ver}: the height of the vertical motion will be higher, which is useful for climbing stairs or overcoming obstacles.
- Increasing h_{step}: this automatically contributes to the maximum height of the vertical motion.

Since we assume that the robot will be walking on a controlled environment (flat ground without obstacles), we can set $t_{\text{ver}} = 0$. It is worth noting that in Figure 4.4b, the swing foot is depicted in the sagittal plane, considering our planar robot assumption.

4.3.2 Implementation of the Swing Foot Trajectory

The flowchart presented in Figure 4.5b illustrates the implementation of the swing foot trajectory. The input consists of motion parameters such as the step period T_{step}, the double phase ratio δ, and the set of right and left foot positions along the path $\{x_0, y_0, \ldots, x_n, y_n\}$. The output provides the 3D position of each foot for the entire path, denoted as $P_R(t)$ for the right foot and $P_L(t)$ for the left foot.

The algorithm is straightforward. It begins by determining which foot will be in support during the next swing phase. Then, it computes the trajectory based on equations 4.7, 4.8, and 4.9, and repeats the process until the algorithm completes for each step.



(a) Time domain **(b) In the sagittal plane**

Figure 4.4: Swing foot trajectory in time and spacial domain. Theses trajectories are generated by taking $\delta = 0.2$, $t_{\text{ver}} = 0.1T_{\text{step}}$ and $h_{\text{step}} = 0.1$

4.4 ZMP Trajectory Generator

This section presents a method for computing the reference ZMP trajectory, ensuring that the ZMP lies within the support polygon as described in [24].

4.4.1 Working principle

To achieve stable walking during the swing phase (or single support phase), it is necessary for the ZMP to remain within the support foot. The position of each foot throughout the path is known thanks to the

Foot Planner. Therefore, the ZMP during the swing phase is defined as the location of the supporting foot.

During the double support phase, the support polygon is determined as the convex hull of the supporting foot area. In this phase, the robot shifts its swing foot to the support foot, simultaneously moving its CoM to the new support foot. Consequently, the ZMP is also adjusted to ensure stable walking. Various approaches can be employed to connect the reference ZMP and facilitate the transition between the swing and stance phases. Considering that the ZMP trajectory will serve as a reference signal for the *Preview Controller*, it is beneficial to smooth out the transition between two reference values. In the double support phase, the ZMP is interpolated using a cubic polynomial function with zero velocities at the beginning and end of the transition phase. The cubic interpolation is defined as:

$$p_{\text{ref}}(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 = \begin{bmatrix} a_{x,0} \\ a_{y,0} \end{bmatrix} + \begin{bmatrix} a_{x,1} \\ a_{y,1} \end{bmatrix} t + \begin{bmatrix} a_{x,2} \\ a_{y,2} \end{bmatrix} t^2 + \begin{bmatrix} a_{x,3} \\ a_{y,3} \end{bmatrix} t^3 \tag{4.10}$$

Here, p_{ref} represents the x and y coordinates of the reference ZMP, a_i denotes the polynomial coefficients, and t is the time variable. To ensure a smooth transition, the following conditions are imposed:

$$p_{\text{ref}}(t_s) = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad p_{\text{ref}}(t_f) = \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} \qquad \frac{\partial p_{\text{ref}}}{\partial t}(t_s) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \frac{\partial p_{\text{ref}}}{\partial t}(t_f) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In the above equations, t_s and t_f represent the starting and final times of the double support phase, respectively. The coefficients can be obtained by solving the following system of equations:

$$\begin{bmatrix} 1 & t_s & t_s^2 & t_s^3 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_s & 3t_s^2 \\ 0 & 1 & 2t_f & 3t_s^2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} p_{\text{ref}}(t_s) \\ p_{\text{ref}}(t_f) \\ 0 \\ 0 \end{bmatrix} \tag{4.11}$$

4.4.2 Implementation of the ZMP Trajectory Generator

The flowchart in Figure 4.5 illustrates the process of this block. Initially, the algorithm determines which foot serves as the support foot. Based on this information, the next ZMP location is computed using the position of the foot for the next step. Depending on the walking phase, the reference value is calculated either through cubic interpolation or directly using the next ZMP value. This process is repeated for each step required to complete the path.

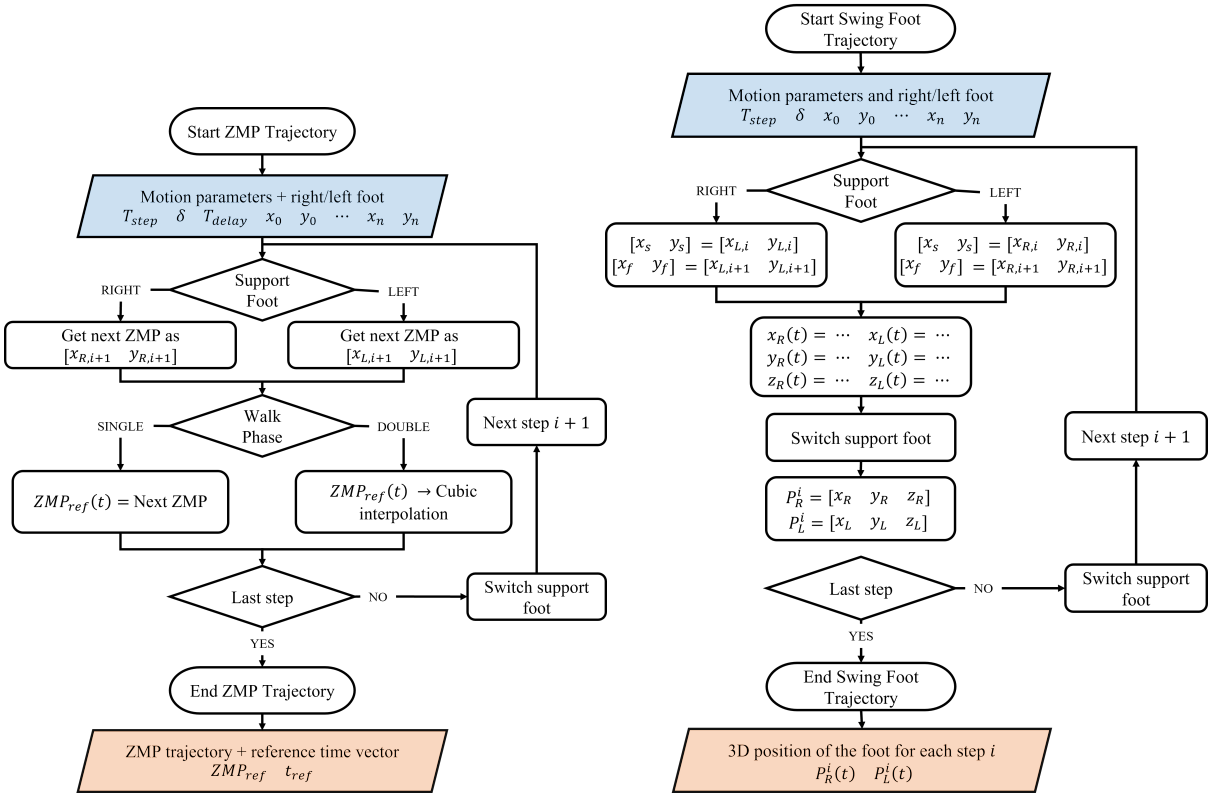
4.5 CoM Trajectory Generator

The *CoM Trajectory Generator* module produces a reference ZMP trajectory, which is then utilised by a controller to generate a CoM trajectory. This block converts the stable ZMP trajectory into the CoM trajectory required by the *Inverse Kinematics* module.

4.5.1 Working Principle

The CoM Trajectory Generator primarily employs the Preview controller, as described in Section 3.3, applied to the cart-table model. Using this model, the CoM trajectory in the x and y directions can be computed, as explained in Section 3.4. In accordance with Kajita *et al.*'s work [10], the CoM's vertical component (z axis) is assumed to remain at a constant height, denoted as z_c.

However, during the initial walking phase, the robot needs to lower its height to avoid entering a singularity configuration while swinging its legs. In this work, we adopt a cubic equation as the height



(a) ZMP Trajectory (main) **(b) Swing Foot Trajectory (main)**

Figure 4.5: Flowchart: ZMP Trajectory and Swing Foot Trajectory. The output is shown in a red box and the input in a blue box.

trajectory, ensuring zero velocities at the beginning and end of the initial walking phase. The trajectory of the hip height $z_h(t)$ is defined as follows:

$$z_h(t) = \begin{cases} a_0 + a_1 t + a_2 t^2 + a_3 t^3 & \text{if } t \leq t_{\text{delay}} \\ L_1 + L_2 + d_1 + d_4 - \Delta z & \text{else} \end{cases} \tag{4.12}$$

Here, a_i are polynomial coefficients that ensure continuity and allow zero velocity at $t = 0$ and $t = t_{\text{delay}}$. t_{delay} represents the duration of the initial walking phase, and Δz is the variation in hip height between the initial and final heights. Notably, L_1 denotes the length of the robot's thigh, L_2 represents the length of the robot's leg, d_1 is the distance between the hip joint and the centre of the hip body projected onto the z-axis, and d_4 is the distance between the ankle joint and the ground. This relationship holds under the assumption that $\alpha = 0$, where α denotes the body orientation.

The parameter Δz is selected to ensure a specific maximum step length $L_{\max} = L_{\max}(\Delta z)$ can be achieved without encountering any singular configurations during the swing phase.

In Kajita *et al.*'s work [10], a cart-table model of a robot with massless legs was proposed, which may not be entirely accurate for our case. Since our biped is primarily modelled as having only two legs, neglecting their dynamics would not be appropriate. The aforementioned hypothesis was utilised to simplify the robot's model, considering the CoM of the robot as the position of its pelvis ($z_h = z_c \forall t$). However, in our implementation, we distinguish between these two measurements. On one side, we have the z-component of the robot's CoM, denoted as z_c, and on the other side, we have the height of the hip body z_h with respect to the ground. In other words, we model the robot as a cart-table model, but the mass is located somewhere between the ground and the hip, rather than at the hip itself. Figure 4.6 illustrates our implementation where we can see the robot's configuration at $t = 0$ and $t = t_{\text{delay}}$,

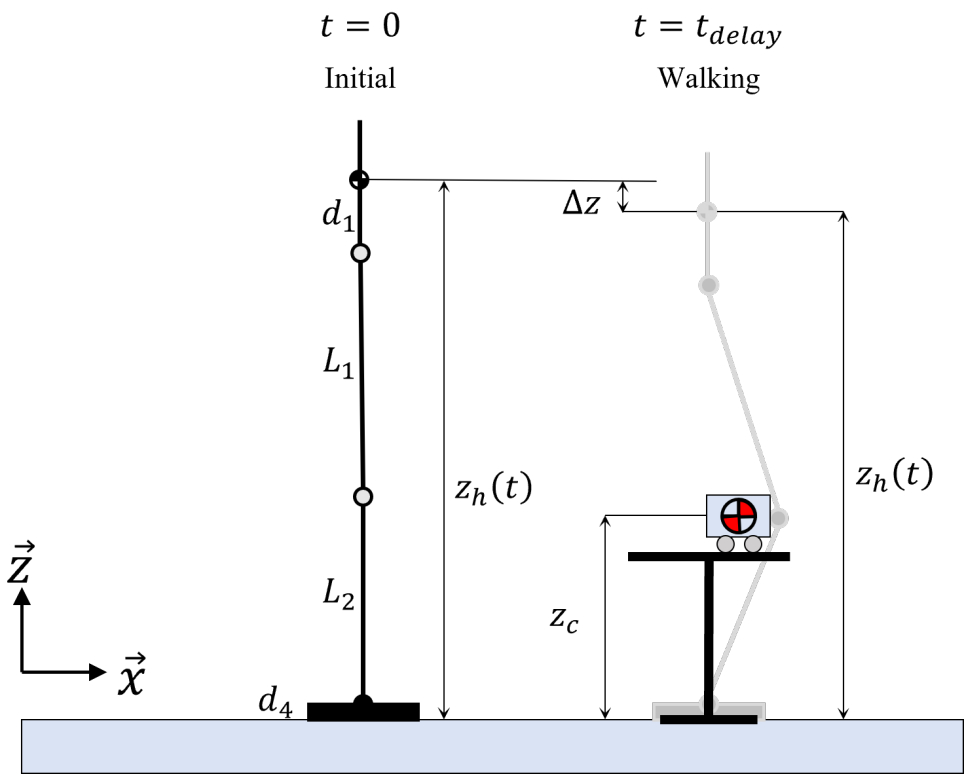


Figure 4.6: Difference between hip height z_h and CoM height z_c. Initial phase occur when $t < t_{\text{delay}}$ and the walking phase is define for $t \geq t_{\text{delay}}$. During initial phase the robot will lower his hip height by Δz.

where both legs are aligned in the sagittal plane.

In our implementation, the *Preview Control* module computes the trajectory of the robot's CoM, which is then transposed to the hip height trajectory. The simplest approach to transpose the cart's motion into hip motion is through a direct transposition:

$$x_h(t) = x_c(t) \qquad y_h(t) = y_c(t)$$

Here, $x_h(t)$ and $y_h(t)$ represent the xy coordinates of the hip with respect to the reference frame. The final component (the z-axis) is still determined by Equation 4.12. Therefore, the hip moves like a cart-table model, while exhibiting behaviour that respects the actual CoM.

4.5.2 Implementation of the CoM Trajectory Generator

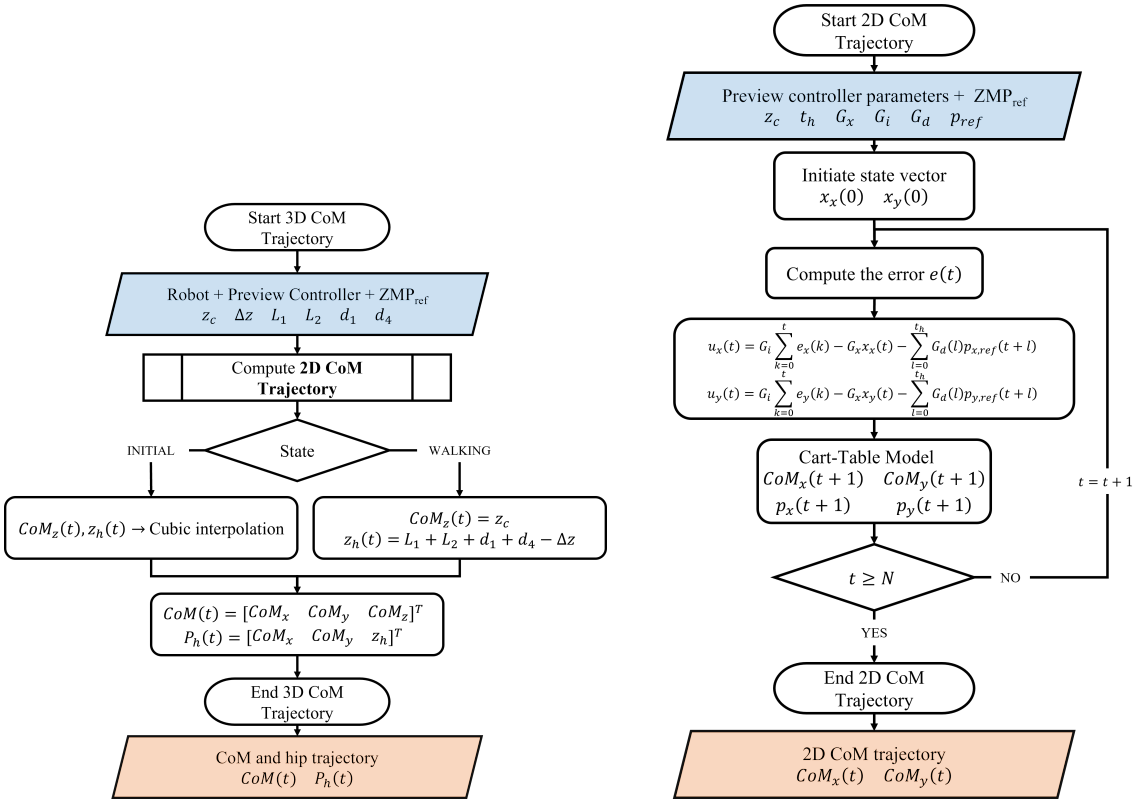
The flowchart depicted in Figure 4.7 outlines the implementation of the CoM Trajectory Generator. Essentially, the algorithm computes the CoM trajectory in the x and y axes using the preview controller, as explained in Section 3.4. Subsequently, the z component is determined based on whether the robot is in the initial walking phase or the regular walking procedure.

4.6 Inverse Kinematics

The *Inverse Kinematics* block allows us to convert a trajectory from the workspace (in xyz coordinates) into joint space (in q coordinates).

4.6.1 Working principle

The inverse kinematics problem for our bipedal robot has already been solved, as discussed in section 2.3.2. However, this solution assumes that the hip joint is not moving. To account for the forward



(a) 3D CoM Trajectory Generator (main) **(b) 2D CoM Trajectory Generator function**

Figure 4.7: CoM Trajectory Flowchart: 3D and 2D CoM Trajectory Generator. The output is shown in a red box and the input in a blue box.

motion of the robot, we need to convert the foot motion into a local frame where the hip joint is stationary. The conversion from the global coordinate frame to the local frame (hip frame) is achieved using a homogeneous transformation:

$$H_h^{GC} = \begin{bmatrix} R_\theta & P_h \\ 0 & 1 \end{bmatrix} \tag{4.13}$$

Here, P_h represents the coordinates of the hip in the global frame, and R_θ is the rotation matrix defined as:

$$R_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

The angle θ corresponds to the orientation of the robot in the xy plane, indicating the direction the robot is facing. Consequently, the local coordinates of the right/left foot are determined to use the following equations:

$$P_R^{\text{local}} = (H_h^{GC})^{-1} P_R \tag{4.15}$$

$$P_L^{\text{local}} = (H_h^{GC})^{-1} P_L \tag{4.16}$$

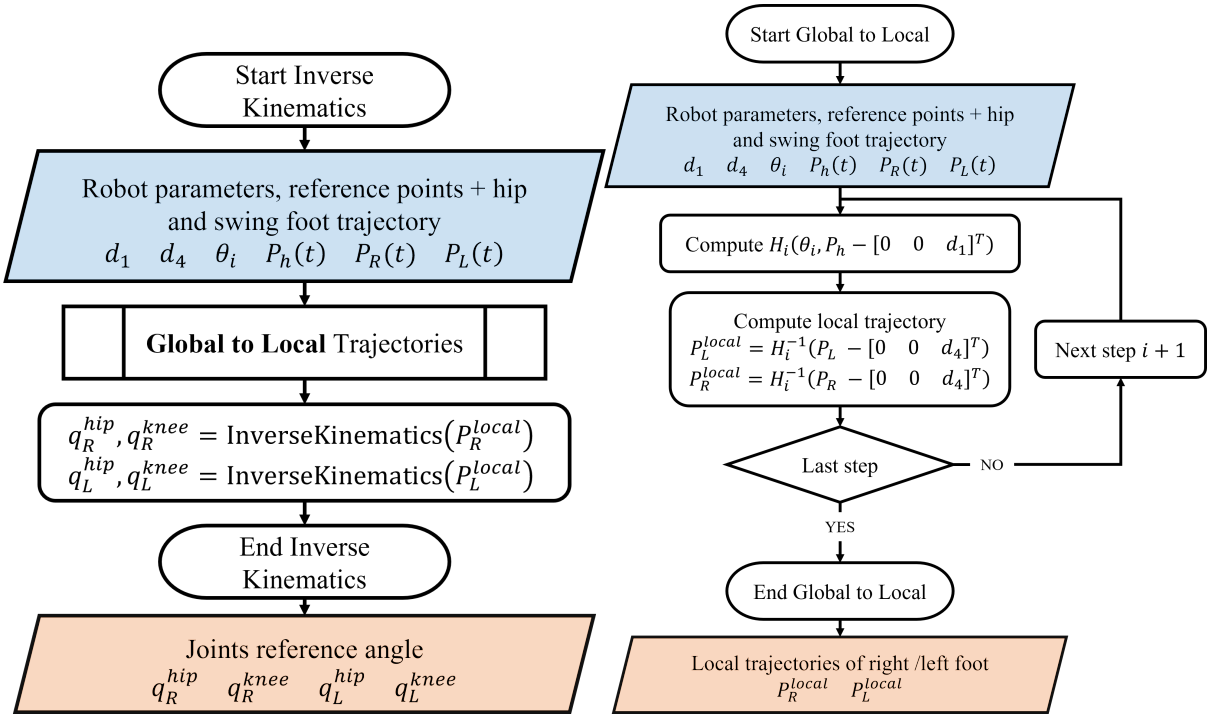
Since the robot can only move on the sagittal plane, the foot trajectory P^{local} is defined only along the local x and z axes, neglecting motion along the y axis. Finally, the inverse kinematics derived in section 2.3.2 provide the joint coordinates.

4.6.2 Implementation of the Inverse Kinematics

In the code, there is a small adjustment. Since the angle θ cannot be directly measured, we approximate it. The *Foot Planner* outputs the position and orientation of each foot. This orientation serves as an

approximation of the tangent line of the path and, therefore, a good approximation of the robot's orientation. Additionally, since the robot only moves on its sagittal plane (planar robot), the orientation remains constant throughout a step.

Another aspect to mention is related to the inverse kinematics. As shown in fig. 2.7, the inverse kinematics is defined for a specific frame. To obtain the hip joint position instead of the hip body position, we need to adjust the hip height P_h by d_1. The algorithm then proceeds as explained in the previous subsection and illustrated in fig. 4.8.



(a) Inverse Kinematics (main) **(b) Global to Local function**

Figure 4.8: Inverse Kinematics Flowchart: Inverse Kinematics and Global to Local. The output is shown in a red box and the input in a blue box.

4.7 Numerical results of the ZMP based controller

Let us consider a controller with the following parameters:

- The path is defined by the following equations:

$$x(t) = 0.01t \text{ m}, \qquad y(t) = 1.18 \text{ m}, \qquad t = 1, ..., 100$$

- Based on the actual robot, we can retrieve its geometric properties:

$$L_1 = 0.20125 \text{ m}, \quad L_2 = 0.172 \text{ m}, \quad d_1 = 0.04025 \text{ m}, \quad d_4 = 0.009 \text{ m}, \quad d = 0.052 \text{ m}$$

- Let's also choose the cart-table parameters:

$$g = 9.81 \text{ m/s}^2, \qquad T_s = 0.005 \text{ s}, \qquad z_c = 0.165 \text{ m}$$

- The parameters for the *Foot Planner* block are set as follows:

$$L_{\max} = 0.1 \text{ m}, \qquad \theta_{\max} = 12^\circ$$

- The *Swing Foot trajectory* requires the following parameters:

$$\delta = 0.05, \qquad h_{\text{step}} = 0.02 \text{ m}, \qquad T_{\text{ver}} = 0 \text{ s}, \qquad T_{\text{delay}} = 1 \text{ s}$$

- The initial ZMP needed in the *ZMP Trajectory Generator* is defined as:

$$p_0 = \begin{bmatrix} x(0) & y(0) \end{bmatrix}^T$$

- The *CoM Trajectory Generator* is defined using the parameters:

$$t_h = 0.99 \text{ s}, \qquad \Delta z = 0.0855 \text{ m}$$

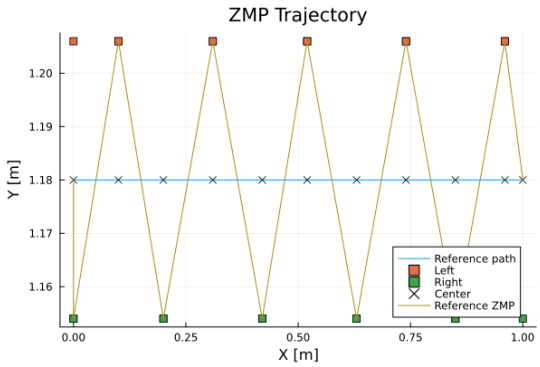
Using this controller, we obtain the results shown in fig. 4.9. If we change the path to a quadrant, the results are shown in fig. 4.10.

4.8 Conclusion

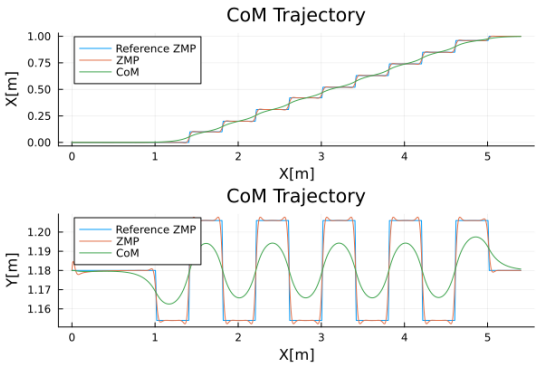
In this chapter, we have introduced our version of the **ZMP-based controller** for a bipedal robot. The controller consists of several functional blocks, including the *Foot Planner*, *Swing Foot Trajectory*, *ZMP Trajectory Generator*, *CoM Trajectory Generator*, and *Inverse Kinematics*.

The *Foot Planner* block determines the desired landing positions for the right and left foot along the chosen path. The *Swing Foot Trajectory* block calculates the 3D trajectory of the right and left foot in the global frame. The *ZMP Trajectory Generator* generates the reference trajectory for the ZMP to ensure stable walking. The *CoM Trajectory Generator* generates the trajectory for the Centre of Mass (CoM) and the hip motion based on a cart-table model. Finally, the *Inverse Kinematics* block converts the hip and foot trajectories from the workspace into joint space.

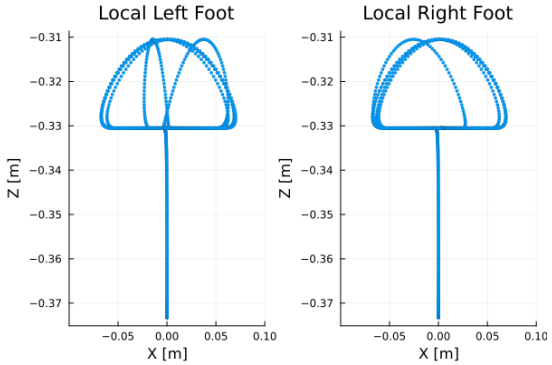
To validate the performance of our controller, we will conduct simulations before implementing it on the actual biped robot. This will allow us to assess its stability and effectiveness in achieving the desired walking behaviour.



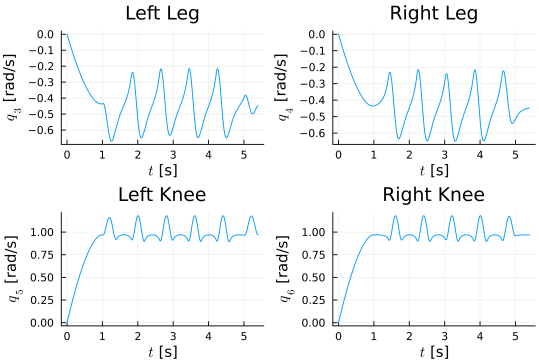
(a) Output from the foot position and ZMP trajectory generator



(b) Output from the COM trajectory generator

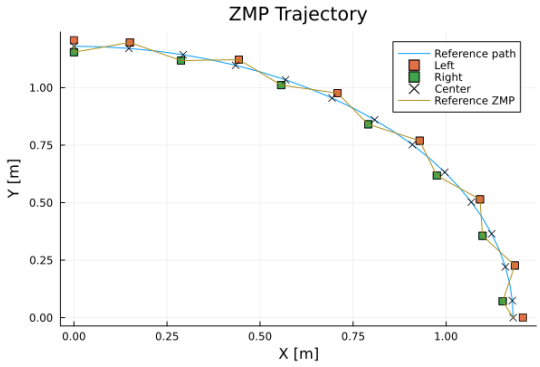


(c) Local trajectories for the left and right foot

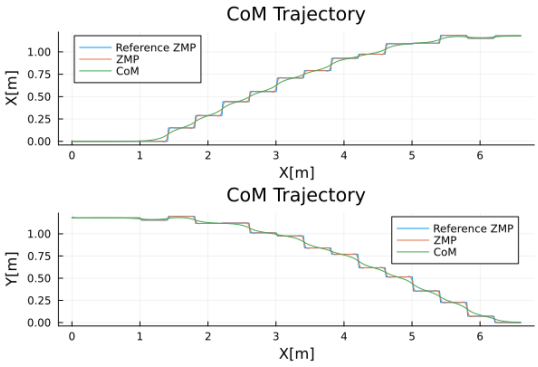


(d) Reference joint trajectories

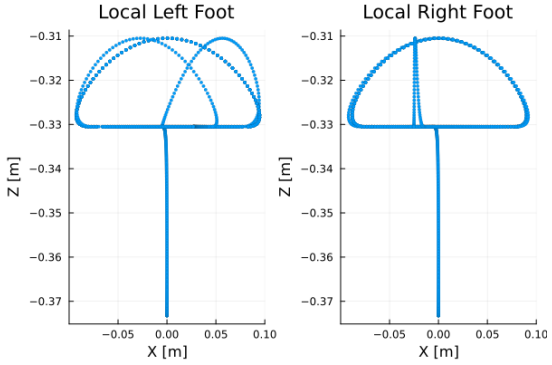
Figure 4.9: Result from the ZMP based controller for a straight path



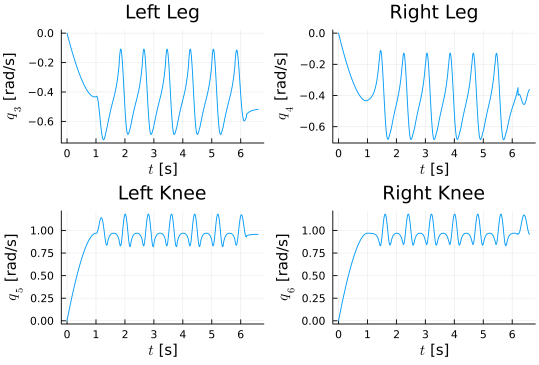
(a) Output from the foot position and ZMP trajectory generator



(b) Output from the COM trajectory generator



(c) Local trajectories for the left and right foot



(d) Reference joint trajectories

Figure 4.10: Result from the ZMP based controller for a quadrant path

Simulation using `RigidBodyDynamics.jl`

5

5.1 Introduction

In order to test the behaviour of the controller in a controlled environment and validate its performance, it is beneficial to develop a simulation environment. The biped robot model we have discussed so far, based on the cart-table model (described in section 3.2), contains inherent approximations due to the nonlinear nature of the robot. Therefore, including a simulation step in our approach is crucial before implementing the controller in physical experiments. The goals of our simulation are as follows:

1. Ensure that the cart-table model is accurate enough to maintain stability for the control approach.
2. Test and validate the ZMP based controller in a controlled environment.
3. Simulate a specific robot configuration to obtain specifications for that configuration.

This chapter will discuss the simulation process using the `RigidBodyDynamics.jl` library. Firstly, we will describe how the robot model is created in the simulator in section 5.2. Then, the simulation environment will be discussed in section 5.3. An optimisation of the ZMP controller will be developed and explained in section 5.4. Finally, we will analyse and discuss the simulation results in section 5.5.

5.2 Robot Model

To facilitate the model, let's introduce some assumptions. First, each body of the robot is approximated as a rectangular box i with a certain mass m_i, a length L_x^i, a width L_y^i, and a height L_z^i. The inertia of each body, in their respective frame, can be computed using the following formulas:

$$I_{xx}^i = m_i \frac{(L_y^i)^2 + (L_z^i)^2}{12} \qquad I_{yy}^i = m_i \frac{(L_x^i)^2 + (L_y^i)^2}{12} \qquad I_{zz}^i = m_i \frac{(L_x^i)^2 + (L_y^i)^2}{12}$$

Additionally, we will assume that each element has its mass distributed equally, which simplifies the computation of their centre of gravity. Furthermore, we assume that the boom will not impact the robot's dynamics, so the boom will have a negligible mass and inertia.

The real robot is shown in Figure 5.1a. As you can see, the robot doesn't have a flat foot. The latest version of the robot is equipped with flat feet, as shown in Figure 5.1b. Table 5.1 presents the mass and length values for each body, which are measured from the real robot. Since the robot is symmetric, the right and left elements are the same (except for the hip, which is unique). The inertia of each element is shown in Table 5.2. However, the inertia for the motors is defined in the datasheet. The inertia is converted to their centre of gravity using Huygens' formula. The inertia for each motor is given by:

$$I^{\text{Hip motor}} = \begin{bmatrix} 5.97 \times 10^{-5} & -8.14 \times 10^{-8} & 1.31 \times 10^{-7} \\ -8.14 \times 10^{-8} & 5.69 \times 10^{-5} & -1.97 \times 10^{-6} \\ 1.31 \times 10^{-7} & -1.97 \times 10^{-6} & 4.91 \times 10^{-5} \end{bmatrix}$$

$$I^{\text{Knee motor}} = \begin{bmatrix} 5.99 \times 10^{-5} & -7.47 \times 10^{-8} & 1.21 \times 10^{-7} \\ -7.47 \times 10^{-8} & 5.71 \times 10^{-5} & -1.93 \times 10^{-6} \\ 1.21 \times 10^{-7} & -1.93 \times 10^{-6} & 4.91 \times 10^{-5} \end{bmatrix}$$

Table 5.1: Parameters of each body in the biped robot

Body i	Mass m_i $[g]$	Length L_x^i $[mm]$	Width L_y^i $[mm]$	Height L_z^i $[mm]$
Hip	6.25	50	100	10
Hip motor	83	28.5	34	46.5
Thigh	118	20	20	154.75
Thigh motor	83	28.5	34	46.5
Leg	100	20	20	160.75
Foot	41	70	40	9

Table 5.2: Inertia of each body in the biped robot.

Body i	$I_{xx}^i[kg \cdot mm^2]$	$I_{yy}^i[kg \cdot mm^2]$	$I_{xx}^i[kg \cdot mm^2]$
Hip	5.26	1.35	6.51
Thigh	239.41	239.41	7.87
Leg	218.67	218.67	6.67
Foot	5.74	17.02	22.21

5.2.1 The URDF file

An **URDF** (Unified Robotics Description Format) is an XML application specifically used to model and visualise multi-body systems such as robots. In our case, the URDF file of the biped robot is introduced to the simulator as a physical and graphic model (fig. 5.2). The parameters of each body in the boom are set in table 5.3. As stated earlier, we neglect here the boom's dynamics because, it is more relevant to compare the predicted result (which have not modelled the boom in the cart-table model) with the simulation result.

Table 5.3: Parameters of each body in the boom

Body i	Mass m_i $[g]$	Length L_x^i $[mm]$	Width L_y^i $[mm]$	Height L_z^i $[mm]$	radius r
Base boom	0	/	/	600	25
Small boom link	0	10	10	5	/
Boom link	0	1095	/	/	6

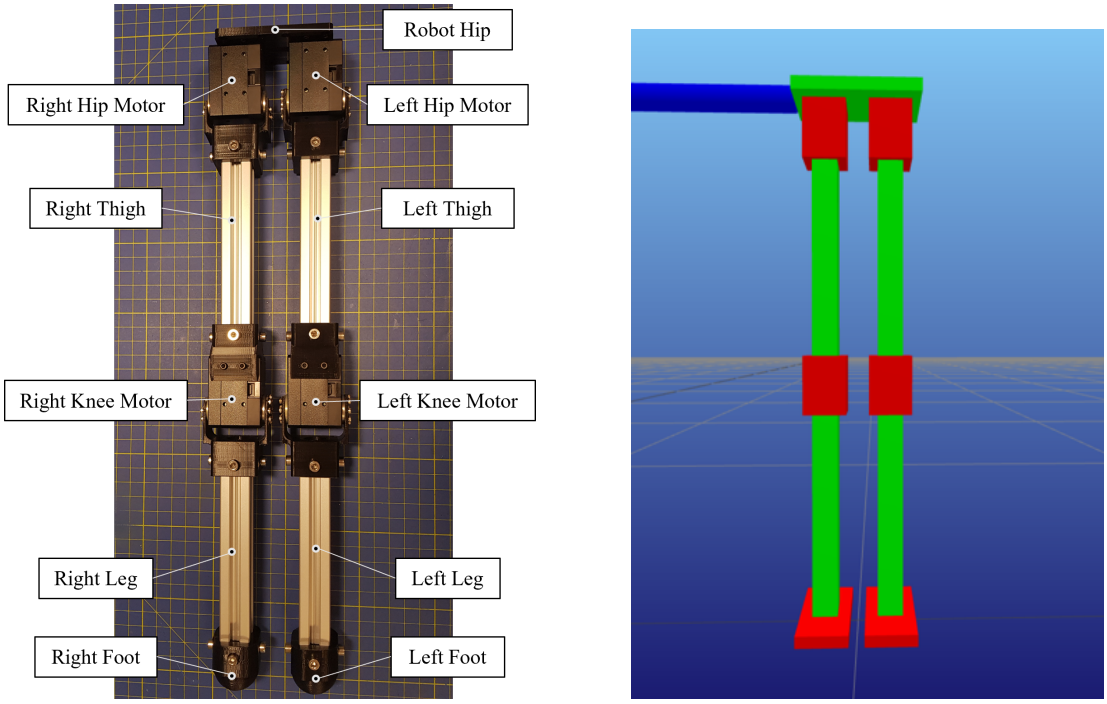
We can distinguish two main models for the biped based on how the boom behaves:

1. 2D Model: Also known as a planar model, the boom slides on a rail along the x-axis.
2. 3D Model: The boom is fixed, allowing only circular motion of the biped.

The joint configuration in the real model is described in table 5.4. Since we have stated that the hip orientation is fixed in section 2.1, joint 3, named 'boom link to hip link', is fixed for each model.

2D Model

In this model, we can describe the generalised coordinates as shown in table 5.5a. The table describes each degree of freedom for our biped robot. The boom can slide along the x-axis, allowing lateral motion in this axis. To suppress the inclination of the hip along the x-axis, we replace the x-revolute



(a) Real Biped robot **(b) URDF Model of the Biped robot**

Figure 5.1: Real Biped robot and simulated model

joint at joint 2 with a z-prismatic joint. Moreover, the joint between 'base boom to small boom link' is fixed to enable walking in a straight plane without rotating along the z-axis.

3D Model

In this model, we will constrain the hip to a circular motion, where the radius is set by the boom link. Each degree of freedom is described in table 5.5b. Thus, this model is the closest to the real model, except that the orientation of the hip is fixed.

5.2.2 Contact Model

In the simulation, each foot of the bipedal robot that is in contact with the ground is modelled using a contact force described in section 2.4.2. To ensure consistent results with Hertzian theory for contact spheres, the Hunt-Crossley model is employed with $n = \frac{3}{2}$. The contact forces are evenly distributed among five spherical contact points on each foot, as depicted in fig. 5.3.

Each contact point exerts both a force and a torque at the centre of gravity of the foot. By utilising this setup, the contact between the foot and the environment can be simulated accurately. Default values are used for the ground parameters, where:

$$k_n = 500 \times 10^2 \text{ N/m} \qquad \alpha = 0.2 \qquad \mu = 0.8 \qquad k_t = 20 \times 10^3 \text{ N/m} \qquad \lambda_t = 100 \text{ kg/m}$$

As a reminder from section 2.4.2:

- k_n is the spring constantly in the normal direction;
- α is the first order coefficient of the linerised coefficient of restitution and it's linked to the damping constant in normal direction with $\lambda_n = \frac{3}{2}\alpha k_n$;
- μ is the coefficient of friction between the foot and the ground;

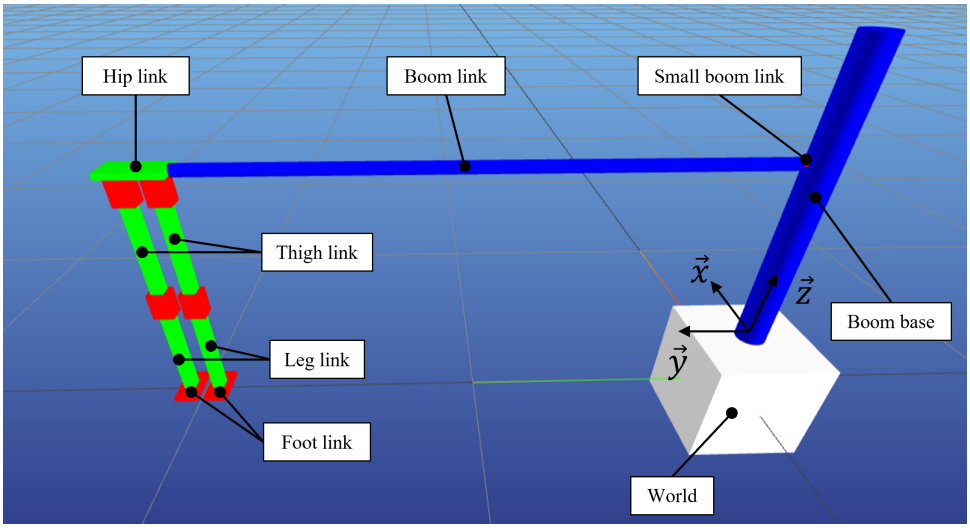


Figure 5.2: URDF model of the experimental setup.

Table 5.4: Description of the generalised coordinate for the real setup

Joint i	Name	Type
·	world to base boom	fixed
1	bass boom to small boom link	z-revolute
2	small boom link to boom link	x-revolute
3	boom link to hip link	y-revolute
4	hip link to left thigh link	y-revolute
5	hip link to right thigh link	y-revolute
6	left thigh link to left leg link	y-revolute
7	right thigh link to right leg link	y-revolute
8	left leg link to left foot link	y-revolute
9	right leg link to right foot link	y-revolute

- k_t is the spring constant in the tangent direction;
- λ_t is the damping constant in tangent direction.

5.3 Simulation environment

Once the robot is modelled, we can perform a simple simulation by tracking the reference joint angles, provided by the *Inverse Kinematics* block, for each joint q_3 to q_6. The simplest controller to perform this task is *Position Control*, which is widely used in industrial robots, especially in unconstrained systems.

5.3.1 Position Control

Position control is a type of control that allows us to regulate the position of the robot's end-effector to follow a desired trajectory in either the workspace or the joint space while rejecting disturbances. Several controllers can perform this task with high accuracy. Some examples include:

Table 5.5: Description of the URDF file

(a) Generalised coordinate for the 2D Biped robot

Joint i	Name	Type
1	World to base boom	x-prismatic
·	base boom to small boom link	fixed
2	small boom link to boom link	z-prismatic
·	boom link to hip link	fixed
3	hip link to left thigh link	y-revolute
4	hip link to right thigh link	y-revolute
5	left thigh link to left leg link	y-revolute
6	right thigh link to right leg link	y-revolute
7	left leg link to left foot link	y-revolute
8	right leg link to right foot link	y-revolute

(b) Generalised coordinate for the 3D Biped robot

Joint i	Name	Type
·	world to base boom	fixed
1	base boom to small boom link	z-revolute
2	small boom link to boom link	x-revolute
·	boom link to hip link	fixed
3	hip link to left thigh link	y-revolute
4	hip link to right thigh link	y-revolute
5	left thigh link to left leg link	y-revolute
6	right thigh link to right leg link	y-revolute
7	left leg link to left foot link	y-revolute
8	right leg link to right foot link	y-revolute

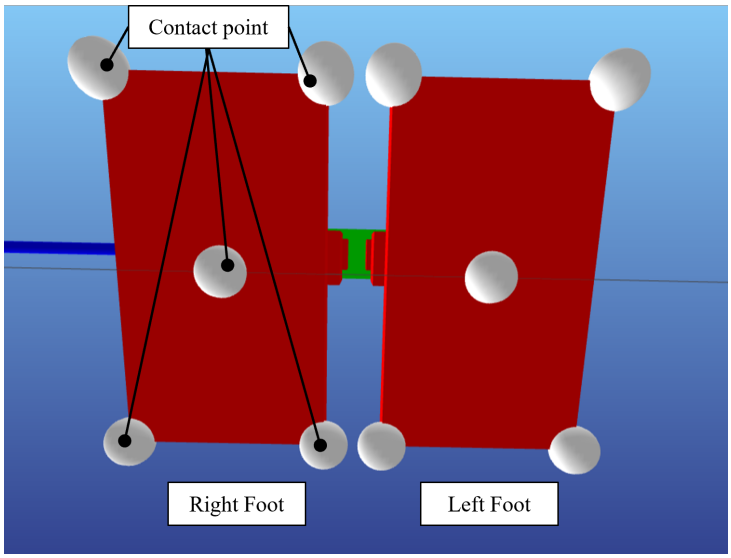


Figure 5.3: Distribution of the contact point on each foot

- Basic PD control:

$$\tau = K_p(q_{\text{ref}} - q_{\text{mes}}) - K_d\dot{q}_{\text{mes}}$$

Here, τ represents the torque vector provided to the actuator, $q$ref and q_{mes} are the reference and measured joint angle vectors, and $\dot{q}_{\text{mes}}$ is the measured joint velocity vector. The PD control has two tunable parameters: K_p and K_d, which form a diagonal matrix of proportional and derivative gains, respectively. This controller is relatively basic since it neglects the robot's dynamics and gravity, leading to potential steady-state errors at the output.

- PD with dynamic compensation:

$$y = \ddot{q}_{\text{ref}} + K_p(q_{\text{ref}} - q_{\text{mes}}) + K_d(\dot{q}_{\text{ref}} - \dot{q}_{\text{mes}})$$
$$\tau = \hat{M}(q)y + \hat{c}(q, \dot{q})$$

In this controller, y represents the new joint acceleration vector, $\dot{q}_{\text{ref}}$ and $\ddot{q}_{\text{ref}}$ are the reference joint velocity and acceleration, and $\hat{M}(q)$ and $\hat{c}(q, \dot{q})$ are the modified mass matrix and modified generalised force bias term, respectively, considering only the actuated joints. This controller takes into account the robot's dynamics, leading to improved control performance. In our case, we will set $\dot{q}_{\text{ref}} = 0$ and $\ddot{q}_{\text{ref}} = 0$.

The generalised force vector Q is updated by considering that joints 1 to 2 and 7 to 8 are not actuated, resulting in the following output:

$$Q = \begin{bmatrix} 0 & 0 & \tau & 0 & 0 \end{bmatrix}^T \tag{5.1}$$

Using a PD controller with dynamic compensation and manually tuned parameters ($K_p = 10000$ and $K_d = 100$) will yield reasonably accurate results. However, employing only position control has some limitations:

- Safety consideration: Since it is a basic position control, the robot will not consider any obstacles while attempting to reach the desired position;
- Interaction with the environment: In the case of position control, when the foot touches the ground, the leg doesn't react appropriately. Torque control is more suitable during the support foot phase.

For simplicity, we have implemented only position control for both the swing and support foot. In the future, it is highly recommended to implement a hybrid position-torque control. This approach would allow position control for the swing foot (unconstrained system) and force control for the support foot (constrained system).

5.3.2 Estimation of the ZMP

Estimating the ZMP during walking is crucial to ensure stability along the path. As mentioned in the previous chapter, the ZMP is defined as the point where both horizontal moments become zero. Therefore, Kajita *et al.* [12] demonstrates how to estimate the ZMP using force/torque sensors at each foot. Let us consider the model shown in Figure 5.4, where $\vec{p}_i$ represents the location of contact point i in the foot frame, $\vec{F}_i$ is the 3D contact force provided by the contact model, and $\vec{R}_L$ and $\vec{\tau}_L$ represent the resulting force and torque for the left leg.

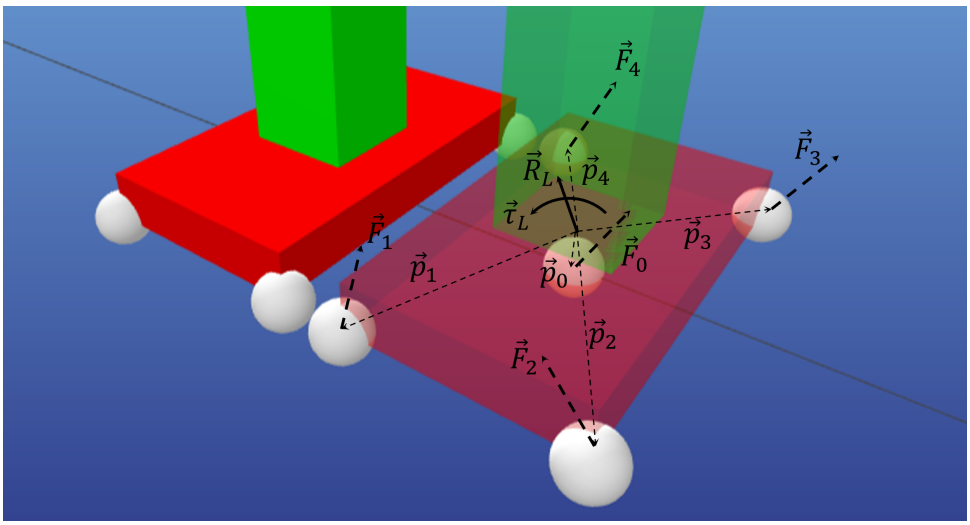


Figure 5.4: Definition of each variable for the ZMP measurement

The resulting force and torque on the left foot can be computed as:

$$\vec{R}_L = \sum_{i=0}^{N-1} \vec{F}_i \qquad \vec{\tau}_L = \sum_{i=0}^{N-1} \vec{p}_i \times \vec{F}_i$$

Here, N represents the amount of contact points on each foot. The resulting force and torque can be easily measured using a 6-axis force/torque sensor at each foot. The moment around the point

$\vec{p} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$ is given by:

$$\tau(p) = \sum_{i=0}^{N-1} (\vec{p}_i - \vec{p}) \times \vec{F}_i + \vec{\tau}_i \tag{5.2}$$

Hence, the position of the ZMP can be obtained by setting the x and y components of eq. (5.2) to zero and solving for p_x and p_y:

$$p_x = \frac{1}{\sum_{j=0}^{N-1} F_{jz}} \sum_{i=0}^{N-1} -\tau_{iy} - (p_{iz} - p_z)F_{ix} + p_{ix}F_{iz} \tag{5.3}$$

$$p_y = \frac{1}{\sum_{j=0}^{N-1} F_{jz}} \sum_{i=0}^{N-1} \tau_{ix} - (p_{iz} - p_z)F_{iy} + p_{iy}F_{iz} \tag{5.4}$$

Here, $\vec{\tau}_i$, $\vec{F}_i$, and $\vec{p}_i$ represent the torque, force, and position, respectively, at the contact point i. They can be expressed as:

$$\vec{\tau}_i = \begin{bmatrix} \tau_{ix} & \tau_{iy} & \tau_{iz} \end{bmatrix}^T$$
$$\vec{F}_i = \begin{bmatrix} F_{ix} & F_{iy} & F_{iz} \end{bmatrix}^T$$
$$\vec{p}_i = \begin{bmatrix} p_{ix} & p_{iy} & p_{iz} \end{bmatrix}^T$$

Equations 5.3 and 5.4 are fundamental equations for computing the ZMP for N contact points. We can distinguish two cases for estimating the ZMP:

- ZMP of each foot considering the reaction force between one of the feet and the ground.
  During the swing phase, the ZMP can be computed using the interaction of the reaction force between the support foot and the ground. Mathematically, for the left foot:

$$p_{Lx} = \frac{-\tau_{Ly} - R_{Lx}d}{R_{Lz}} \tag{5.5}$$

$$p_{Ly} = \frac{\tau_{Lx} - R_{Ly}d}{R_{Lz}} \tag{5.6}$$

  Here, d represents the distance between the foot's frame and the ground, and $\vec{p}_L$, $\vec{\tau}_L$, and $\vec{R}_L$ are defined as:

$$\vec{p}_L = \begin{bmatrix} p_{Lx} & p_{Ly} & p_{Lz} \end{bmatrix}^T$$
$$\vec{\tau}_L = \begin{bmatrix} \tau_{Lx} & \tau_{Ly} & \tau_{Lz} \end{bmatrix}^T$$
$$\vec{R}_L = \begin{bmatrix} R_{Lx} & R_{Ly} & R_{Lz} \end{bmatrix}^T$$

  It should be noted that eq. (5.5) and eq. (5.6) can be directly computed from eq. (5.3) and eq. (5.4) by considering an equivalent single contact point located at the Centre of Gravity (CoG) of the foot, which measures a resulting force $\vec{R}_L$ and a resulting torque $\vec{\tau}_L$ at this point.

- ZMP considering the reaction force between both feet and the ground.
  During the stance phase, both feet touch the ground. By using eq. (5.3) and eq. (5.4) for the right and left contact points, we can compute the ZMP as:

$$p_x = \frac{p_{Rx}F_{Rz} + p_{Lx}F_{Lz}}{F_{Rz} + F_{Lz}} \tag{5.7}$$

$$p_y = \frac{p_{Ry}F_{Rz} + p_{Ly}F_{Lz}}{F_{Rz} + F_{Lz}} \tag{5.8}$$

In summary, eq. (5.7) and eq. (5.8) will give the same result regardless of the walking phase. During the single support phase, the reaction force should be zero, yielding the same equations as in eq. (5.5) and eq. (5.6).

5.4 Optimisation of the ZMP Based Controller

In Chapter 4, a ZMP-based controller was developed to ensure stable walking of a bipedal robot. However, the controller's parameters were not properly set. This section aims to optimise the controller's parameters to achieve specific objectives.

5.4.1 Problem Formulation

The optimisation problem involves finding the optimal values for the controller's parameters, such as z_c, Δz, $L_{\max}$, h_{step}, T_{step}, T_{ver}, t_h, and δ. These parameters control various aspects of the walking motion:

- z_c corresponds to the height of the cart in the cart-table model (in metres);
- Δz as the variation in height during the initial walking process (in metres);
- $L_{\max}$ is the maximum step length (in metres);
- h_{step} represents the height of each step (in metres);
- T_{step} as the step period (in seconds);
- T_{ver} reflects the duration of the vertical motion during the swing phase (in seconds);
- t_h is the preview time used to foresee the future reference signal (in seconds);
- δ represents the ratio between the double support duration and the step period T_{step}.

Moreover, there are also some fixed design parameters, including:

- L_1, L_2, d_1, and d_4, which are geometric aspects of the robot;
- T_s, the sampling time of the controller;
- T_{delay}, the initial delay of the controller.

Let us define a candidate x that characterises a *ZMP-based controller* as a vector:

$$x = \begin{bmatrix} z_c & \Delta z & L_{\max} & h_{\text{step}} & T_{\text{step}} & T_{\text{ver}} & t_h & \delta \end{bmatrix}^T \tag{5.9}$$

To optimise the walking motion, we define three objective functions and perform a **multi-objective optimisation** to find the best candidate x (or set of controller parameters). Based on [4], we can formulate the objective functions as follows:

1. **Maximise the Average Walking Speed**:
   This objective aims to maximise the average speed of the hip $\dot{P}h(t)$ over one step:

$$f_1(x) = \arg\max_x \frac{1}{T_{\text{step}}} \int_0^{T_{\text{step}}} \dot{P}_h(t) dt \tag{5.10}$$

2. **Minimise the Energy Cost**:
   This objective aims to minimise the integral of the absolute value of the work provided by each actuator over one metre of displacement:

$$f_2(x) = \arg\min_x \frac{1}{L_{\max}} \int_0^{T_{\text{step}}} |\tau_u^T(t)| \cdot |\dot{q}_u(t)| dt \tag{5.11}$$

   Where $\tau_u^T(t)$ is the input torque vector for each actuator and $\dot{q}_u(t)$ is the measured joint speed.

3. **Minimise the Torque Cost**:
   This objective aims to minimise the integral of the squared input torque per metre of displacement:

$$f_3(x) = \arg\min_x \frac{1}{L_{\max}} \int_0^{T_{\text{step}}} \tau_u^T(t) \cdot \tau_u(t) dt \tag{5.12}$$

Because the walking pattern is mostly cyclic, it is easier to define these objective functions for one step. Moreover, each objective function, especially for Equation 5.11 and Equation 5.12, must be normalised to enable comparison. For example, candidate x_1 may have a larger step length, requiring more torque to take a larger step compared to candidate x_2, which has a smaller step length.

The maximisation of Equation 5.10 allows for defining the fastest displacement of the robot for a given set of parameters.

The minimisation of Equation 5.11 aims to reduce the variation of mechanical energy in the system. Since the bipedal robot is not equipped with regenerative braking, it is assumed that negative work produced by slowing down an actuator cannot be used by another one or during the acceleration phase. Therefore, the negative work has to be produced by the actuators, and the absolute value of the work is considered.

The minimisation of Equation 5.12 aims to minimise the losses in the actuators. In any electrical motor, neglecting mechanical losses for one cycle of walk, most losses are due to the Joule effect. Therefore, minimising the torque will also reduce the overall current through the motors.

5.4.2 Constraints formulation

5.4.2.1 Constraints

In the context of finding an optimal controller for a given robot with specific actuators, it is essential to consider constraint functions due to physical limitations and other factors. The following constraints are taken into account:

1. **Actuator's Torque limitation**:

$$\max \tau_i - \tau_{\max,\,i} \leq 0 \quad \forall i = 3, ..., 6 \tag{5.13}$$

   Here, τ_i represents the torque for joint i, and $\tau_{\max,\,i}$ is the maximum torque for that joint specified in the actuator's datasheet.

2. **Actuator's Speed limitation**:

$$\max \dot{q}_i - \omega_{\max,\,i} \leq 0 \quad \forall i = 3, ..., 6 \tag{5.14}$$

   In this constraint, $\dot{q}_i$ denotes the velocity of joint i, and $\omega_{\max,\,i}$ represents the maximum speed specified in the actuator's datasheet.

3. **Step length limitation**:

$$L_{\max} - \sqrt{\Delta z\big(2(L_1 + L_2) - \Delta z\big)} \leq 0 \tag{5.15}$$

   The step length constraint ensures that the robot's step length, represented by $L_{\max}$, remains within a specific range. Here, L_1 and L_2 denote the lengths of the thigh and leg, respectively.

4. **Practical limitation**:

$$T_{\text{ver}} = 0 \tag{5.16}$$

$$\delta T_{\text{step}} > 2T_s \tag{5.17}$$

The first constraint sets the vertical motion T_{ver} to zero, indicating that vertical motion is not considered. The second constraint ensures a double support duration with at least 3 time samples, where T_s represents the sampling time of the controller.

The torque and speed limitations (eq. (5.13) and eq. (5.14)) are imposed by the actuators and are derived from their datasheets. For the bipedal robot discussed here, the hip motor is equipped with Dynamixel XH430-W350-T, and the knee motor is equipped with Dynamixel XH430-W210-T. The maximum torque and no-load speed for each motor are specified in table 5.6.

Table 5.6: Maximal characteristic for each Dynamixel at an input voltage of 12 V

Motor	Stall torque [Nm]	No load speed [rad/s]
Hip	3.40	3.14
Knee	2.50	5.23

To derive the constraint eq. (5.15), let look at the fig. 5.5. We can distinguish 2 cases:

- Start Walking: the robot start to take his first step (red triangle);
- Walking: the robot has already made a few steps (blue triangle).

One can remark that the red triangle is the half of the blue triangle and thus these two cases are identical. Using Pythagorean theorem in the red triangle, we can write:

$$L_{\text{max}} = \sqrt{(L_1 + L_2)^2 - (L_1 + L_2 - \Delta z)^2} \tag{5.18}$$

Simplifying eq. (5.18) by expanding the square will result the constraint in eq. (5.15).

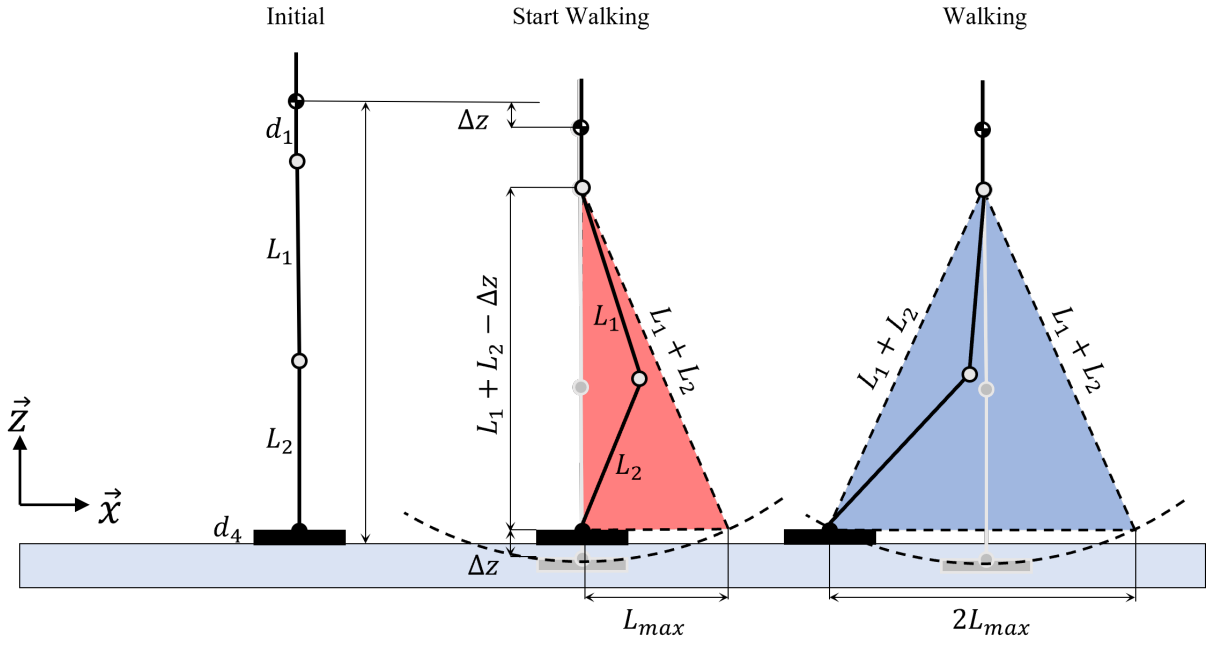


Figure 5.5: Schematic of the step length L_{max} constraint. The dashed circle define the maximal workspace of the leg. For both phase (start walking and during walking), the red triangle is the half of the blue triangle.

5.4.2.2 Non-Independent Input Parameters

Depending on how we define the candidate vector x, there might be some redundancy. In our example, we initially defined x as:

$$x = \begin{bmatrix} z_c & \Delta z & L_{\max} & h_{\text{step}} & T_{\text{step}} & T_{\text{ver}} & t_h & \delta \end{bmatrix}^T \tag{5.19}$$

However, some of these variables are dependent on each other, such as:

$$z_c = f(\Delta z) \qquad t_h = g(z_c)$$

Therefore, if we know the value of Δz, we can determine z_c, and if we know z_c, we can determine t_h. As a result, we can rewrite the new vector x as:

$$x = \begin{bmatrix} \Delta z & L_{\max} & h_{\text{step}} & T_{\text{step}} & T_{\text{ver}} & \delta \end{bmatrix}^T \tag{5.20}$$

The functions $f(\Delta z)$ and $g(z_c)$ can be determined as follows: z_c reflects the true position of the CoM of the robot, and t_h needs to be the optimal preview time in order to save computational power.

To compute z_c and t_h, the following steps can be followed:

1. Start by initialising a biped robot with a given vector x.
2. Set $z_c = z_{\max}$ and $t_h = 2T_{\text{delay}}$.
3. Compute the angle (using inverse kinematics) needed to lower the height of the hip by Δz.
4. Simulate the robot to reach the final height.
5. Measure the CoM and set z_c to its new value.
6. Compute the preview gain G_d based on the new Δz.
7. Determine the optimal value of t_h where G_d will have almost no contribution.
8. Repeat steps 3–7 if necessary.

In most cases, only one iteration is enough to obtain accurate results for z_c and t_h.

5.4.3 Optimisation for a discrete input-space

To solve the optimisation problem, various algorithms such as genetics, heuristics, etc., based on search techniques can be used. However, the controller has been constructed in such a way that each parameter cannot take any random value. In fact, each time variable has to be a multiple of the sampling time; otherwise, information loss may occur.

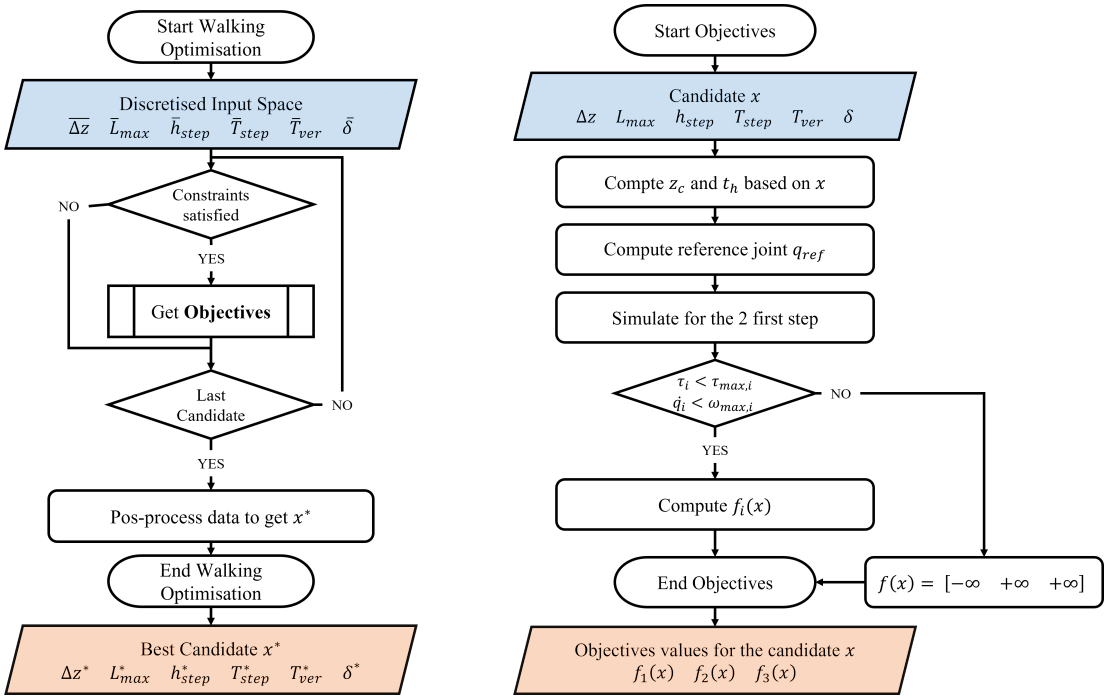
Thus, we decided to redefine the input space by discretising it and simulating each candidate in this reduced state space. By carefully choosing the final input space, we define it as shown in Table table 5.7. This results in a maximum of 2880 candidates, which is further reduced if they do not satisfy the constraints defined earlier. The flowchart of the optimisation algorithm is shown in Figure 5.6.

Once the input space is defined, the algorithm iterates through all the candidates in the predefined input space. First, it checks if the constraints are satisfied using equations 5.15 and 5.17. If the candidate is valid, it computes its objective values. Each objective value is obtained as shown in equations 5.10, 5.11, and 5.12, using simulation data from the first three steps. Only the second step is used to compute each objective value. However, if during the second step, the controller exceeds the torque or speed limits, the objective value is set to their least likely values.

The last step is the pre-processing step, which processes the data to retrieve the best candidate x^*. This step will be explained in more detail in the following section.

Table 5.7: Discritised input space for the Optimisation Algorithm

Variable [·]	Range	Step
$\bar{\Delta}z$ [m]	$[0.05; 0.3]$	0.05
$\bar{L}_{\max}$ [m]	$[0.05; \frac{L_1+L_2}{2}]$	0.05
$\bar{h}_{\text{step}}$ [m]	$[0.02; 0.05]$	0.01
$\bar{T}_{\text{step}}$ [s]	$[0.1; 1]$	0.1
$\bar{T}_{\text{ver}}$ [s]	0	/
$\bar{\delta}$ [·]	$[0.05; 0.2]$	0.05



(a) Walking Optimisation function (main) **(b) Objective function**

Figure 5.6: Flowchart: Optimisation algorithm. Input is in blue box and output are in red box.

5.5 Results

5.5.1 Optimisation result

We have presented a general method for tuning the ZMP-based controller through the solution of a multi-objective problem. In this section, we will discuss the results obtained from optimising the controller using a 2D bipedal robot model and position control, with manually tuned gains of $K_p = 10000$ and $K_d = 100$. The choice of the 2D model is arbitrary and simplifies the computation of the walking speed $f_1(x)$ by measuring the displacement along the x-prismatic joint q_1 of the boom, assuming negligible boom dynamics.

The final optimisation result is shown in Figure 5.7, where the optimisation process yielded 1249 valid candidates out of the initial 2880 candidates.

Using Figure 5.7 to identify an appropriate candidate with a good compromise between each objective value is challenging, given that multi-objective optimisation typically involves a Pareto optimal front.

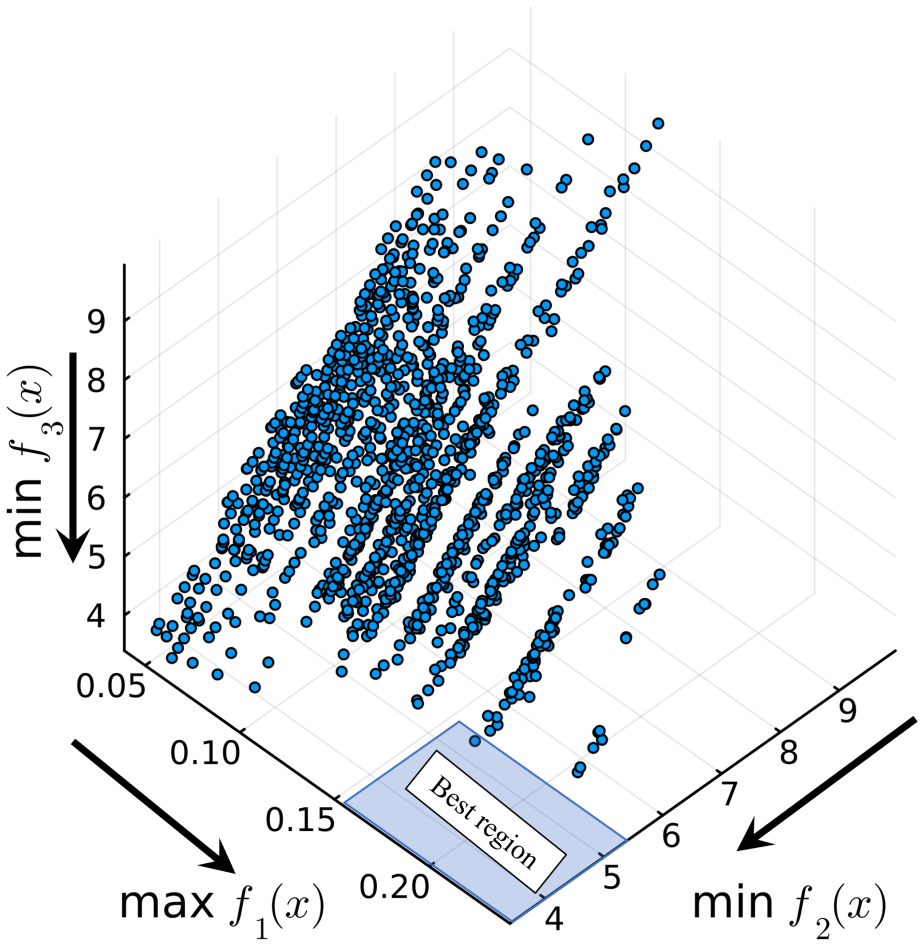


Figure 5.7: Optimisation result where we need to maximise the walking speed $f_1(x)$, minimise the energy cost $f_2(x)$ and the torque cost $f_3(x)$. The best region is define where each objective functions is optimised.

However, we propose a simpler approach to find a candidate that offers good performance. First, we introduce a small trick: sorting each candidate by their walking speed $f_1(x)$ and normalising each objective value $f_i(x)$ by the maximum value among all candidates:

$$F_1(x) = \frac{f_1(x)}{\max f_1(x)} \qquad F_2(x) = \frac{f_2(x)}{\max f_2(x)} \qquad F_3(x) = \frac{f_3(x)}{\max f_3(x)}$$

Here, $F_i(x)$ represents the normalised objective value, ranging between 0 and 1. By plotting the results (Figure 5.8), we can make the following observations:

- As $F_1(x)$ increases, $F_2(x)$ tends to stay constant. This phenomenon indicates that as the walking speed increases, the system experiences the same variation in mechanical energy for one step.
- As $F_1(x)$ increases, $F_3(x)$ tends to decrease. This behaviour suggests that as the walking speed increases, the torque required per step decreases. With a higher walking speed ($f_1(x) = \frac{L_{\max}}{T_{\text{step}}}$), the step duration T_{step} decreases. Consequently, each actuator needs to provide torque for a shorter time, resulting in lower overall current consumption.
- $F_1(x)$ exhibits a step-like behaviour, leading to a speed 'plateau' among the candidates.

Based on these observations, we can identify the best set of candidates that offer a good trade-off between the objective functions. This information is highlighted in the blue box in Figure 5.8. Our analysis identifies one potential candidate:

$$f_1(x) = 0.235 \text{ m/s} \qquad f_2(x) = 4.802 \text{ J/m} \qquad f_3(x) = 4.128 \text{ N}^2\text{m s}$$

and the controller is defined by :

$$\Delta z = 0.04225 \text{ m} \quad L_{\max} = 0.1 \text{ m} \quad h_{\text{step}} = 0.02 \text{ m} \quad T_{\text{step}} = 0.4 \text{ s} \quad T_{\text{ver}} = 0 \text{ s} \quad \delta = 0.05$$

5.5.2 Testing the Optimised Controller

The simulation results for the candidate are shown in Figure 5.9 for a straight path of 1 m. The result for a circular path is shown in Figure 5.10 which utilises the 3D model instead of the 2D one. From theses results, several observations can be made:

- The joint positions q in Figure 5.9a and 5.10a exhibit large errors during the double support phase, which can be attributed to the utilisation of position control.
- Both the joint velocities $\dot{q}$ (Figure 5.9b) and joint torque τ (Figure 5.9c) remain within the limitations set by the motors which is not the case for the circular motion in Figure 5.10b and Figure 5.10c);
- The small variation around z_c in Figure 5.9d and Figure 5.10d occurs when the robot lifts its leg and changes its CoM location.
- The ZMP (Figure 5.9e and Figure 5.10e) deviates from the reference value as the robot walks. In other words, the position error of the robot increases as it walks. This error can arise from various factors such as model errors with the cart-table model, PD position control, disturbances, etc.
- The touchdown event occurs slightly earlier than expected for both controllers (around 1.3 s instead of 1.4 s), as seen in Figure 5.9f and Figure 5.10f. The small oscillation after touchdown is due to the contact model.

Table table 5.8 provides the peak values of torque/speed for each controller and also the ZMP error ΔZMP at the end of walking. During circular motion, the robot undergoes a slight tilt in the frontal plane caused by the constraint imposed by the boom (refer to Figure 5.10). As a result, there are variations in the maximum torque and speed values between the right and left sides of the robot. In light of this observation, we suggest introducing a relaxed controller by increasing the value of T_{step} to 0.6 seconds, which allows the robot to achieve a walking speed of 0.163 m/s.

Table 5.8: Performance achieved of the ZMP based controller in open-loop. Note that $*$ indicate the relaxed controller where $T_{\text{step}} = 0.6$ s.

Motion	$\lvert\dot{q}^{\text{hip}}_{\max}\rvert$ [rad/s]	$\lvert\dot{q}^{\text{knee}}_{\max}\rvert$ [rad/s]	$\lvert\tau^{\text{hip}}_{\max}\rvert$ [Nm]	$\lvert\tau^{\text{knee}}_{\max}\rvert$ [Nm]	ΔZMP [m]
Straight	2.800	2.898	1.581	1.006	0.043
Circular	3.833	3.690	3.450	1.041	0.394
Straight*	2.016	2.191	1.452	1.018	0.045
Circular*	2.742	2.563	3.208	0.923	0.410

In the end, the actual ZMP based controller need some adjustments to improve his tracking behaviour and his robustness to disturbance. From actual result, the relaxed controller seems more appropriate for circular motion. Recall that the optimisation result optimise only for the 2D model. Next chapter will treat a way to improve the actual controller by feed-backing the measured ZMP into the *Preview Control* block.

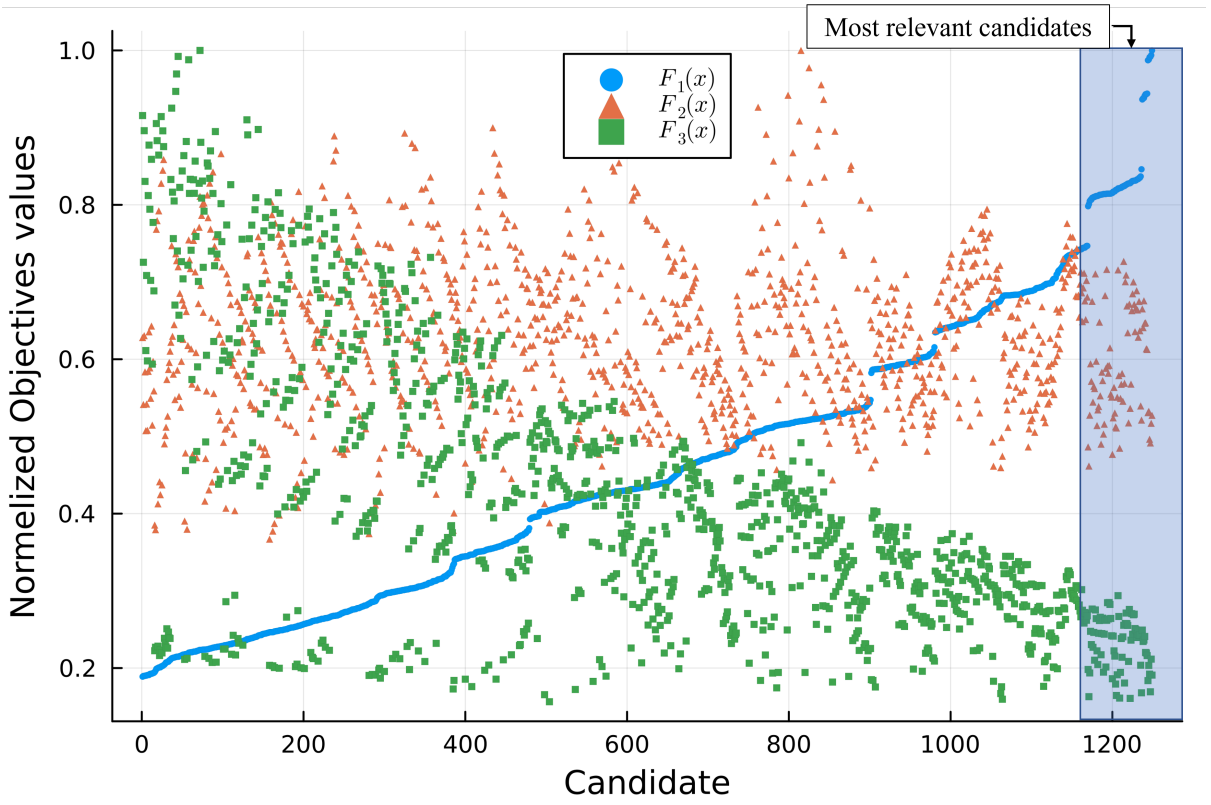


Figure 5.8: Optimisation result where objectives value are normalised and sorted by $f_1(x)$. The most relevant candidates are candidates where the walking speed are high enough.

5.6 Conclusion

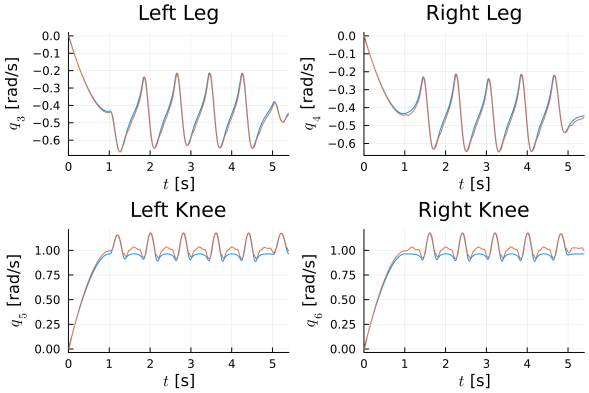
In this chapter, we presented the modelling and simulation of a bipedal robot using the URDF format and the RigidBodyDynamics.jl library. We developed a simplified 2D and 3D model of the bipedal robot and implemented a ZMP-based controller for walking.

We formulated an optimisation problem tuning the controller parameters based on multiple objectives, including maximising walking speed, minimising energy cost, and torque cost. Through the optimisation process, we obtained a set of candidate solutions that represent a trade-off between the different objectives.

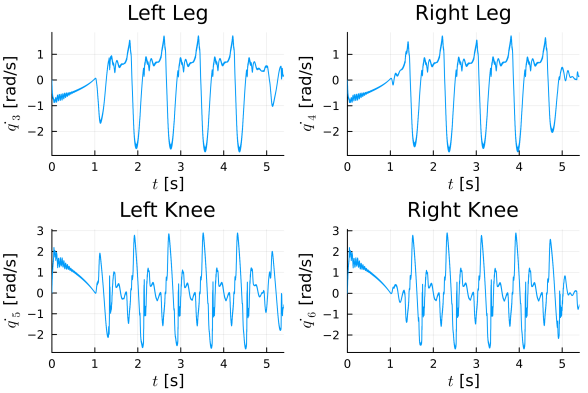
We then tested the best candidate from the optimisation results in simulations of a straight path and a circular path. The results showed that the controller was able to generate stable walking motions for straight path only. For circular motion, we introduced an relaxed controller which was able to respect the torque/speed limitation. However, some issues were observed, such as position errors during the double support phase and deviations of the ZMP from the reference value as the robot walked. These issues are likely due to the simplified model, the use of a position control, disturbances, and other factors.

To address these issues and improve the controller, we will explore the integration of feedback from measured ZMP into the Preview Control framework in the next chapter. This feedback-based approach can help improve tracking performance and enhance the robustness of the controller to disturbances.

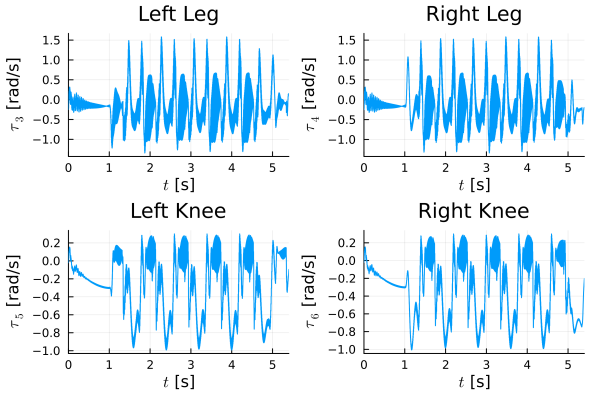
Overall, this chapter provides a foundation for the modelling, simulation, and optimisation of a bipedal robot. The results obtained so far demonstrate the potential of the ZMP-based controller and highlight areas for further improvement and research.



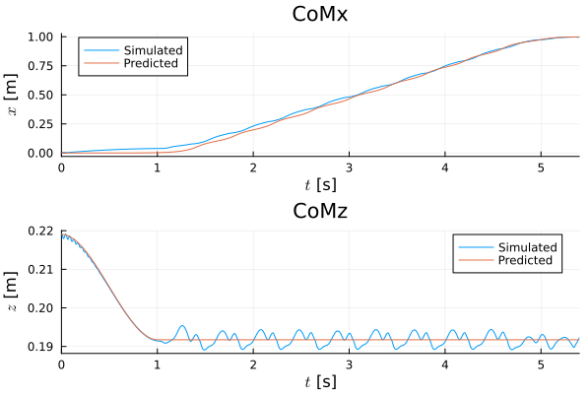
(a) Actuated joints position



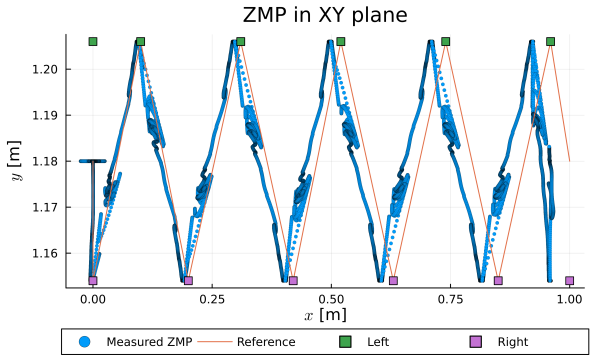
(b) Actuated joint velocity



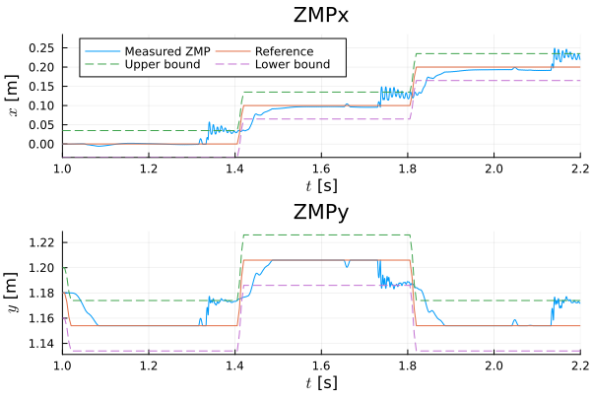
(c) Actuated joint torque



(d) CoM position of the robot

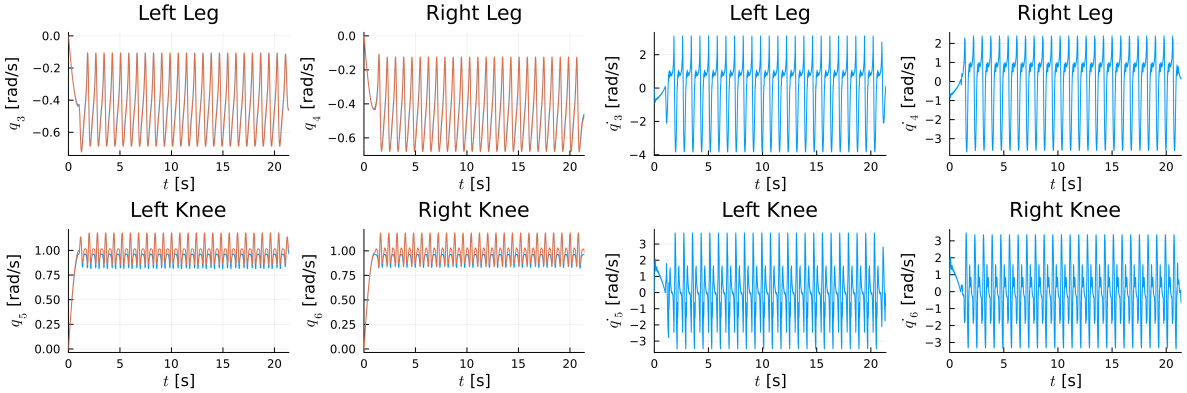


(e) ZMP location in the XY plane



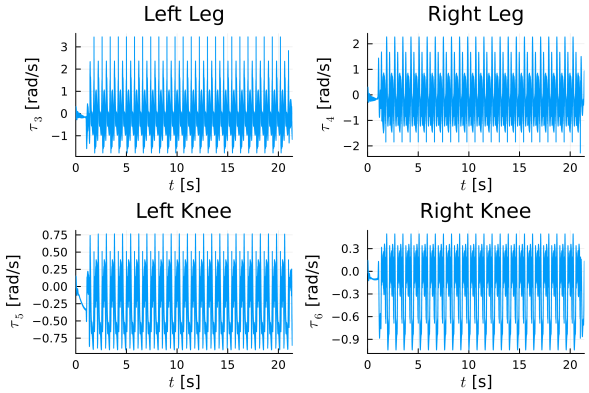
(f) x and y component of the ZMP for the first 2 steps

Figure 5.9: Simulation result for a straight path.

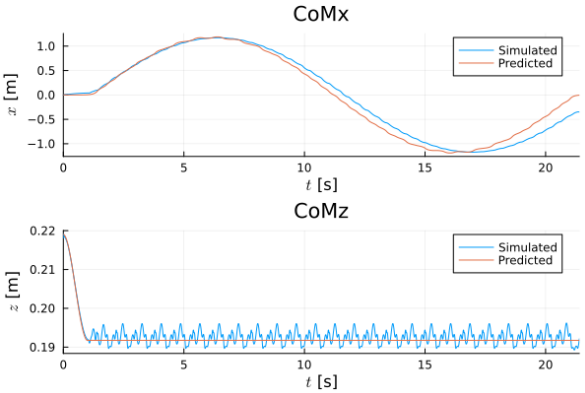


(a) Actuated joints position

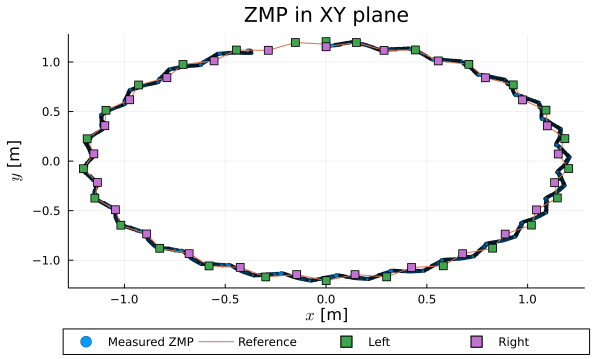
(b) Actuated joint velocity



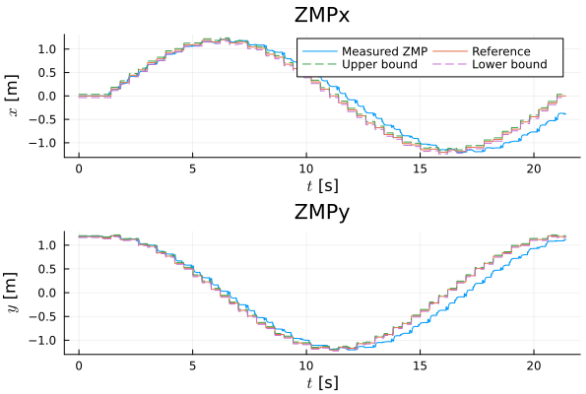
(c) Actuated joint torque



(d) CoM position of the robot



(e) ZMP location in the XY plane



(f) x and y component of the ZMP

Figure 5.10: Simulation results for circular motion

Closed-loop form of our ZMP based controller

6

6.1 Introduction

In the previous chapter (chapter 5), several issues were identified in our ZMP-based controller. Due to our design choices, there was a mismatch between the actual ZMP of the robot and the predicted ZMP. We hypothesized that these errors were attributed to our previous design choices, such as the cart-table model and position control.

This chapter aims to address and resolve these design issues by implementing a feedback loop in our controller.

Section 6.2 discusses the adaptation of our ZMP-based controller to incorporate ZMP feedback. We will implement a closed-loop system in our simulation environment. Lastly, in section 6.3, we will discuss future perspectives for further improvement.

6.2 Closed loop system

As described in Chapter 4, our previous version of the ZMP-based controller did not interact with the simulation environment in a closed-loop manner. In this improved version, we aim to implement a ZMP feedback system instead of an open-loop system. The basic diagram of the closed-loop form is shown in Figure 6.1, illustrating the interaction between the Preview Control and the simulation environment. The main loop consists of the Preview Control, inverse kinematics process, and the simulation environment where the position control is applied.

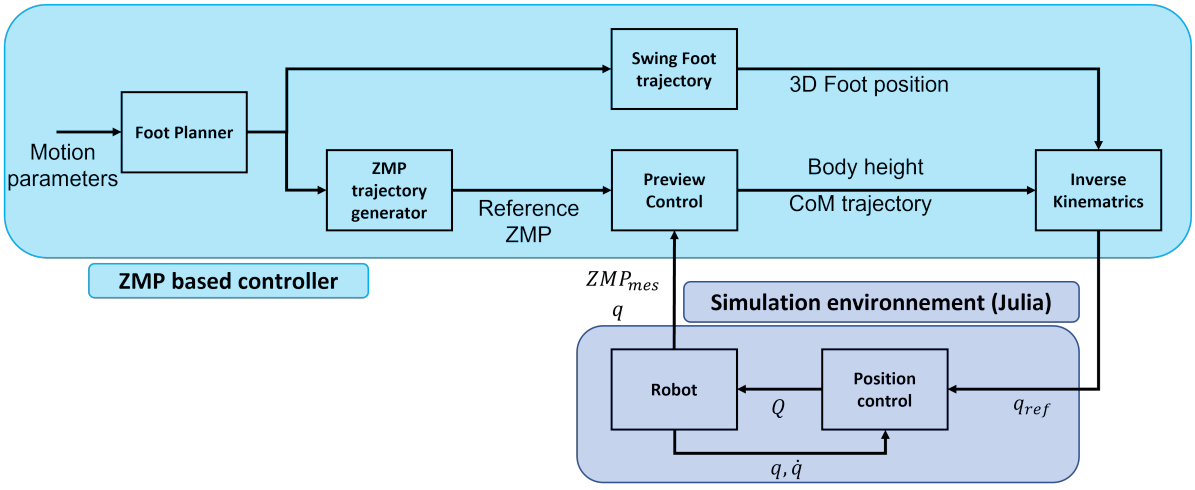


Figure 6.1: Closed loop form of our ZMP based controller. Here we estimate the actual ZMP of the robot in order to feed it back to the CoM Trajectpry Generator. Moreover, the joint position are measured to feed it back to the COM Trajectory Generator.

6.2.1 Dynamics Filter based on Preview Control

In Chapter 3 (Bipedal Walking), we introduced the Preview Control, which generates the CoM trajectory based on a cart-table model. However, this method does not guarantee the stability of the motion, as it cannot represent changes in the robot's body posture during walking.

To address this issue, Kajita introduced a structure called the **Dynamic Filter based on Preview Control** (Figure 6.2) in [12]. This structure utilizes a cart-table model as an error system to compensate for the actual ZMP error using preview control.

The inputs to the dynamic filter are the target ZMP (ZMP^{ref}) and the robot state, which includes the angles, velocities of all joints, and the Pelvis configuration. The structure first computes the ZMP error (ΔZMP) from these inputs and stores it in a FIFO buffer. The robot state is also stored in another FIFO buffer. By retrieving ΔZMP from the FIFO stack after a delay of preview time t_h, the future ZMP error is obtained. The preview control then computes the proper CoM trajectory compensation (ΔCoM) based on these future values of ZMP error. Moreover, the delayed robot state provides the actual hip position, which is corrected by ΔCoM.

Kajita refers to the system in Figure 6.2 as a **Dynamic Filter**, as it converts a given motion pattern into an improved one.

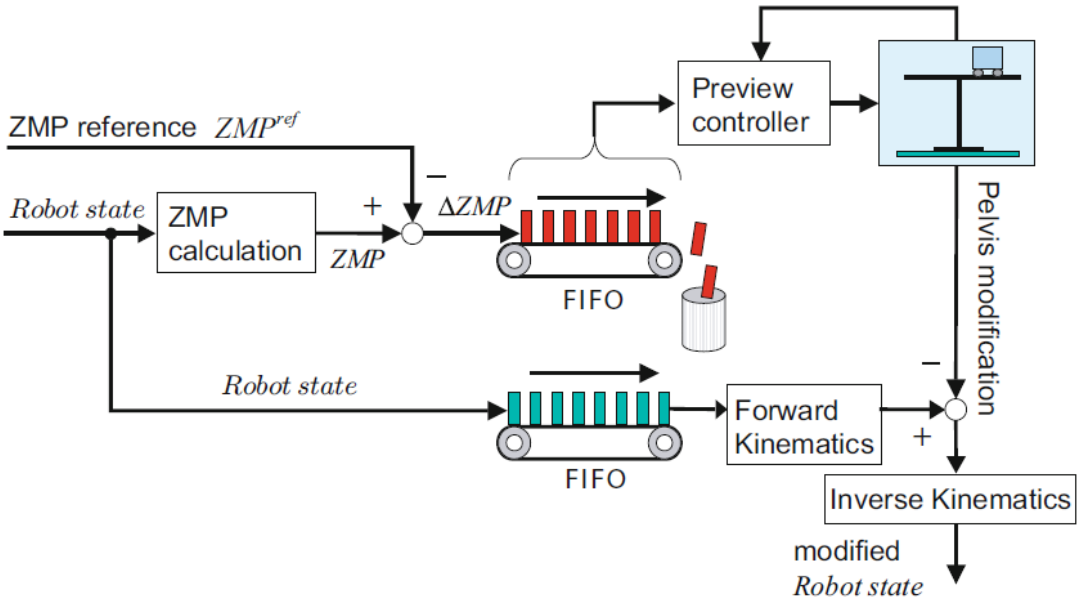


Figure 6.2: Dynamic filter based on Preview Control [12]

6.2.2 Integration in the simulation environment

Based on Kajita *et al.* initial structure [12], we have implemented a version suitable for our bipedal robot (Figure 6.3).

In our version, we also have two FIFO buffers to store the ZMP error and the actuated joint angles. However, for simplicity, let's assume that the robot will walk in a straight path. Therefore, we can assume that the ZMP needs to be compensated only in the walking direction (since our robot is planar).

The FIFO stack outputs a vector into the Preview Control, which provides the hip height compensation $\Delta P_h(t-h)$. Here, $P_h(t)$ represents the hip height trajectory, $t-h$ is the actual time, and h is the preview length. The first popped value from the stack is deleted. In the second FIFO stack, the output is used in the *Direct Kinematics* (presented in section 2.3.1), which provides the hip height at the actual time $t-h$. The remaining structure is the same as the *ZMP-based controller*, which essentially represents the simulation environment.

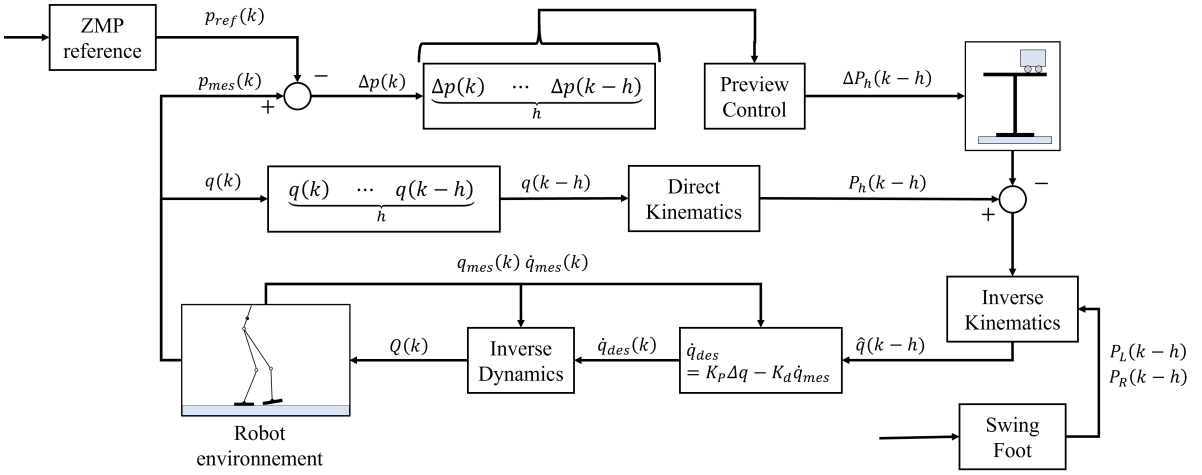


Figure 6.3: Our version of dynamic filter based on Preview Control. As in Kajita *et al.*'s book [12], we have two FIFO stacks for the ZMP error and the joint angles.

It is important to note that at the initial time, the robot needs to stack up the FIFO stacks until t reaches $t = h$. During this time, we set the robot to its initial configuration, where $q_u = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$.

However, our implemented structure does not react as expected. It appears that the hip height is not properly compensated or not adjusted at the right time, leading to instabilities in our control structure. Unfortunately, we have not yet identified the root cause of this issue.

6.3 Futures perspectives

In addition to the advancements made in our ZMP-based controller, there are several directions to further improve its capabilities for implementation on a real bipedal robot.

One important aspect is to enhance the stability of the closed-loop system. Currently, we have not successfully implemented a proper *dynamic filter* in our controller. Furthermore, the cart-table model we used is outdated (with the first paper dating back to 2003 [10]). More recent research has demonstrated reliable performance and stability using advanced control techniques such as hybrid zero dynamics ([5, 9, 20]), Lyapunov control ([2]) , or bio-inspired control ([1, 25]).

Another key aspect is to upgrade the current controller to reject disturbances such as external unwanted forces, uneven ground, unexpected obstacles, etc. For instance, upgrading the position control to impedance control would already enhance the overall performance. Additionally, in this study, the focus has primarily been on a controlled environment to simplify the control strategy. However, it is crucial to acknowledge that real-world scenarios often deviate from these ideal conditions. Therefore, it becomes imperative to develop a controller that exhibits robustness and can effectively handle such disturbances in practical applications.

In Section 4.1, we discussed some differences between our version and existing ZMP-based controllers on the market. We could incorporate *Active Balance Control* into our controller to relax the

assumptions used in modeling the robot.

Furthermore, the ultimate goal is to implement the controller on a real robot to benchmark its performance. In this study, the robot was not available for experimental measurements. Therefore, an interesting future direction would be to compare the results of real-world experiments with the simulation results presented in this work. The simulation engine can be used to redesign the real bipedal robot if necessary.

By exploring these future aspects, we can develop a more advanced controller and potentially refine the model of the bipedal walker to make it more accurate.

6.4 Conclusion

n this chapter, we explored a way to improve our ZMP-based controller by implementing a structure called *dynamic filter*, which was originally proposed by Kajita to compensate for ZMP errors.

We presented our own interpretation of the dynamic filter and attempted to integrate it into our controller within the simulation environment. However, we encountered stability issues and have not yet identified the cause.

Finally, we discussed future perspectives for further enhancing the ZMP-based controller, including improving stability, robustness, and performance, as well as conducting real-world experiments. Overall, this chapter serves as a starting point for future developments in this area.

Conclusion

7

The inverted pendulum model, particularly the Cart-Table model, is widely used for generating walking patterns in bipedal robots due to its simplicity, linear properties, and computational efficiency for real-time control.

In this study, we have explored the application of the Zero Moment Point (ZMP) stability criterion in bipedal walking, specifically in the context of the cart-table model. We have thoroughly investigated its capabilities and demonstrated that this model, using Preview Control, can generate appropriate Centre of Mass (CoM) trajectories corresponding to the desired reference ZMP trajectories, reflecting predefined walking tasks such as straight walks or circular motions.

Building upon existing literature on control methods in bipedal robots, we have proposed our own version of the ZMP-based controller by integrating various techniques available. This includes foot planner, swing foot control, reference ZMP calculation, CoM trajectory generation, and inverse kinematics. Extensive simulations using a dynamic bipedal robot model have been conducted to validate the effectiveness of the proposed controller. The evaluation of the controller's performance considers factors such as walking speed, energy efficiency, and trajectory tracking. The optimal controller demonstrates stable walking motion for both straight paths and circular trajectories, although it exhibits an increase in hip position error over time.

We have summarized the simulation results of the optimal configuration in Table 7.1. The open-loop simulation shows that the ZMP-based controller achieves a maximum walking speed of 0.235 m/s for straight motion and 0.163 m/s for a circular trajectory, while still respecting the limitations imposed by the actuators.

Table 7.1: Performance achieved of by the chosen ZMP based controller in open-loop where T_{step} denote the step period, L_{max} represent the maximal step length, v_{step} is the walking speed and $\Delta\text{ZMP}_{\text{max}}$ reflect the ZMP error at the end of walking.

Motion	T_{step} [m]	L_{max} [m]	v_{step} [m/s]	$\Delta\text{ZMP}_{\text{max}}$ [m]
Straight	0.4	0.1	0.235	0.043
Circular	0.6	0.1	0.163	0.410

To further improve the controller, we have implemented a closed-loop form of the ZMP-based controller using a Dynamic Filter. This new structure takes into account the error model and disturbances, compensating for ZMP errors. However, during our implementation, we have encountered stability issues with the control, which require further investigation.

This research opens up several future perspectives for exploration. Some interesting ideas to enhance the current controller include addressing the stability issues associated with the Dynamic Filter structure, simulating the resulting controller in various scenarios (such as disturbances, uneven terrain, and obstacles), and testing the controller on a physical bipedal robot. This list of ideas is not exhaustive and offers potential directions for future development.

Overall, this study contributes to the field of bipedal robotics by providing a comprehensive analysis and implementation of a ZMP-based controller for an eight-degree-of-freedom bipedal robot. The findings offer valuable insights into the design considerations and challenges involved in developing effective control strategies for bipedal locomotion.

Bibliography

[1] A. D. Ames. "Human-Inspired Control of Bipedal Walking Robots". In: *IEEE Transactions on Automatic Control* 59.5 (May 2014), pp. 1115–1130. ISSN: 0018-9286, 1558-2523, 2334-3303. DOI: 10.1109/TAC.2014.2299342. (Visited on 01/20/2023).

[2] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle. "Rapidly Exponentially Stabilizing Control Lyapunov Functions and Hybrid Zero Dynamics". In: *IEEE Transactions on Automatic Control* 59.4 (Apr. 2014), pp. 876–891. ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2014.2299335. (Visited on 01/20/2023).

[3] *ASIMO*. In: *Wikipédia*. Page Version ID: 200919921. Jan. 30, 2023. (Visited on 05/31/2023).

[4] C. Chevallereau and Y. Aoustin. "Optimal reference trajectories for walking and running of a biped robot". In: *Robotica* 19.5 (Sept. 2001), pp. 557–569. ISSN: 0263-5747, 1469-8668. DOI: 10.1017/S0263574701003307. (Visited on 04/13/2023).

[5] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E. R. Westervelt, C. Canudas-de-Wit, and J. W. Grizzle. "Designing, building, and controlling an experimental platform for the study of walking." In: *IEEE Control Systems Magazine* (2003).

[6] R. Featherstone. *Rigid body dynamics algorithms*. OCLC: ocn190774140. New York: Springer, 2008. 272 pp. ISBN: 978-0-387-74314-1 978-0-387-74315-8.

[7] R. Ferreira, N. Shafii, N. Lau, L. P. Reis, and A. Abdolmaleki. "Diagonal walk reference generator based on Fourier approximation of ZMP trajectory". In: *2013 13th International Conference on Autonomous Robot Systems*. 2013 13th International Conference on Autonomous Robot Systems (Robotica). Lisbon, Portugal: IEEE, Apr. 2013, pp. 1–6. ISBN: 978-1-4799-1247-6 978-1-4799-1246-9. DOI: 10.1109/Robotica.2013.6623534. (Visited on 12/20/2022).

[8] D. Gouaillier, C. Collette, and C. Kilner. "Omni-directional closed-loop walk for NAO". In: *2010 10th IEEE-RAS International Conference on Humanoid Robots*. 2010 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2010). Nashville, TN, USA: IEEE, Dec. 2010, pp. 448–454. ISBN: 978-1-4244-8688-5. DOI: 10.1109/ICHR.2010.5686291. (Visited on 12/20/2022).

[9] A. Hereid, C. M. Hubicki, E. A. Cousineau, J. W. Hurst, and A. D. Ames. "Hybrid zero dynamics based multiple shooting optimization with applications to robotic walking". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015 IEEE International Conference on Robotics and Automation (ICRA). Seattle, WA, USA: IEEE, May 2015, pp. 5734–5740. ISBN: 978-1-4799-6923-4. DOI: 10.1109/ICRA.2015.7140002. (Visited on 12/20/2022).

[10] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. "Biped walking pattern generation by using preview control of zero-moment point". In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. IEEE International Conference on Robotics and Automation. IEEE ICRA 2003. Taipei, Taiwan: IEEE, 2003, pp. 1620–1626. ISBN: 978-0-7803-7736-3. DOI: 10.1109/ROBOT.2003.1241826. (Visited on 12/20/2022).

[11] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. "The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation". In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*. RSJ/IEEE International Conference on Intelligent Robots and Systems. Vol. 1. Maui, HI, USA: IEEE, 2001, pp. 239–246. ISBN: 978-0-7803-6612-1. DOI: 10.1109/IROS.2001.973365. (Visited on 12/20/2022).

[12] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi. *Introduction to Humanoid Robotics*. Vol. 101. Springer Tracts in Advanced Robotics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. ISBN: 978-3-642-54535-1 978-3-642-54536-8. DOI: 10.1007/978-3-642-54536-8. (Visited on 01/19/2023).

[13] T. Katayama, T. Ohki, T. Inoue, and T. Kato. "Design of an optimal controller for a discrete-time system subject to previewable demand". In: *International Journal of Control* 41.3 (Mar. 1985), pp. 677–699. ISSN: 0020-7179, 1366-5820. DOI: 10.1080/00207185089611 56. (Visited on 12/20/2022).

[14] R. Khusainov, A. Sagitov, A. Klimchik, and E. Magid. "Arbitrary Trajectory Foot Planner for Bipedal Walking:" in: *Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics*. 14th International Conference on Informatics in Control, Automation and Robotics. Madrid, Spain: SCITEPRESS - Science and Technology Publications, 2017, pp. 417–424. ISBN: 978-989-758-263-9 978-989-758-264-6. DOI: 10.5220/0006442504170424. (Visited on 12/20/2022).

[15] A. Kojima and S. Ishijima. "H-∞ performance of preview control systems". In: *Automatica* 39.4 (Apr. 2003), pp. 693–701. ISSN: 00051098. DOI: 10.1016/S0005-1098(02)00286-8. (Visited on 05/04/2023).

[16] D. Marhefka and D. Orin. "A compliant contact model with nonlinear damping for simulation of robotic systems". In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 29.6 (Nov. 1999), pp. 566–572. ISSN: 10834427. DOI: 10.1109/3468.798060. (Visited on 03/07/2023).

[17] T. McGeer. "Passive Dynamic Walking". In: *The International Journal of Robotics Research* 9.2 (Apr. 1, 1990). Publisher: SAGE Publications Ltd STM, pp. 62–82. ISSN: 0278-3649. DOI: 10.1177/027836499000900206. (Visited on 06/01/2023).

[18] T. McMahon. "Mechanics of Locomotion". In: *The International Journal of Robotics Research* 3.2 (June 1, 1984). Publisher: SAGE Publications Ltd STM, pp. 4–28. ISSN: 0278-3649. DOI: 10.1177/027836498400300202. (Visited on 06/01/2023).

[19] *NAO (robotique)*. In: *Wikipédia*. Page Version ID: 204015353. May 6, 2023. (Visited on 05/26/2023).

[20] I. Poulakakis and J. Grizzle. "The Spring Loaded Inverted Pendulum as the Hybrid Zero Dynamics of an Asymmetric Hopper". In: *IEEE Transactions on Automatic Control* 54.8 (Aug. 2009), pp. 1779–1793. ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2009.2024565. (Visited on 12/20/2022).

[21] J. Pratt, P. Dilworth, and G. Pratt. "Virtual model control of a bipedal walking robot". In: *Proceedings of International Conference on Robotics and Automation*. International Conference on Robotics and Automation. Vol. 1. Albuquerque, NM, USA: IEEE, 1997, pp. 193–198. ISBN: 978-0-7803-3612-4. DOI: 10.1109/ROBOT.1997.620037. (Visited on 12/20/2022).

[22] J. K. Rai and R. Tewari. "Quintic polynomial trajectory of biped robot for human-like walking". In: *2014 6th International Symposium on Communications, Control and Signal Processing (ISCCSP)*. 2014 6th International Symposium on Communications, Control and Signal Processing (ISCCSP). Athens, Greece: IEEE, May 2014, pp. 360–363. ISBN: 978-1-4799-2890-3. DOI: 10.1109/ISCCSP.2014.6877888. (Visited on 05/17/2023).

[23] N. Shafii, A. Abdolmaleki, N. Lau, and L. P. Reis. "Development of an Omnidirectional Walk Engine for Soccer Humanoid Robots". In: *International Journal of Advanced Robotic Systems* (2015), p. 1. ISSN: 1729-8806. DOI: 10.5772/61314. (Visited on 12/20/2022).

[24] M. Vukobratović and J. Stepanenko. "On the stability of anthropomorphic systems". In: *Mathematical Biosciences* 15.1 (Oct. 1972), pp. 1–37. ISSN: 00255564. DOI: 10.1016/0025-5564(72)90061-2. (Visited on 01/19/2023).

[25] H.-H. Zhao, W.-L. Ma, A. D. Ames, and M. B. Zeagler. "Human-inspired multi-contact locomotion with AMBER2". In: *2014 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. 2014 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS). Berlin, Germany: IEEE, Apr. 2014, pp. 199–210. ISBN: 978-1-4799-4930-4 978-1-4799-4931-1. DOI: 10.1109/ICCPS.2014.6843723. (Visited on 01/20/2023).

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**