# Contents

# 1 Introduction

This chapter provides an overview of the present state of development in the field of quadruped robots, it also presents, a fundamental exposition of the bionic mouse robot *Nermo* that is currently under development at the institution TUM School of Computation, Information and Technology. It proceeds to identify the technical challenges that are addressed in this study as well as the envisaged outcomes. Finally, this chapter includes with a basic outline of the overall structure of this paper.

## 1.1 Initial Situation

**Current Quadruped Robot Platforms.** To date, a multitude of quadruped robots have been developed for use in scientific research and development. Numerous studies have leveraged these robots as a foundation for conducting investigations into diverse areas such as gait realization, stability control, and energy consumption analysis, among others.

Minitaur [1] has successfully realized diverse dynamic running gaits. StarlETH [2] and HyQ [3] have employed distinct methods to achieve trotting gaits and are capable of autonomously navigating through challenging terrains. Boston Dynamics' BigDog [4] and WildCat [5] have exhibited remarkable robustness in achieving walking and fast running gaits in outdoor environments. Research platforms such as MIT Cheetah 2, featuring unique electric actuation systems for the study of high-speed quadrupedal locomotion [6] [7], have successfully achieved a rapid and highly energy-efficient trotting gait [8] as well as dynamic quadrupedal bounding and galloping gaits [9] [10]. Additionally, MIT Cheetah 3 has demonstrated the ability to leap onto high platforms [11], while MIT Mini Cheetah can execute 360º backflips [12].

**Control Algorithms for Robot Locomotion.** Numerous articles have been published regarding control methods for quadruped robots.

One approach involves using hierarchical manipulation space control based on floating body dynamics [13] [14]. This approach uses the self-stabilization of the Spring-Loaded Inverted Pendulum (SLIP) model and a foot placement strategy to plan motion, which is then encoded in the robot by solving inverse dynamics in the form of manipulative space control. Researchers have successfully realized a dynamic trotting gait at 0.7 m/s (1.5 body length/s) in StarlETH using this approach, and another dynamic trotting gait has been demonstrated in [15] using the hydraulic robotic platform HyQ. In this case, the desired trajectory of the robot foot is calculated using an algorithm inspired by the Central Pattern Generator (CPG) and implemented on the robot using feed-forward torque provided by floating-basis inverse dynamics with PD control. HyQ is currently capable of operating at speeds of up to 2.5 m/sec using off-board power.

Additionally, a boundary gait control algorithm was proposed in [16] that allows for variable-speed operation in MIT Cheetah 2. Using a simple pulse planning algorithm, the force distribution design maintains linear momentum over a complete step, providing periodicity in

horizontal and vertical velocity. MIT Cheetah 2 is able to run in a stable manner over a wide range of speeds from 0 m/sec to 4.5 m/sec without re-tuning or re-optimizing any control parameters using this approach. Using the same robot platform and a similar control algorithm, a gallop gait at speeds ranging from 3 m/s to 14.9 m/s has been achieved in [9].

The recent hardware results of ANYmal [17] have also demonstrated the potential of reinforcement learning in dynamic leg motion.

In recent years, optimization-based techniques, notably Model Predictive Control (MPC), have gained increasing attention in relation to legged robot control. MPC has demonstrated remarkable performance in planning and controlling various dynamic motions for both humanoid [18] [19] and quadrupedal [20] [21] robots.

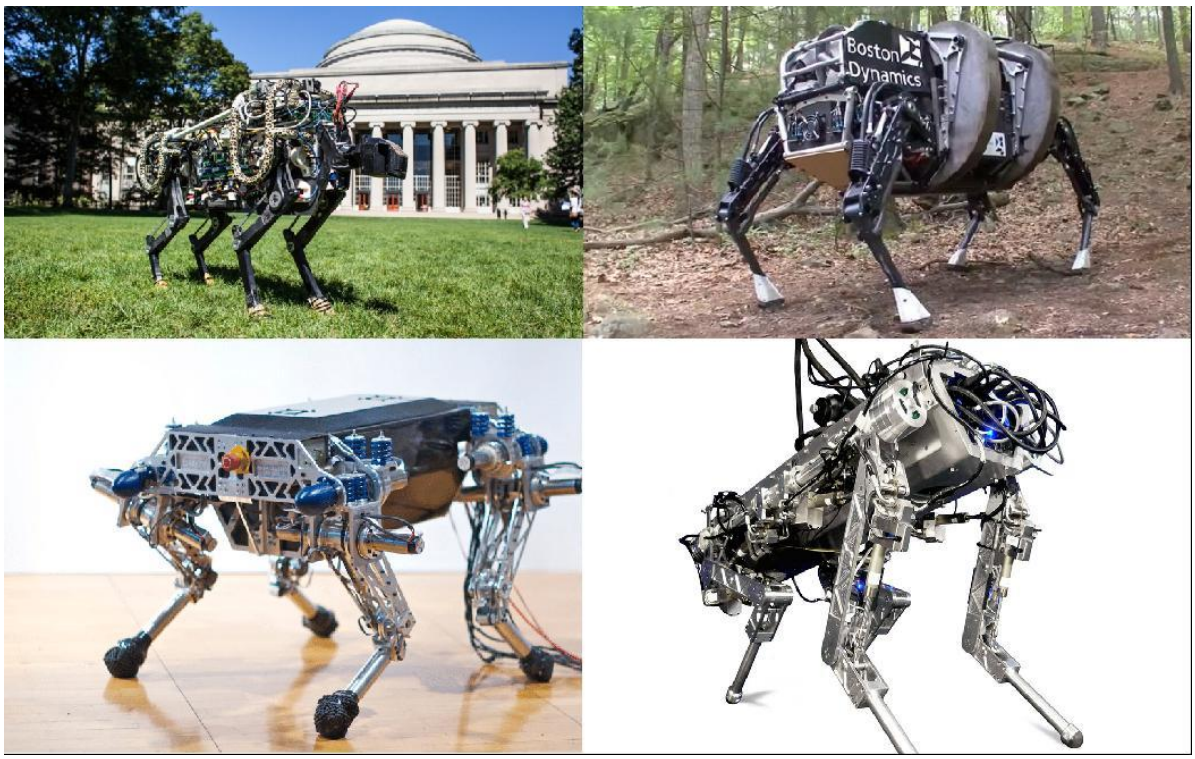


*Figure 1.1: Presented herein are some of the developed quadruped robot platforms. The upper left picture is MIT Cheehah 2, the upper right picture is Boston Dynamics' Big Dog, the lower-left picture is StarlETH, and the lower-right picture is HyQ.*

**Nermo Robot.** In contrast to the aforementioned quadruped robots, the quadruped robot "Nermo" developed by the Informatics 6 - Chair of Robotics, Artificial Intelligence and Real-Time Systems at the School of Information of Technical University Munich (TUM), draws inspiration from the morphology and behavior of mice, rendering it distinctive in terms of its structural and kinematic features.

Nermo is a distinctive quadruped robot for several reasons. Firstly, Nermo boasts a complex and highly compliant/pliable skeleton with a flexible ribcage that accommodates the electronics. Secondly, Nermo features a fully compliant spine that can be actuated in the frontal and sagittal planes, along with compliant knee and ankle joints in the legs. Thirdly, Nermo employs a tendon-driven actuation system. The main physical characteristics of Nermo are summarized in Table 1.1. Notably, Nermo is battery-powered and fully

autonomous, representing the cutting-edge in this type of robotic platform.

Regarding mechanical design, Nermo features a lightweight, conformable, and modular plastic skeleton that interconnects its entire body. The skeleton includes eight compliant joints for the spine, with four for horizontal extension and contraction and four for vertical extension and contraction. These eight joints connect to two spine servo motors (vertical and lateral servo) via tendons, enabling active swing control of the spine.

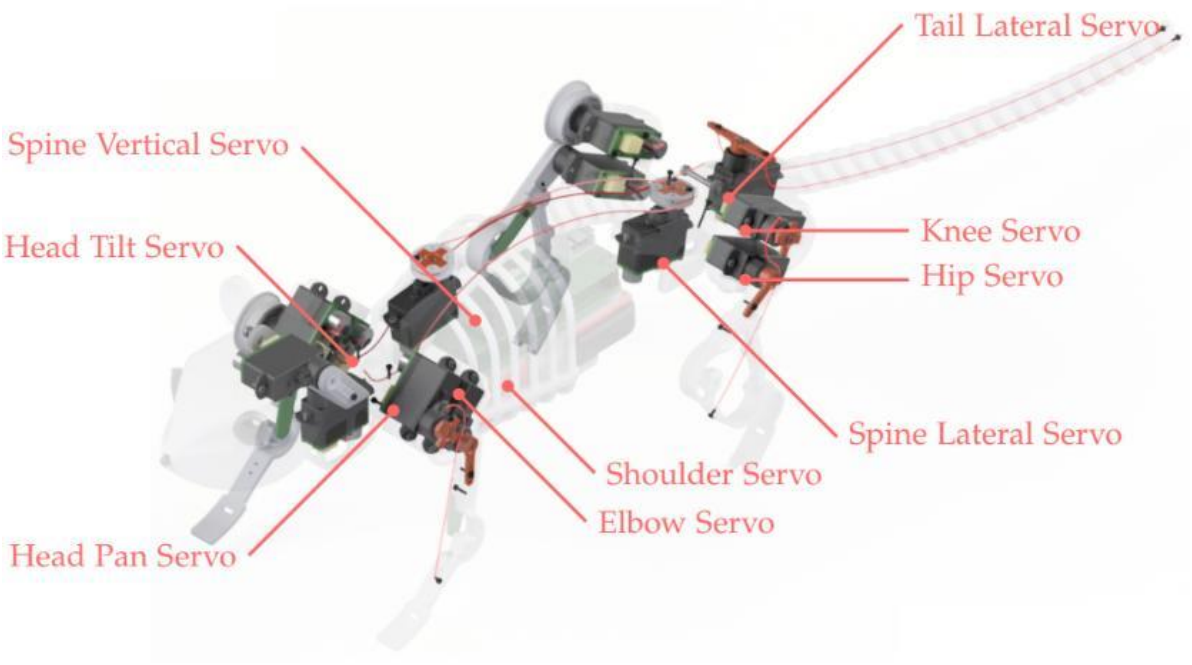


*Figure 1.2: This diagram presents detailed information regarding the servo motors and tendons utilized in Nermo to control its locomotion. The tendons are securely fastened at the attachment points via a press-fit mechanism and M1 screws.*

As illustrated in Figure 1.2, the current mechanical design of Nermo features two joints on each leg, including the shoulder joint and knee joint. The former is directly actuated by a servo motor, while the latter is actuated via a tendon connected to the servo motor. Since this thesis aims to enable Nermo to achieve a running gait, the dynamic characteristics of the leg design must fulfill higher requirements. Therefore, subsequent simulation analysis is necessary to evaluate the feasibility of the existing leg structure. Moreover, Nermo's skeleton also comprises a fully compliant tail consisting of 18 series-connected compliant joints.

Regarding electrical design, Nermo possesses a total of 13 degrees of freedom, as presented in Table 1.2. Nermo's computing platform utilizes the Raspberry Pi Zero, which confers several advantages such as the following: 1) wireless communication with a PC, 2) the capacity for performing computations via a Linux-based operating system, and 3) modularity for possible upgrades in the future, pertaining to both sensors and motions.

Nermo is equipped with several integrated sensors as listed in Table 1.3. These sensors, including servo, leg, and foot sensors, are utilized to regulate Nermo's locomotion.

| Specification | Detail |
|---|---|
| Length | 405 mm |
| Length Scapula-Pelvis | 117 mm |
| Height | 91 mm |
| Width | 90 mm |
| Weight | 275 g |
| Actuated DOFs | 13 |
| Battery | 7.4V 1000mAh LiPo |
| Run Time | 25 min. |
| Main Computer | Raspberry Pi Zero |
| Cost | Approx. 1000EUR |

*Table 1.1: Overview of mechanical and electrical specifications of Nermo.*

| Component | DOF | Details |
|---|---|---|
| Per Leg | 2 | Hip rotation and knee flexion |
| Spine | 2 | Lateral and lumbar flexion |
| Tail | 1 | Lateral flexion |
| Head | 2 | Pan and tilt |

*Table 1.2: List of all 13 degrees of freedom of Nermo. Each degree of freedom is controled by one servo in Figure 1.2.*

| Component | Sensory Capability |
|---|---|
| Servos | Absolute position |
| Legs | Absolute knee angle |
| Feet | Ground pressure |
| Head | 2 HD cameras |

*Table 1.3: List of the integrated sensors in Nermo.*

## 1.2 Problem Definition and Motivation

On the Nermo platform, previous research [22] has developed a method to generate reference motion trajectories for each of the four feet relative to their corresponding shoulder or hip joint. These reference trajectories vary with time and dictate the desired spatial position of the foot relative to the shoulder or hip joint, which is then transformed into the required rotation angle for the leg motors through leg inverse kinematics. These angles are then transmitted to the corresponding position control motor through a control chip to achieve leg movement. By coordinating the movements of the four legs, various static gaits such as “trot” and “walk” can be achieved.

The “zero moment point” (ZMP) method is used to achieve the stability and coordination of the entire body. Specifically, kinematics is utilized to determine the position of the body's center of mass (COM) relative to the standing stability triangle of the body, which affects the calculation of the reference motion trajectory of each foot. This ensures that the robot moves in a stable manner.

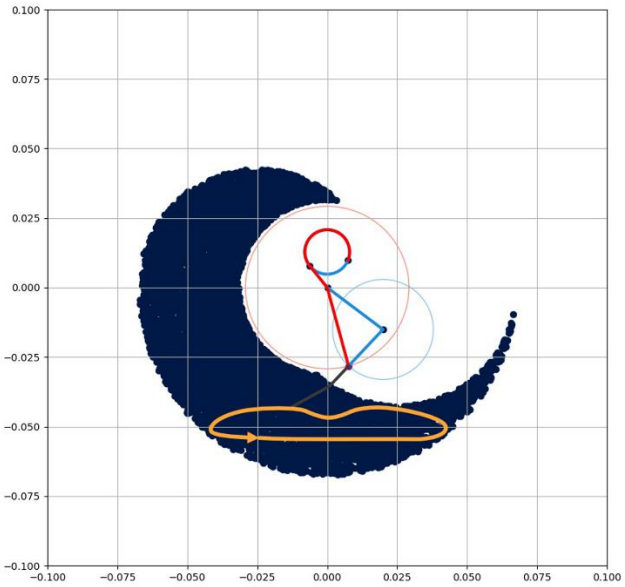


*Figure 1.3: This is a mathematical simulation of the single-leg structure in the prior study. The circular shape with thick edges positioned approximately at the center of the coordinate axis denotes the location of the shoulder or hip joint. The solid lines in blue and dark green represent the leg structure, while the solid red line indicates the tendon and connects the shoulder or hip joint and the leg, which imparts power to the lower leg. The thick and closed yellow curve depicts a potential reference motion trajectory of the foot, while the entire blue region denotes the location that the foot can occupy under the current leg structure. [22]*

However, for dynamic gaits such as bounding, galloping, and running trot, the approach based on robot kinematics and the position control motor is not suitable. This is due to the quadruped’s airborne phase during the dynamic gait cycle, where all four legs leave the ground simultaneously. The duration and jump height of the next airborne phase are determined by the contact time and ground reaction force of each leg upon landing. Additionally, the swing effect of the legs, which pertains to whether and how effectively the foot reaches the desired position, during the airborne phase requires precise force control and is influenced by the duration of the airborne phase. Consequently, achieving dynamic gaits necessitates more advanced control algorithms and more precise force-controlled motors.

Hence, this study aims to investigate and identify a sophisticated control algorithm for Nermo to attain dynamic gaits, drawing insights from other scholarly research concerning quadruped robotic platforms.

## 1.3 Objective

Firstly, this study aims to identify an appropriate control algorithm by examining the relevant literature. It endeavors to review and understand the relevant algorithms and potential programming codes. Subsequently, it establishes the control algorithm in a Python code framework, which is associated with the physical simulation environment; it then tests the algorithms to ascertain its ability to drive a simple quadruped robot in the physical simulation environment, achieving steady movement with diverse static and dynamic gaits. If the algorithm can enable a range of gaits, it will be tested for application in controlling the Nermo robot in subsequent work, thereby providing insights regarding the structural design and motor selection of the Nermo robot.

## 1.4 Outline

Chapter 2 of this paper introduces the utilized physical simulation environment MuJoCo and provides a brief introduction to the movement patterns of quadrupeds. Furthermore, the control algorithm RF-MPC and optimization algorithm qpSWIFT utilized in the control loop are presented. Chapter 3 discusses the derived kinematic and dynamic formulas of various legs and spines, which will then be integrated into the control framework in this paper and in future work to control quadruped robots with different structures. Chapter 4 introduces the integration of the control algorithm in Chapter 2 and the insights regarding kinematics and dynamics in Chapter 3; it also adds other necessary modules to build a complete control loop for simulating a quadruped robot locomotion in MuJoCo. In Chapter 5, experiments concerning achieving quadruped robot gaits are conducted, and the results are analyzed. Chapter 6 provides a summary of this paper, while Chapter 7 analyzes possible future research directions.

# 2 State of the Art

This chapter first introduces the simulation environment adopted in this paper along with its comparative advantages over other simulation environments. Subsequently, it presents an exposition regarding quadruped robots' gaits, expounding on the characteristics of each gait. Finally, based on the gait state, it introduces the control algorithm and optimization method employed in this paper.

## 2.1 Simulation Frameworks MuJoCo

Simulation environments provide a cost-effective and efficient means to test new designs and control algorithms for robots by using virtual robots (also known as digital twins) in virtual environments. To achieve this, advanced robotic simulators require strong modeling capabilities, API integration (C++ and Python), and graphical interface options. Additionally, simulators must be physically accurate and leverage efficient recursive algorithms in joint and velocity space to accurately compute contact dynamics.

As this paper builds upon previous research, it utilizes the MuJoCo robot simulation environment. One of the primary advantages of MuJoCo over other simulation software is its ability to simulate tendons, which is a key design feature of Nermo's tendon-driven legs, spine, and tail. Nevertheless, one of the most popular reinforcement learning toolkits, OpenAI Gym, implements the environments used for baseline testing of reinforcement learning algorithms in MuJoCo.

| Simulator | RGBD & LiDAR | Force Sensor | Tendon Actuator | Soft-Body Contacts | Real Rendering | IK |
|---|---|---|---|---|---|---|
| CoppeliaSim | √ | √ | × | × | × | √ |
| Gazebo | √ | √ | × | × | × | √ |
| MuJoCo | √ | √ | √ | √ | × | × |
| PyBullet | √ | √ | × | √ | × | √ |
| Chrono | √ | √ | √ | √ | √ | √ |

*Table 2.1: Summary of contemporary robotic simulators utilized in research. This summary has been collated and condensed from reference [23].*

Additionally, MuJoCo has diverse and versatile applications. It enables the implementation of model-based computations, such as control synthesis, state estimation, system identification, mechanism design, data analysis via inverse dynamics, and parallel sampling for machine learning applications. Furthermore, it can serve as a conventional simulator, which can be used for gaming and for interactive virtual environments.

## 2.2 Quadruped Animal Locomotion

The movements of legs in biped or quadruped animals, follows periodic patterns, known as the “gait cycle”. Quadrupeds exhibit different “rhythms” in the periodic movement of their legs, depending on the desired stride length. These “rhythms” are referred to as “gait patterns” or simply “gaits”. Figure 2.1 presents six typical gaits of quadrupeds, including “walk”, “trot”, “pace”, “bound”, “rotary gallop”, and “transverse gallop”. To describe the movements of the four legs more explicitly, this paper uses an abbreviation for each leg: left forelimb (LF), right forelimb (RF), right hindlimb (RH), and left hindlimb (LH).

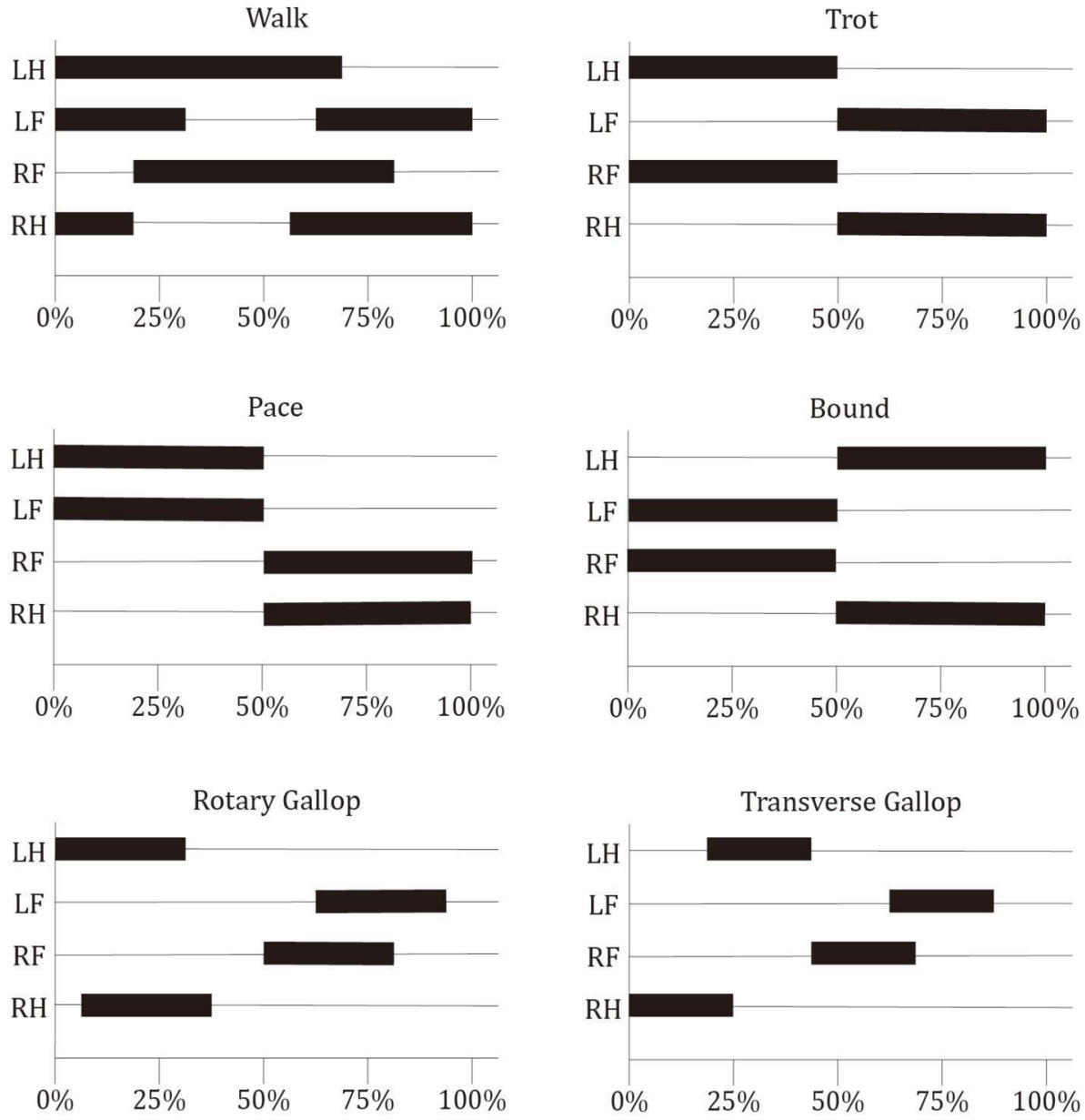


*Figure 2.1: The Gait Cycle of Quadrupedal Robots illustrated in Graphs [24] [25]. The solid blocks represent the stance phase, while the lines signify the swing phase. The duration and phase relationship of the four-legged stance and swing phases are not fixed but vary depending on the desired stride length of the robot. The following is a general description of the characteristics of each gait.*

In the figure corresponding with each gait in Figure 2.1, the vertical axis represents the four legs of the quadruped, namely LH, LF, RF, and RH, while the horizontal axis corresponds with the phase in a gait cycle, ranging from the beginning (0%) to the end (100%) of a cycle. Each leg exhibits different states during different phases of the gait cycle, the solid blocks denote the "stance phase", indicating that the leg is in contact with the ground, while the lines represent the "swing phase", indicating that the leg is in the air and not in contact with the ground.

The duty factor, defined as the ratio of the stance phase time to the total gait cycle time, is a key parameter used to classify gaits [24]. Walking gaits have duty factors greater than 0.5, which means that during any given phase, at least one leg must be in contact with the ground [26] [27]. On the other hand, running gaits have duty factors of less than 0.5, which means that none of the four legs are in contact with the ground during some phases [28]. Hence, the duty factor and phase offset (as depicted in Figure 2.1) are functional parameters that define different gait types mathematically. Among the six gaits introduced in Figure 2.1, "walk" is an absolute walking gait, while the two variants of the gallop gait, namely "rotary gallop", and "transverse gallop", are absolute running gaits. The duty factor of the other three gaits, namely "trot", "pace" and "bound", can be greater than or less than 0.5, depending on the desired stride length, allowing for a relatively large range of robot forward speed under these three gaits.

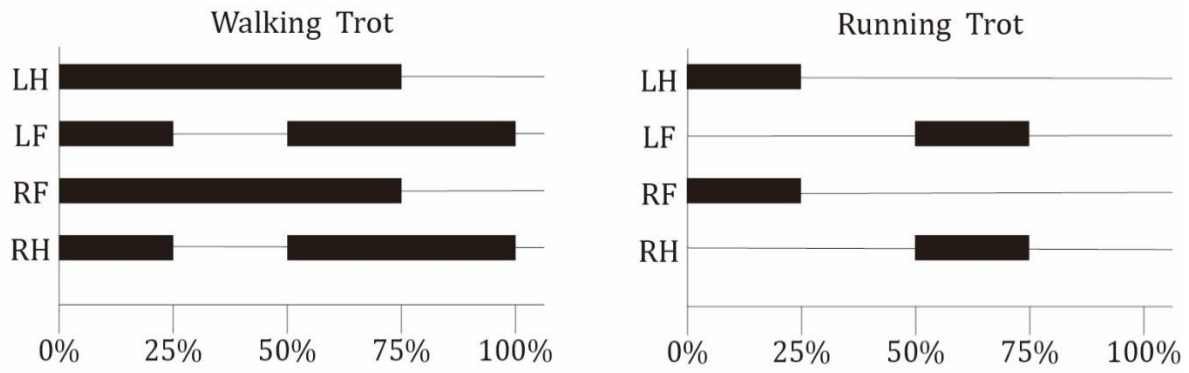


*Figure 2.2: The duty factor of the trot gait can be modified, with the running trot exhibiting a duty factor of less than 0.5 and the walking trot exhibiting a duty factor greater than 0.5. Similar gait types with analogous features encompass the bound gait and pace gait.*

The overall forward velocity of a quadruped is predominantly determined by two factors, namely "gait frequency" and "stride length". Gait frequency signifies the number of gait cycles performed within a specific period, while stride length refers to the distance traveled by the quadruped during one gait cycle. Walking gaits are suitable for low-speed locomotion, while high-speed locomotion necessitates not only a proportional increase in gait frequency but also a realization of the desired stride length through running gaits, such as "trot" or "gallop" [16] [30]. Since quadruped robots are designed differently and since natural quadrupeds vary in size, it is challenging to determine which gait corresponds to a particular forward speed range. However, in [25], the author implemented several gaits on the same robot, including 0 - 1.5 m/s for the "walk" gait, 1.0 - 2.5 m/s for the "trot" gait, 2.0 - 3.5 m/s for the "pace" gait, and 3.5 - 5.5 m/s for the "transverse gallop" gait, which partially

uncovered the relationship between the overall forward speed required by the quadruped robot and the adopted gait. Moreover, an experimental gait analysis conducted by several researchers [26] [31] indicated the relationship between the overall forward velocity and gait frequency. Gait frequency varies from 2.4 Hz (slower gaits) to 8.0 Hz (faster gaits), with a mean gait frequency of 3.7 Hz being applicable to walking gaits.

In addition to leg movement, spinal movement also plays a role in the overall forward locomotion of quadrupeds. According to a study by Fischer et al. [32], rodents rely on the movement of the pelvis for forward locomotion. This movement is caused by the flexion of the spine along the sagittal plane, with joint amplitudes ranging from 3 to 12 degrees. Furthermore, Fischer et al. discovered that for rapid gait types such as bound and gallop, sagittal plane spine movement contributes approximately half of the total forward locomotion of rodents. This suggests that rodents utilize sagittal spinal movement as an effective mechanism for extending stride length and increasing locomotion speed in relation to rapid gait types.

## 2.3 Representation-Free Model Predictive Control (RF-MPC)

To achieve a dynamic quadrupedal gait, a comprehensive analysis of robot dynamics and inverse dynamic is necessary, and the performance of the robot's control chip and environmental constraints must be considered. In previous years, when control chip performance was limited and optimization algorithms were not yet well developed, robot control algorithms relied on open-loop control with minimal computational demands. Trajectories or joint torques were predetermined, limiting the robot’s ability to respond to complex external conditions [33] [34] [35] [36].

The advancement of hardware technology and optimization algorithms has facilitated the current chip performance and real-time calculation capabilities to support intricate optimization algorithms. Consequently, this progress has contributed to the widespread development and utilization of Model Predictive Control (MPC) in the realm of robot control algorithms. In contrast to traditional control algorithms, MPC formulates the control problem as an optimization problem that accounts for the constraints of system dynamics as well as other system-internal and environmental constraints. By predicting the future state of the robot at multiple time steps based on the current state, MPC can confer a significant advantage over previous control algorithms. Furthermore, the optimization-based nature of MPC enables increased flexibility in its description equation, enabling it to be modified, increased, or decreased as per the specific needs and control requirements of robots operating in varied scenarios and structures. Currently, MPC has been successfully applied to both humanoid [18] [37] and quadruped [20] robots, demonstrating its capability to effectively control complex dynamic motions and to respond to complex external conditions.

Despite the aforementioned merits and the widespread usage of MPC, its optimization equations may become intricate for systems with high degrees of freedom (DOFs), and its solution may demand a significant number of computational resources, thereby compromising its real-time performance. To reduce the complexity of the system model and its corresponding optimization equation, it is necessary to simplify the system model to some

extent. For example, the linear inverted pendulum (LIPM) model was adopted in [38], while the robot model was simplified into a plane single rigid body model in [29], this enabled the MIT Cheetah 2 robot to surmount obstacles of different heights. An additional example is the spring-mass model used in [39]. These simplified models are referred to as central dynamics models [40], which primarily consider the dynamics of the main part of the robot (torso) and separate the legs from the main part, thus simplifying the optimization equations that are ultimately derived.

Building upon the simplification of the system model, the further simplification of the dynamic description equation of the system can be achieved. Currently, the Euler angle is commonly used to represent the direction of the robot [41] [42] [43] [44], but it suffers from singularity issues [45]. When the robot approaches the singular direction, such as nearly vertical upward or rolling movements, it can produce in disorderly motion. On the other hand, quaternions [46] do not suffer from singularity issues, but they have two local charts and do not have a one-to-one correspondence with SO(3). Rather, they have a one-to-two relationship, which can generate an expansion phenomenon [47]. Initially, the spatial orientation of a single rigid body was represented by a rotation matrix that evolved on SO(3) [48]. Although Euler angles and quaternions emerged as alternative representations that can correspond to the rotation matrix, the rotation matrix is generally more compact, global, and devoid of singularities. Accounting for the aforementioned factors, this paper employs the RF-MPC approach to govern the locomotion of the robot [49].

In contrast to the conventional MPC method, the RF-MPC approach employs a rotation matrix to depict the spatial orientation of the robot. Subsequently, the RF-MPC algorithm linearizes the rotation matrix using a variation-based linearization technique [49] [50] [51] and computes an approximate original equation's error term to compensate for the linearization error. As a result, RF-MPC preserves the non-singularity and globality features of the rotation matrix while linearizing the nonlinear rotation matrix, converting RF-MPC into a Quadratic Program (QP). In contrast to the resolution of nonlinear optimization problems, the QP solution time of RF-MPC is significantly faster.

The fundamental concept behind RF-MPC for quadruped robot motion control can be elucidated as follows: Firstly, the robot torso is isolated to derive dynamic equations, which constitute one of the constraints in RF-MPC. Secondly, the motion constraints of the robot, such as the friction cone and ground reaction force (GRF) constraints, are taken into consideration. Finally, an error equation between the current robot state and the desired state is established as the objective equation for the optimization problem. During each sampling time, RF-MPC as a optimization problem is solved for a finite horizon (timestep) towards the future, and the control parameter for the robot at the next sampling time is obtained based on the result of the first horizon in the optimization result.

The following section provides a brief explanation of the formula derivation in RF-MPC. For a detailed derivation of the formulas, refer to [49] and unless otherwise stated, all formula derivations in this section are sourced from [49].

### 2.3.1 General Formulation of MPC

The general formulation of MPC can be expressed as follow:

$$\text{minimize} \quad \ell_T(\boldsymbol{x}_{t+N|t}) + \sum_{k=0}^{N-1} \ell(\boldsymbol{x}_{t+k|t}, \boldsymbol{u}_{t+k|t}) \tag{2-1a}$$

$$\text{subject to} \quad \boldsymbol{x}_{t+k+1|t} = \boldsymbol{f}(\boldsymbol{x}_{t+k|t}) + \boldsymbol{g}(\boldsymbol{x}_{t+k|t})\boldsymbol{u}_{t+k|t} \tag{2-1b}$$

$$k = 0, \cdots, N-1 \tag{2-1c}$$

$$\boldsymbol{x}_{t+k|t} \in \mathbb{X} \tag{2-1d}$$

$$\boldsymbol{u}_{t+k|t} \in \mathbb{U} \tag{2-1e}$$

$$\boldsymbol{x}_{t|t} = \boldsymbol{x}(t) = \boldsymbol{x}_{op} \tag{2-1f}$$

$$\boldsymbol{x}_{t+N|t} \in \mathbb{X}_f \tag{2-1g}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$ are the state vector and input vector with vector lengths $n$ and $m$. $N$ is the prediction horizon (timesteps to be predicted). $\boldsymbol{x}_{t+k|t}$ denotes the state vector at timestep $t+k$ that is predicted at time $t$, the same situation applies to $\boldsymbol{u}_{t+k|t}$.

The cost function of MPC involves two components: the terminal cost function $\ell_T$: $\mathbb{R}^n \to \mathbb{R}$ and the stage cost function $\ell$: $\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$. The stage cost function captures the deviation of the system state and input from their respective references over the first $N-1$ predicted horizons, while the terminal cost function quantifies the deviation of the system state and input from their references over the $N$-th predicted horizon. These two cost functions that are discussed in this paper have similar expressions.

Equation $f(x) + g(x) \cdot u$ represent the general form of the control affine dynamic update equation. The feasible polyhedral sets for the state and input vector are denoted by $\mathbb{X} \in \mathbb{R}^n$ and $\mathbb{U} \in \mathbb{R}^m$, respectively. The final state set (state vector at last horizon predicated at time $t$) is then represented by $\mathbb{X}_f \in \mathbb{R}^n$. $\boldsymbol{x}_{t|t}$ or $\boldsymbol{x}_{op}$ denotes the current state vector of the robot or the initial state of the MPC in this control iteration, the subscript *op* "operating point" means the current sampling time.

As previously noted, when employing the whole-body dynamics equation of a quadruped robot or utilizing the rotation matrix to characterize the robot's orientation, the control affine dynamic update equation $f(x) + g(x) \cdot u$, becomes nonlinear in nature. Therefore, the solution of this equation necessitates the use of Nonlinear Programming (NLP) techniques [52] [53].

The solution of Nonlinear Programming (NLP) problems is typically computationally demanding and time-consuming. To achieve real-time control over robotic motion, it is frequently necessary to simplify the optimization equation. As discussed in reference [49], for instances in which the total mass of all legs constitutes less than 10% of the total body mass, the dynamics of the robot can be modeled using a single rigid body (SRB) model, namely the central dynamics model, as opposed to the full-body dynamics model.

### 2.3.2 3D Single Rigid Body Model

The silhouette presented below depicts a basic quadruped robot created in MuJoCo. This section shall establish the dynamic equation (2-1b) in RF-MPC using this model as a basis.

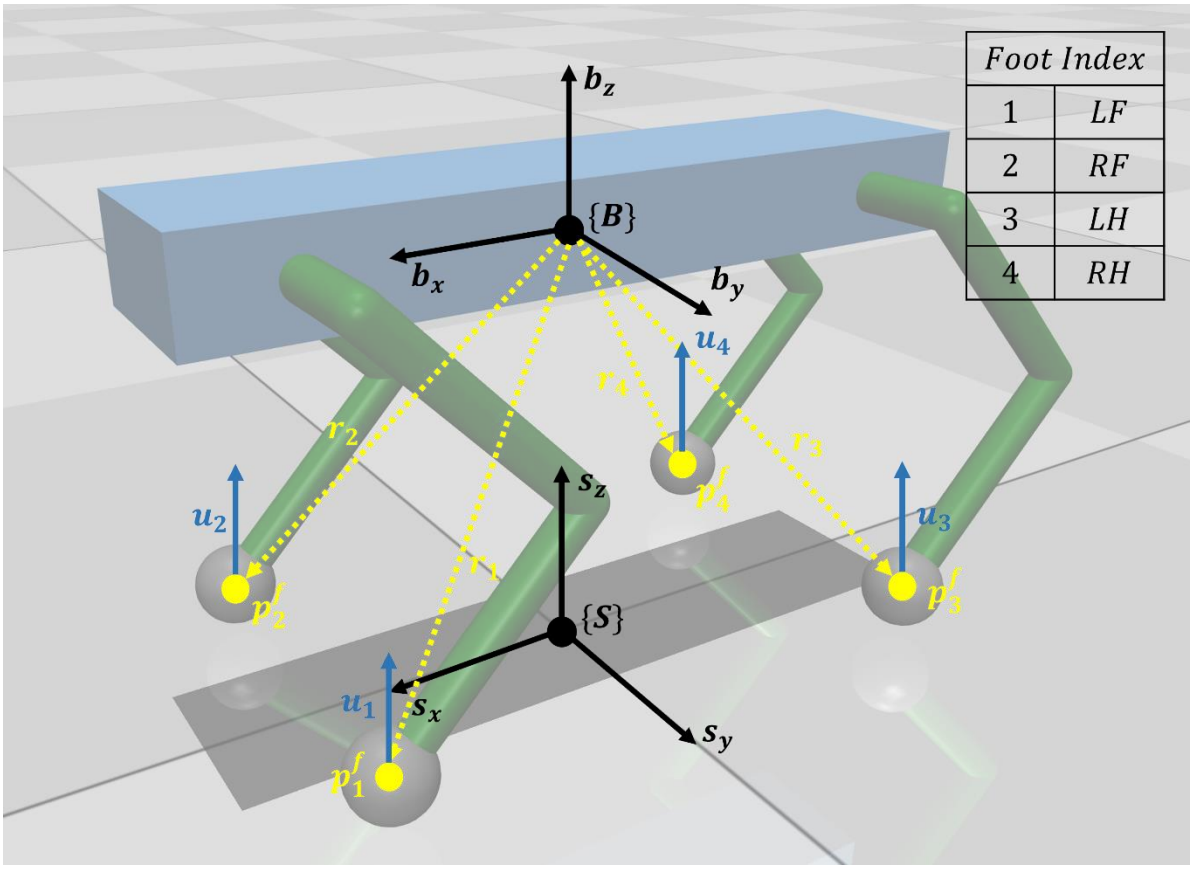


*Figure 2.3: Illustration of the coordinate system and the 3D single rigid body model. The coordinate system $\{S\}$ represents the inertial system, namely the world coordinate system, while $\{B\}$ represents the body COM coordinate system. Each foot's spatial position in $\{S\}$ is denoted by $p_i^f$, with $r_i$ representing the position vector from the COM position in $\{S\}$ to $p_i^f$, and $u_i$ representing the ground reaction force on the $i$-th contact foot in $\{S\}$. The convention for foot numbering is LF for the left front leg and RH for the right rear leg.*

Initially, the state vector of the SRB in Figure 2.3 is defined as follows:

$$\boldsymbol{x} \coloneqq \left[\boldsymbol{p},\ \dot{\boldsymbol{p}},\ \boldsymbol{R},\ {}^{B}\boldsymbol{\omega}\right] \in \mathbb{R}^{18} \qquad (2-2)$$

where $\boldsymbol{p}$ denotes the linear position of the COM of the robot with respect to the world coordinate system, $\dot{\boldsymbol{p}}$ represents the linear velocity of the COM in the world coordinate system, and $\boldsymbol{R} \in SO(3) = \{\boldsymbol{R} \in \mathbb{R}^{3\times3} | \boldsymbol{R}^T\boldsymbol{R} = \mathbb{I}, \det(\boldsymbol{R}) = 1\}$ is represented as a rotation matrix in which $\det(\cdot)$ is shorthand for the matrix determinant. This matrix can be utilized to convert the vector expressed in the body coordinate system to the world coordinate system. The rotation matrix $\boldsymbol{R}$ is reshaped into vector form in the state vector $\boldsymbol{x}$, namely $\mathbb{R}^{3\times3} \rightarrow \mathbb{R}^9$. ${}^{B}\boldsymbol{\omega}$ indicates the angular velocity of COM in the body coordinate system.

Notably in this paper, all variables without superscript in the upper-left-hand corner ${}^{B}(\cdot)$ are presumed to be within the world coordinate system.

When considering only the dynamics of the robot's body, the input vector $\mathbf{u}_i \in \mathbb{R}^3$ of the

system is the GRF received by the four legs at the foot-end position $\mathbf{p}_i^f \in \mathbb{R}^3$, with $\boldsymbol{u} \coloneqq [\boldsymbol{u_1},\ \boldsymbol{u_2},\ \boldsymbol{u_3},\ \boldsymbol{u_4}] \in \mathbb{R}^{12}$. The moment arm of the GRF on the COM can be represented by the position vector pointing from the COM to the foot-end position $\mathbf{r}_i = \mathbf{p}_i^f - \boldsymbol{p}$, where the index $\boldsymbol{i} \in \{1,2,3,4\}$ corresponds to the four legs [LF, RF, LH, RH], as depicted in Figure 2.3. Notably, the GRF and its moment arm must be expressed in the same reference system, which is consistently defined within the world coordinate system $\{S\}$ throughout this paper.

Hence, the resultant external force $\boldsymbol{F}$ and moment $\boldsymbol{M}$ applied to the COM can be formulated as follows:

$$\boldsymbol{F} = \sum_{i=1}^{4} \boldsymbol{u_i} \tag{2-3}$$

$$\boldsymbol{\tau} = \sum_{i=1}^{4} \hat{\boldsymbol{r}}_i \boldsymbol{u_i} \tag{2-4}$$

The hat map $\hat{\boldsymbol{r}}_i$ represents the skew-symmetric matrix of $\boldsymbol{r_i}$; thus, the cross-product can be represented by $\hat{\boldsymbol{r}}_i \boldsymbol{u_i} = \boldsymbol{r_i} \times \boldsymbol{u_i}$. The inverse of the hat map is the vee map with $(\hat{\boldsymbol{r}}_i)^V = \boldsymbol{r_i}$.

Building upon the aforementioned information, the body dynamics of the robot can be mathematically formulated as follows:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \ddot{\boldsymbol{p}} \\ \dot{\boldsymbol{R}} \\ {}^B\dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \frac{1}{M}\boldsymbol{F} + \boldsymbol{a}_g \\ \boldsymbol{R} \cdot {}^B\dot{\boldsymbol{\omega}} \\ {}^B\boldsymbol{I}^{-1}\left(\boldsymbol{R}^\top \boldsymbol{\tau} - {}^B\hat{\boldsymbol{\omega}}\, {}^B\boldsymbol{I}\, {}^B\boldsymbol{\omega}\right) \end{bmatrix} \tag{2-5}$$

where $\boldsymbol{a}_g = [0, 0, -g]^T$ corresponds to the gravitational acceleration vector, $M$ is the total mass of the robot, and ${}^B\boldsymbol{I} \in \mathbb{R}^{3\times3}$ is the body inertia tensor with only diagonal terms in the body frame $\{B\}$.

The rotation matrix of the robot $\boldsymbol{R}$ is determined by the Euler angle $\boldsymbol{\Theta} = [\phi\ \theta\ \psi]^T$ representation of the robot, which is used in RF-MPC to represent the orientation of the robot. This paper presumes that the solution sequence of $\boldsymbol{R}$ is X-Y-Z. In this sequence, a vector in the robot body coordinate system is first rotated by the roll angle $\phi$ around the X-axis (i.e., multiplied by the rotation matrix around X-axis $\boldsymbol{R_x}$), followed by a rotation of the pitch angle $\theta$ around the Y-axis (i.e., multiplied by the rotation matrix around Y-axis $\boldsymbol{R_Y}$); subsequently, it is finally rotated by the yaw angle $\psi$ around the Z-axis (i.e., multiplied by the rotation matrix around Z-axis $\boldsymbol{R_Z}$). The resulting vector represents the equivalent vector in the world coordinate system. The rotation matrix around the axis in three directions can be computed by the following:

$$\boldsymbol{R_x}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2-6}$$

$$\boldsymbol{R_y}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad (2-7)$$

$$\boldsymbol{R_z}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2-8)$$

Therefore, the complete rotation matrix considering the rotation around all three axes can be computed as follows:

$$\boldsymbol{R} = \boldsymbol{R_z}(\psi)\boldsymbol{R_y}(\theta)\boldsymbol{R_x}(\phi) \quad (2-9)$$

Another approach to calculate the rotation matrix utilizing Euler angles can be formulated as follows:

$$\boldsymbol{R} = expm(\widehat{\boldsymbol{\theta}}) \quad (2-10)$$

where $expm(\boldsymbol{x})$ computes the matrix exponential of $\boldsymbol{x}$ with $\boldsymbol{x} \in \mathbb{R}^{3\times3}$. This command can be used directly in MATLAB, and it is necessary to call it through the "scipy.linalg" library when using Python.

Given that the rotation matrix in the dynamic equation (2-5) is a nonlinear term, it is imperative to linearize it to obtain a linear dynamic equation that can be solved using QP. This constitutes the most prominent disparity between RF-MPC and alternative MPC approaches.

### 2.3.3 Linearization, Vectorization and Discretization

Nonlinear Model Predictive Control (MPC) methods suffer from slow solution processes, poor real-time performance, and the risk of local optima. To address these limitations, RF-MPC introduces a variation-based rotation matrix linearization scheme, converting nonlinear MPC into Quadratic Programming (QP). Specifically, the differential of the rotation matrix relative to the operating point is solved to predict the value of the rotation matrix on the predicted horizon [50]. In this manner, formula (2-5) can be fully transformed into variational dynamics [54].

Notably, this section solely introduces the relevant formulas for the conclusion. The full derivation of the formulas is elucidated in the literature [49].

The horizon interval of MPC is typically short, specifically on the order of 1e-3 to 1e-2, which makes it possible to solve the variation of the rotation matrix $\delta\boldsymbol{R}$ based on the error function of the rotation matrix under $SO(3)$. $\delta\boldsymbol{R} \in SO(3)$ is a local approximate linearization of a rotation matrix, through which the first-order Taylor expansion of the rotation matrix can be used to solve the rotation matrix of the k-th horizon:

$$\boldsymbol{R_k} \approx \boldsymbol{R_{op}}(\mathbb{I} + \delta\boldsymbol{R_k}) \quad (2-11)$$

Based on this assumption, the variation of the rotation matrix at the k-th horizon, namely the third function in (2-5), can be obtained simultaneously:

$$\dot{\boldsymbol{R}}_k = \boldsymbol{R}_{op}\widehat{\boldsymbol{\omega}}_{op} + \boldsymbol{R}_{op}\widehat{\boldsymbol{\omega}}_{op}\delta\boldsymbol{R}_k + \boldsymbol{R}_{op}\widehat{\delta\boldsymbol{\omega}_k} \tag{2-12}$$

The fourth function in (2-5) can be linearized as follows:

$$\begin{aligned} {}^{\boldsymbol{B}}\boldsymbol{I}\dot{\boldsymbol{\omega}}_k = {} & \boldsymbol{R}_{op}^{\top}\boldsymbol{\tau}_{\boldsymbol{op}} + \delta\boldsymbol{R}_k^{\top}\boldsymbol{\tau}_{op} + \boldsymbol{R}_{op}^{\top}\delta\boldsymbol{\tau}_k + \\ & -\widehat{\boldsymbol{\omega}}_{op}\,{}^{\boldsymbol{B}}\boldsymbol{I}\boldsymbol{\omega}_{op} - \delta\widehat{\boldsymbol{\omega}}_k\,{}^{\boldsymbol{B}}\boldsymbol{I}\boldsymbol{\omega}_{op} - \widehat{\boldsymbol{\omega}}_{op}\,{}^{\boldsymbol{B}}\boldsymbol{I}\delta\boldsymbol{\omega}_k, \end{aligned} \tag{2-13}$$

$\delta\boldsymbol{\tau}_k$ can be expressed as follows:

$$\delta\boldsymbol{\tau}_{\boldsymbol{k}} = \left(\sum_{i=1}^{4}\widehat{\boldsymbol{u}}_{i,op}\right)\delta\boldsymbol{p}_k + \left(\sum_{i=1}^{4}\hat{\boldsymbol{r}}_{i,op}\cdot\delta\boldsymbol{u}_{i,k}\right) \tag{2-14}$$

Notably, *op* denotes the current step, which is the initial state of the MPC. Therefore, all parameters labeled as *op* can be directly obtained or calculated from the current state. In other words, these parameters are regarded as "known". For example, $\boldsymbol{u}_{op}$ is the GRF applied at the current step and can be obtained by the MPC results in the last iteration (timestep). $\delta\boldsymbol{u}$ is the variation of GRF from $\boldsymbol{u}_{op}$; thus, the GRF applied at the first horizon can be expressed by $\boldsymbol{u}_{op} + \delta\boldsymbol{u}_{\boldsymbol{1}}$.

At this juncture, the dynamic equation still incorporates differential terms that are unknown. To further simplify the equation, the Kronecker product [55] and vectorization are utilized.

Assuming the existence of a vector $\boldsymbol{\xi} \in \mathbb{R}^3$ whose skew-symmetric matrix equals $\delta\boldsymbol{R}$, namely $\hat{\boldsymbol{\xi}} = \delta\boldsymbol{R}$, the vectorization $\mathbb{R}^{3\times3} \to \mathbb{R}^9$ of a vector's skew-symmetric matrix is related to this vector as follows:

$$vec(\hat{\boldsymbol{\xi}}) = N \cdot \boldsymbol{\xi} \tag{2-15}$$

with

$$N = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{2-16}$$

, then

$$vec(\delta\boldsymbol{R}) = N \cdot \boldsymbol{\xi} \tag{2-16}$$

Given this assumption, it is possible to solve equations (2-12) in discrete form with

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + dt\boldsymbol{N}^*\left(\mathbb{I}\otimes\boldsymbol{R}_{op}^{\top}\right)\left(\boldsymbol{C}_{\xi}^{c} + \boldsymbol{C}_{\xi}^{\xi}\boldsymbol{\xi}_k + \boldsymbol{C}_{\xi}^{\omega}\boldsymbol{\omega}_k\right) \tag{2-17}$$

where the constants are defined as follows:

$$\begin{aligned} \boldsymbol{C}_{\xi}^{c} &= \mathrm{vec}\left(\boldsymbol{R}_{op}\widehat{\boldsymbol{\omega}}_{op}\right) - \left(\mathbb{I}\otimes\boldsymbol{R}_{op}\right)\boldsymbol{N}\boldsymbol{\omega}_{op} \\ \boldsymbol{C}_{\xi}^{\xi} &= \left(\mathbb{I}\otimes\boldsymbol{R}_{op}\widehat{\omega}_{op}\right)\boldsymbol{N} - \left(\mathbb{I}\otimes\boldsymbol{R}_{op}\right)\boldsymbol{N}\widehat{\boldsymbol{\omega}}_{op} \\ \boldsymbol{C}_{\xi}^{\omega} &= \left(\mathbb{I}\otimes\boldsymbol{R}_{op}\right)\boldsymbol{N} \end{aligned} \tag{2-18}$$

$\otimes$ is the Kronecker tensor operator. $dt$ is also a constant and denotes the MPC sampling time.

The equation (2-13) can also be solved as follows:

$$ {}^{B}\boldsymbol{I}\dot{\boldsymbol{\omega}}_k = \boldsymbol{C}_{\dot{\omega}}^{c} + \boldsymbol{C}_{\dot{\omega}}^{\delta p}\boldsymbol{p}_k + \boldsymbol{C}_{\dot{\omega}}^{\xi}\boldsymbol{\xi}_k + \boldsymbol{C}_{\dot{\omega}}^{\dot{\omega}}\boldsymbol{\omega}_k + \boldsymbol{C}_{\dot{\omega}}^{\delta u}\delta\boldsymbol{u}_k, \qquad (2-19) $$

with

$$ \begin{aligned} \boldsymbol{C}_{\dot{\omega}}^{c} &= -\hat{\boldsymbol{\omega}}_{op}\,{}^{B}\boldsymbol{I}\boldsymbol{\omega}_{op} + \boldsymbol{R}_{op}^{T}\boldsymbol{\tau}_{op} - \left(\widehat{{}^{B}\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op}\,{}^{B}\boldsymbol{I}\right)\boldsymbol{\omega}_{op} \\ &\quad -\boldsymbol{R}_{op}^{\top}\left(\sum\hat{\boldsymbol{u}}_{op}\right)\boldsymbol{p}_{op} \\ \boldsymbol{C}_{\dot{\omega}}^{\delta p} &= \boldsymbol{R}_{op}^{\top}\left(\sum_{i}\hat{\boldsymbol{u}}_{op}^{i}\right) \\ \boldsymbol{C}_{\dot{\omega}}^{\xi} &= \left(\mathbb{I}\otimes\boldsymbol{\tau}_{op}^{\top}\right)\boldsymbol{N} - \left(\widehat{{}^{B}\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op}\,{}^{B}\boldsymbol{I}\right)\hat{\boldsymbol{\omega}}_{op} \\ \boldsymbol{C}_{\dot{\omega}}^{\omega} &= \widehat{{}^{B}\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op}\,{}^{B}\boldsymbol{I} \\ \boldsymbol{C}_{\dot{\omega}}^{\delta u} &= \boldsymbol{R}_{op}^{\top}\left[\hat{\boldsymbol{r}}_{op}^{1}, \hat{\boldsymbol{r}}_{op}^{2}, \hat{\boldsymbol{r}}_{op}^{3}, \hat{\boldsymbol{r}}_{op}^{4}\right]. \end{aligned} \qquad (2-20) $$

The discrete dynamics of $\boldsymbol{\omega}$ is then propagated using $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + dt \cdot \dot{\boldsymbol{\omega}}_k$.

Through the aforementioned series of transformations, the system state vector in formula (2-2) is converted into the following form:

$$ \boldsymbol{x}_{\boldsymbol{k}} \coloneqq [\boldsymbol{p}_{\boldsymbol{k}},\ \dot{\boldsymbol{p}}_{\boldsymbol{k}},\ \boldsymbol{\xi}_k,\ \boldsymbol{\omega}_k] \in \mathbb{R}^{12} \qquad (2-21) $$

The input vector is rewritten as the variation in the GRF of the four legs from the operating point $\boldsymbol{u}_{\boldsymbol{i,op}}$:

$$ \delta\boldsymbol{u}_{\boldsymbol{k}} \coloneqq \left[\delta\boldsymbol{u}_{\boldsymbol{1,k}},\ \delta\boldsymbol{u}_{\boldsymbol{2,k}},\ \delta\boldsymbol{u}_{\boldsymbol{3,k}},\ \delta\boldsymbol{u}_{\boldsymbol{4,k}}\right] \in \mathbb{R}^{12} \qquad (2-22) $$

The full input vector at horizon k+1 $\boldsymbol{u}_{k+1}$ can then be calculated as $\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + dt \cdot \dot{\boldsymbol{u}}_k$.

Using the new state and input vector, the dynamic function of the system (2-5) can be rewritten in discrete form as follows:

$$ \boldsymbol{x}_{t+k+1|t} = \boldsymbol{A}|_{op} \cdot \boldsymbol{x}_{t+k|t} + \boldsymbol{B}|_{op} \cdot \delta\boldsymbol{u}_{t+k|t} + \boldsymbol{d}|_{op} \qquad (2-23) $$

where $\boldsymbol{A}|_{op} \in \mathbb{R}^{n\times n}, \boldsymbol{B}|_{op} \in \mathbb{R}^{n\times m}, \boldsymbol{d}|_{op} \in \mathbb{R}^{n}$. The contents of these three matrices can be determined using the formulas derived in the early portion of this section and the last section. $t$ corresponds the operating point, and $k+1$ is the current to be predicted horizon.

Thus, the derivation of the optimization problem's equation (2-1b) has been completed thus far.

### 2.3.4 Cost Function

This section presents the derivation of equation (2-1a) in the optimization problem.

As previously stated, the cost function consists of two main components: the stage cost function and the terminal cost function. The derivation of the stage cost based on the current state vector $\boldsymbol{x}_{\boldsymbol{k}}$ and input vector $\boldsymbol{u}_{\boldsymbol{k}}$ is as follows:

$$ \ell(\boldsymbol{x}_k, \boldsymbol{u}_k) = \left\|\boldsymbol{x}_{\boldsymbol{k}} - \boldsymbol{x}_{\boldsymbol{d,k}}\right\|_{\boldsymbol{Q}_x}^{2} + \left\|\boldsymbol{u}_{\boldsymbol{k}} - \boldsymbol{u}_{\boldsymbol{d,k}}\right\|_{\boldsymbol{R}_{\boldsymbol{u}}}^{2} \qquad (2-24) $$

$\|\boldsymbol{x}\|_{\boldsymbol{Q}_x}^{2}$ is a shorthand notation for the matrix norm $\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{Q}\boldsymbol{x}$ and $\boldsymbol{Q}$ is a positive definite

weighting matrix. The $\boldsymbol{Q}_x$ and $\boldsymbol{R}_u$ matrices are diagonal square matrices with user-defined values on the diagonal. These values represent the relative weights of different error terms in the optimization process. The $\boldsymbol{Q}_x$ and $\boldsymbol{R}_u$ used by different gaits have different values. Appendix 1 provides a detailed value reference. $\boldsymbol{x}_{d,k}$ and $\boldsymbol{u}_{d,k}$ are the desired state and input vectors at the k-th predicated horizon. The determination of their values necessitates a consideration of the gait knowledge outlined in Chapter 2.2 and the associated desired foot trajectory. Different gaits require distinct desired states and input vectors. A more detailed account of the application of this element is provided in subsequent chapters.

The first part of formula (2-24) can be further decomposed into the following:

$$\left\|\boldsymbol{x}_k - \boldsymbol{x}_{d,k}\right\|_{\boldsymbol{Q}_x}^2 = \left\|\boldsymbol{e}_{\boldsymbol{p}_k}\right\|_{\boldsymbol{Q}_p}^2 + \left\|\boldsymbol{e}_{\dot{\boldsymbol{p}}_k}\right\|_{\boldsymbol{Q}_{\dot{p}}}^2 + \left\|\boldsymbol{e}_{\boldsymbol{R}_k}\right\|_{\boldsymbol{Q}_R}^2 + \left\|\boldsymbol{e}_{\boldsymbol{\omega}_k}\right\|_{\boldsymbol{Q}_\omega}^2 \quad (2-25)$$

where $\boldsymbol{Q}_p, \boldsymbol{Q}_{\dot{p}}, \boldsymbol{Q}_R, \boldsymbol{Q}_\omega$ are diagonal matrices with positive values on the diagonal, representing the weight of each state parameter in the optimization process and $\boldsymbol{Q}_x = diag(\boldsymbol{Q}_p, \boldsymbol{Q}_{\dot{p}}, \boldsymbol{Q}_R, \boldsymbol{Q}_\omega)$. $\boldsymbol{e}_{\boldsymbol{p}_k}, \boldsymbol{e}_{\dot{\boldsymbol{p}}_k}, \boldsymbol{e}_{\boldsymbol{R}_k}, \boldsymbol{e}_{\boldsymbol{\omega}_k}$ denote the errors between each state parameter and the corresponding desired state parameter, which is expressed as follows:

$$\boldsymbol{e}_{\boldsymbol{p}_k} = \boldsymbol{p}_k - \boldsymbol{p}_{d,k} \quad (2-26)$$

$$\boldsymbol{e}_{\dot{\boldsymbol{p}}_k} = \dot{\boldsymbol{p}}_k - \dot{\boldsymbol{p}}_{d,k} \quad (2-27)$$

$$\boldsymbol{e}_{\boldsymbol{R}_k} = \boldsymbol{log}(\boldsymbol{R}_{d,k}^V \cdot \boldsymbol{R}_{op})^V + \boldsymbol{\xi}_k \quad (2-28)$$

$$\boldsymbol{e}_{\boldsymbol{\omega}_k} = \boldsymbol{\omega}_k - \boldsymbol{R}_k^T \cdot \boldsymbol{R}_{d,k}\boldsymbol{\omega}_{d,k} \quad (2-29)$$

It should be noted that in formula (2-29), the current centroid angular velocity $\boldsymbol{\omega}_k$ and the desired centroid angular velocity $\boldsymbol{\omega}_{d,k}$ are based on different reference frames. Therefore, it is necessary to first convert these two values to the same reference frame and then to calculate the error function.

The second part of formula (2-24) can also be further decomposed into the following:

$$\left\|\boldsymbol{u}_k - \boldsymbol{u}_{d,k}\right\|_{\boldsymbol{R}_u}^2 = \left\|\boldsymbol{u}_{op} + \delta\boldsymbol{u}_k - \boldsymbol{u}_{d,k}\right\|_{\boldsymbol{R}_u}^2 \quad (2-30)$$

The solution to the terminal cost function adheres to a similar procedure as formula (2-25), with the exception that the terminal cost function does not contain the input vector error term in comparison with the stage cost function.

### 2.3.5 Force Constraints

In addition to the aforementioned constraints, the optimization problem must also account for the system input resulting from the solution. Specifically, the physical validity of the ground reaction force at the foot must be ensured. Consequently, the ground reaction force constraint must be incorporated into the optimization problem, which stipulates that the vertical component of the ground reaction force must point upwards when the foot is in contact with the ground. Additionally, the ground reaction force in all three directions must satisfy the friction cone constraint:

$$\{\boldsymbol{u_i}|\boldsymbol{u_i^n} > \boldsymbol{0}, \|\boldsymbol{u_i^t}\|_2 \leq \mu|\boldsymbol{u_i^n}|\} \qquad (2-31)$$

where $\mu$ represents the coefficient of friction, and the superscripts $(\cdot)^t$ and $(\cdot)^n$ represent tangential and normal force components, respectively. $\|\cdot\|_2$ means the 2-Norm and $|\cdot|$ computes the absolute value.

The 2-Norm is a quadratic nonlinear term in the force constraint formulation; thus, it must be linearized to match the QP. An approximation method for the friction cone constraints that satisfies the linearizability requirement is introduced as a conservative friction pyramid [56].

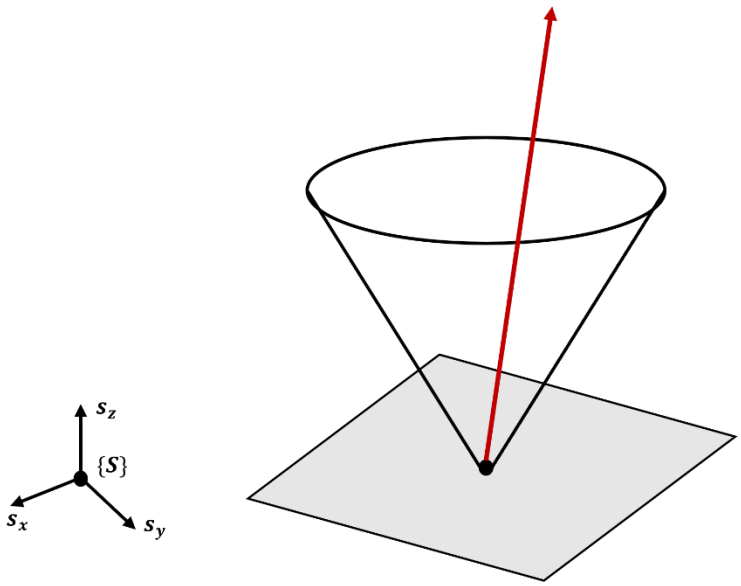


*Figure 2.4: Illustration of the conservative friction cone, where the GRF direction is represented by a red arrow. To prevent object slippage, the red arrow must remain within the friction cone.*

With this method, the variation of the input vector can be expressed as follows:

$$\mathbb{U}_i := \left\{ \delta\boldsymbol{u}_i \middle| \begin{matrix} \left|u_{i,op}^{x,y} + \delta u_{i,k}^{x,y}\right| \leq \mu\left|u_{i,op}^{z} + \delta u_{i,k}^{z}\right|, \\ u_{i,k}^{z,min} \leq u_{i,op}^{z} + \delta u_{i,k}^{z} \leq u_{i,k}^{z,max}, \\ u_{i,k}^{z,min} \geq 0 \end{matrix} \right\} \qquad (2-32)$$

With $\mathbb{U} := [\mathbb{U}_1,\ \mathbb{U}_2,\ \mathbb{U}_3,\ \mathbb{U}_4] \in \mathbb{R}^{12}$. The symbol "$z$" in the upper-left corner represents the GRF component along the z-axis, which corresponds to the "$n$" in the formula (2-31). The symbols "$x$" and "$y$" in the upper-left corner represent the GRF components along the x and y axes, respectively, which correspond to the "$t$" in the formula (2-31). Additionally, the GRF component along the z-axis must satisfy the non-negativity constraint $u_{i,k}^{z,min} \geq 0$ and be lower than a specified maximum value of $u_{i,k}^{z,max}$. When the feet are not in contact with the ground, the upper and lower limits of the GRF are both set to 0 to enforce that the optimized GRF is also 0.

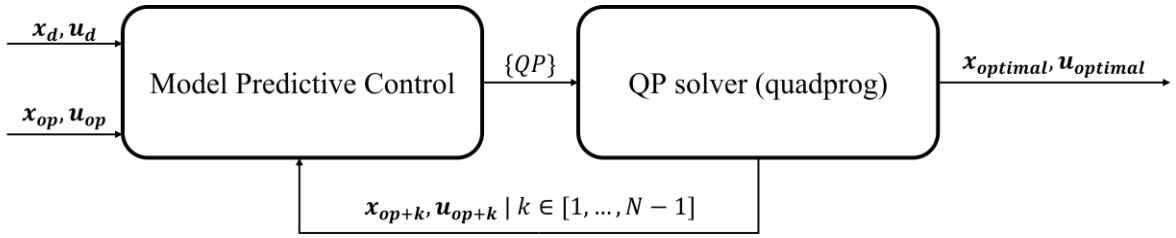


*Figure 2.5 : The present flow chart depicts the solution process for the MPC optimization problem, where N denotes the predetermined predicate horizon size.*

### 2.3.6 Formulation of RF-MPC

After the derivation from Section 2.3.2 to 2.3.5, the general expression (2-1) of MPC can be transformed into the following expression of RF-MPC:

$$\text{minimize} \quad \gamma^N \ell_T(\boldsymbol{x}_{t+N|t}) + \sum_{k=0}^{N-1} \gamma^N \ell(\boldsymbol{x}_{t+k|t}, \boldsymbol{u}_{t+k|t}) \tag{2-33a}$$

$$\text{subject to} \quad \boldsymbol{x}_{t+k+1|t} = \boldsymbol{A}\boldsymbol{x}_{t+k|t} + \boldsymbol{B}\delta\boldsymbol{u}_{t+k|t} + \boldsymbol{d} \tag{2-33b}$$

$$k = 0, \cdots, N-1 \tag{2-33c}$$

$$\delta\boldsymbol{u}_{t+k|t} \in \mathbb{U} \tag{2-33d}$$

$$\boldsymbol{x}_{t|t} = \boldsymbol{x}(t) = \boldsymbol{x}_{op} \tag{2-33e}$$

The cost function here is multiplied by a decay rate factor of $\gamma \in (0,1]$, which is utilized for tuning the weights of predictions that are more distant from the present moment.

Although the current optimization equation does not have an explicit constraint formula for the state vector, the cost function optimization process implicitly constrains the state vector.

If the prediction horizon of MPC is set to 6, i.e., each invocation of MPC solves the state and input vectors for the next six time steps, then, according to the existing MPC equation (2-33), six optimization processes must be executed, as depicted in Figure 2.4, which requires significant computational resources. According to the formulation proposed in [57], the six-step MPC optimization equations can be combined into a single integrated optimization equation, wherein the state and input vectors for all six prediction horizons can be directly solved through a single MPC optimization, as illustrated in Figure 2.5. By adopting this approach, the variables in the optimization equation can be rewritten as follows:

$$\boldsymbol{z} \coloneqq \left[\delta\boldsymbol{u}_0{}^T,\ \mathbf{x_1}{}^T, \cdots, \delta\boldsymbol{u}_{N-1}{}^T,\ \mathbf{x_N}{}^T\right]^T \in \mathbb{R}^{24N} \tag{2-34}$$

where $N$ refers to the predicate horizon. The state vector and input vector of the original MPC optimization problem are integrated into the variables of the new optimization problem. The original optimization problem has 12 state parameters (2-21) and 12 input parameters (2-22) in each predicate horizon. Therefore, the synthesized parameter vector $\mathbf{z}$ has a total of $24N$ parameters.

According to [57], the new optimization problem can be rewritten into the standard QP form [58]:

$$\text{minimize} \quad \frac{1}{2}\boldsymbol{z^T P z} + \boldsymbol{c^T z} \qquad (2-35a)$$

$$\text{subject to} \quad \boldsymbol{A_{ineq} \cdot z \leq b_{ineq}} \qquad (2-35b)$$

$$\boldsymbol{A_{eq} \cdot z = b_{eq}} \qquad (2-35c)$$

where $\boldsymbol{P} \in \mathbb{R}^{N(n+m)}$ is a symmetric positive definite matrix assembled from the gain matrices $\boldsymbol{Q_x}$ and $\boldsymbol{R_u}$; the inequality constraint $\boldsymbol{A_{ineq} \cdot z \leq b_{ineq}}$ imposes the force constraints; and the equality constraint $\boldsymbol{A_{eq} \cdot z = b_{eq}}$ respects the linear dynamics. It must be emphasized that $N$ is the prediction horizon, $n$ is the number of state variables, and $m$ is the number of input variables.

The standard QP form can be directly solved via the linear optimization method "quadprog" provided by MATLAB. In Python, multiple Python libraries, such as "scipy.optimize.minimum", "qpsolvers.solve_qp", and "CVXOPT", also offer QP solution algorithms. Among them, the library "scipy.optimize.minimum" is a more general optimization solver. Therefore, when employing this library to solve QP, specific settings for the solver must be applied.

During practical usage, it has been observed that the solution time for calling the existing QP solver is unacceptable for realizing dynamic gaits when the predicate horizon N is set to 6, resulting in a total of 144 variables in the RF-MPC. The time required for an iteration of the QP solution is about 0.5-0.6s, while the gait frequency of quadrupeds can reach 3-5Hz or even higher frequencies. Moreover, the time required for quadrupeds to complete a gait cycle is even less than the time required for a one-time QP solution.

Furthermore, the parameter matrix $\boldsymbol{A_{eq}}$ in equation (2-35c) has dimensions of 144 x 144 when the prediction horizon is set to 6. However, the elements with non-zero values in $\boldsymbol{A_{eq}}$ are primarily concentrated near the diagonal, while the elements in other positions are predominantly 0. Similar circumstances apply to other parameter matrices. As a result, utilizing a general QP solver to solve a QP problem with this type of structure would result in a considerable number of redundant calculations.

Therefore, it is imperative to identify a more powerful solution method for solving RF-MPC.

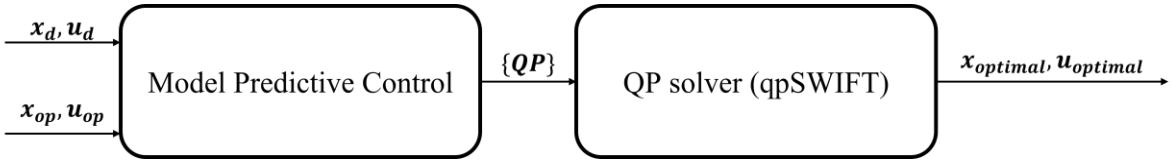


*Figure 2.6: Above is the MPC flowchart after incorporating the formulation proposed in [57]. Compared to Figure 2.4, the optimization problem for the entire predicate horizon N of 6 can now be solved by calling the new QP solver, qpSWIFT, only once.*

## 2.4 QP Solver qpSWIFT

qpSWIFT is a QP optimization solver designed specifically for controlling quadruped robots. Its approach involves optimizing and solving QP problems by extracting the effective portion in the sparse parameter matrices, thus averting the need to solve the complete sparse matrix and resulting in significant computational savings.

qpSWIFT was proposed on the paper [59], and it employs the Primal-Dual Interior-Point Method with Mehrotra predictor-corrector steps along with Nesterov-Todd scaling. The Primal-Dual Interior-Point Method requires the solution of the Karush-Kuhn-Tucker (KKT) linear system during each iteration. However, the KKT linear system obtained through the original QP problem is exceptionally sparse in most cases, resulting in many meaningless operations during the solution process. To solve this problem, sparse Cholesky factorization [56] and dynamic regularization are used to factorize symmetric positive semi-definite matrices and to prevent fatal errors, such as division by zero. Furthermore, to enhance the suitability of the sparse Cholesky factorization algorithm for optimal control of quadruped robots, [59] continues to conduct partial optimization on sparse Cholesky factorization.

In the final experimental section of the study [59], qpSWIFT was compared with mainstream QP solvers, including quadprog, OSQP, ECOS, and others, to solve the same optimization problems such as portfolio optimization, SVM problems, and MPC problems. The results demonstrated that qpSWIFT achieved convergence with fewer iterations and in less time compared to other QP solvers. Notably, when solving MPC problems, qpSWIFT's performance ratio was almost an order of magnitude higher than that of the quadprog solver. These findings establish qpSWIFT as an efficient QP solver for real-time sparse QP problems.

Furthermore, qpSWIFT was applied to physically small quadruped robots to realize various gaits, such as pose control, walking trot, and bounding. As a result, qpSWIFT demonstrated excellent performance in gait control, especially in relation to the bounding gait, for which the maximum single solution time of qpSWIFT was less than 4.5ms, and it achieved a control frequency of 160Hz, confirming its suitability for solving RF-MPC and robot gait control tasks in this study.

The source code of qpSWIFT has been released in open-source format on GitHub, enabling its installation in various programming languages such as C++, Python, and MATLAB. By utilizing qpSWIFT to solve the RF-MPC problem in place of “scipy.optimize.minimum” in this paper, the entire cycle time in one control loop can be restricted to 0.01s. Although this may not enable an optimization time as low as that attained in [59], it is adequate for the robot control task addressed in this study.

# 3 Kinematic and Dynamic

The previous chapter described RF-MPC, one of the current mainstream MPC control algorithms, and the corresponding QP solution algorithm, namely qpSWIFT. Through the combination of these two algorithms, the expected system input $\boldsymbol{u_{optimal}}$ can be determined according to the current state and input of the robot $\boldsymbol{x_{op}}, \boldsymbol{u_{op}}$ and the expected state and input of the robot $\boldsymbol{x_d}, \boldsymbol{u_d}$, as illustrated in Figure 2.6. This expected system input is expressed as the GRF that the feet of the four legs must be subjected to at the current sampling time.

However, this result cannot be directly used for controlling real quadruped robots and quadruped robots in physical simulation platforms. It is also necessary to calculate the torque output of the motors that control the movement of the legs. The computation of the torque is based on the leg structure of the quadruped robot and the GRF required at the foot. After the computation, the calculated torque output will be transmitted to the control motor through the chip to finally realize control over the robot's motion.

The calculation from GRF to the motor torque must use the kinematic formula of the leg, which is the primary subject matter introduced in this chapter.

This chapter presents the computation of the kinematic and dynamic equations for the two-link leg, as well as the kinematic equations for the leg structure and the spine with two DOFs, as currently adopted by Nermo.

Notably, the leg structure employed in the quadruped robot with the simple structure as depicted in Figure 2.3, which utilizes RF-MPC for realizing gait control in the MuJoCo simulation environment, is a basic two-link leg. Only leg kinematics equations are currently employed in the control loop in this paper. However, as the RF-MPC primarily controls the characteristics of central dynamic models and therefore only provides the GRF received by the legs, the leg motion control is not directly involved, and the control motor torque must be further computed. Hence, replacing the leg structure and the corresponding control code is not a challenging task.

In addition, as of now, the leg dynamics equations and spine kinematics equations have not been incorporated into the control loop. The reason for introducing the derivation process of these equations in this chapter is that they were developed by me and my colleague Yuhong Huang, who is part of the same research group. In future work, these equations may be added to the control loop to tackle more complex motion control issues.

## 3.1 Two-DOF Two-Link Leg

Most of the existing designs for the leg structure of quadruped robots are based on a two-link model, which includes the MIT Cheehah 3 [60], various types of quadruped robots from Unitree, Boston Dynamic Spot, etc. In this design, the upper link, or the thigh, is driven directly by a motor located at the shoulder or hip joints, while the lower link is driven by another motor also located at the shoulder or hip joints through hinge transmission. This design helps to concentrate the weight of the entire robot on the torso, thereby reducing the moment of inertia of the robot and increasing the hardware's durability.

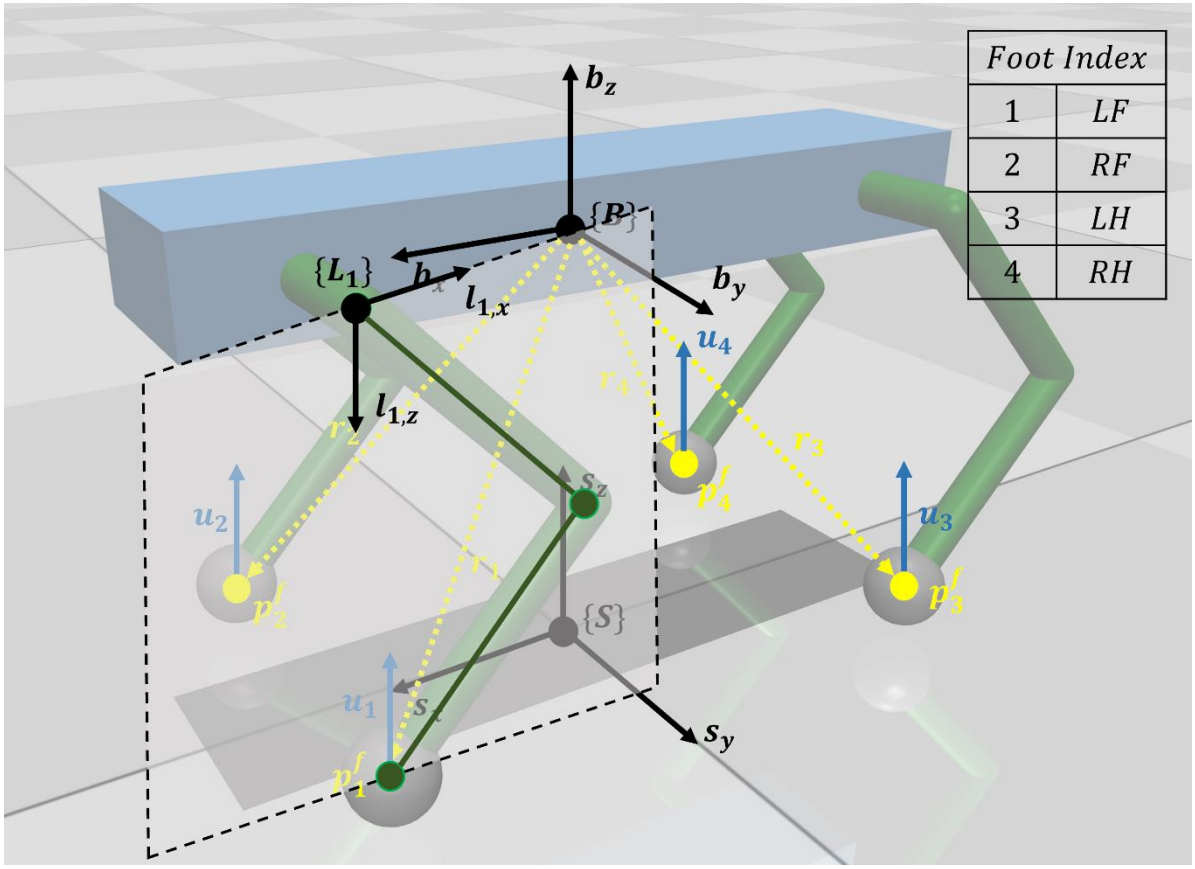


*Figure 3.1: To facilitate the derivation of the leg kinematic equations, leg coordinate systems $\{\boldsymbol{L_i}\}$ are introduced in addition to Figure 2.3. The leg structure and joint positions are represented by dark green lines and points in the leg coordinate system $\{\boldsymbol{L_i}\}$. The origin of $\{\boldsymbol{L_i}\}$ is located at the shoulder or hip joints, and its $\boldsymbol{l}_{i,x}$ and $\boldsymbol{l}_{i,z}$ directions are opposite to the $\boldsymbol{b}_x$ and $\boldsymbol{b}_z$ directions of the body coordinate system $\{\boldsymbol{B}\}$, respectively. Each leg has an independent leg coordinate system. In this figure, only the leg coordinate system of the left front leg is drawn, and other leg coordinate systems can be obtained by analogously.*

When constructing a quadruped robot model in MuJoCo, the motor design can be omitted, and the weight of the motor can be added to the robot's torso model. The leg model is simply designed using two "capsule" models, with a "sphere" model added to the lower end of the lower link to improve contact with the ground.

The current two-link design only provides two DOFs, which allow the legs to move in the $\boldsymbol{b}_x$ and $\boldsymbol{b}_z$ directions of the body coordinate system $\{\boldsymbol{B}\}$. To better describe leg movement on a two-dimensional plane, leg coordinate systems $\{\boldsymbol{L_i}\}$ are established based on the $\boldsymbol{b}_x$ and $\boldsymbol{b}_z$ directions of the body coordinate system, with the directions $\boldsymbol{l}_{i,x}$ and $\boldsymbol{l}_{i,z}$ opposite to the $\boldsymbol{b}_x$ and $\boldsymbol{b}_z$ directions, respectively. The following two-dimensional two-link leg kinematics derivation will be performed on the leg coordinate system $\{\boldsymbol{L_i}\}$. The conversion between $\{\boldsymbol{L_i}\}$ and $\{\boldsymbol{B}\}$ only requires adding or subtracting the length in the $\boldsymbol{b}_y$ direction, which is the length from the shoulder or hip joints to the body COM in the $\boldsymbol{b}_y$ direction. For SRB, this length remains constant.

The leg plane is extracted separately, and the effect picture is presented in Figure 3.2:

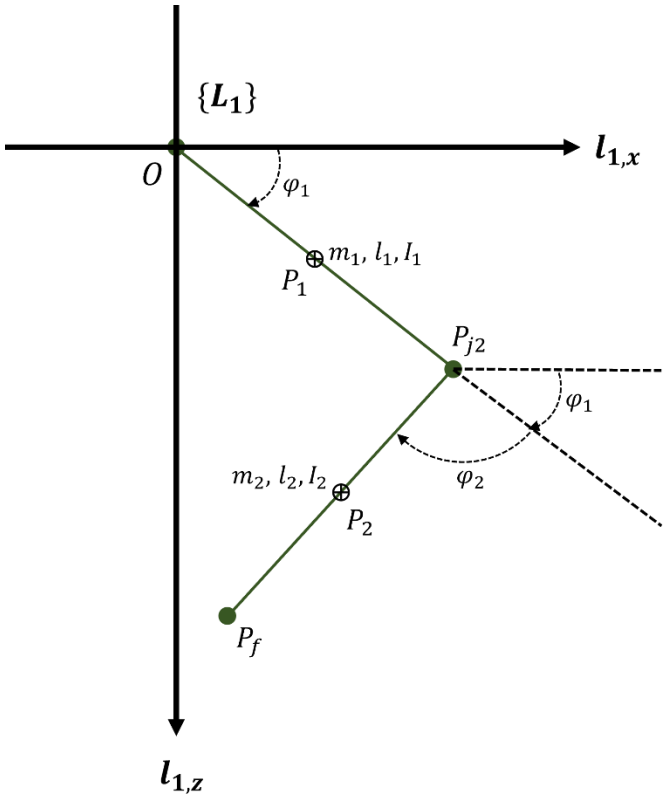


*Figure 3.2: Schematic diagram of the two-link leg structure in the leg coordinate system $\{\boldsymbol{L_1}\}$, where the knee joint is located at $P_{j2}$, the foot position is $P_f$, and the origin of the coordinate system is set to the shoulder or hip joints and denoted as $O$. The masses of the upper and lower connecting rods are $m_1$ and $m_2$ respectively, the COM position are $P_1$ and $P_2$, the lengths are $l_1$ and $l_2$, and the moments of inertia are $I_1$ and $I_2$. Assuming that all links are homogeneous, the COMs of the links are located in the middle of the links. The angle between the upper link and the positive semi-axis of $\boldsymbol{l}_{1,x}$ is denoted as $\varphi_1$, while the angle between the lower link and the upper link is represented by $\varphi_2$. To prevent the occurrence of the anti-joint phenomenon, $\varphi_2$ is constrained to a value greater than 0.*

### 3.1.1 Forward Kinematics of the Two-DOF Two-Link Leg

There are many different methods to solve the kinematics of the two-link structure, here the algebraic method is used, which is the simplest and most direct method. The foot position in relation to the origin can then be calculated as follows:

$$x_f = l_1\, cos(\varphi_1) + l_2\, cos(\varphi_1 + \varphi_2) \qquad (3-1)$$

$$z_f = l_1\, sin(\varphi_1) + l_2\, sin(\varphi_1 + \varphi_2) \qquad (3-2)$$

where $x_f$ and $z_f$ represent the horizontal and vertical coordinates of $P_f$.

The forward kinematics method enables the determination of the position of the foot relative to the shoulder or hip joints given the joint rotation angle. Moreover, it has extensive applications in solving inverse kinematics, Jacobian matrices, and dynamics, which are elaborated upon in the subsequent sections and chapters.

### 3.1.2 Inverse Kinematics of the Two-DOF Two-Link Leg

In certain scenarios, for instance, in the PD control of the foot-end position, it may be necessary to determine the desired joint rotation angle based on the known relative position of the foot with respect to the shoulder or hip joint. Thus, this section introduces the inverse kinematics derivation of the two-link leg.

Both sides of equations (3-1) and (3-2) should be squared and added together, and the sum and difference angle formulas of the following trigonometric functions should be utilized:

$$cos(\varphi_1 + \varphi_2) = cos(\varphi_1)\, cos(\varphi_2) - sin(\varphi_1)\, sin(\varphi_2) \qquad (3-3)$$

$$sin(\varphi_1 + \varphi_2) = sin(\varphi_1)\, cos(\varphi_2) + cos(\varphi_1)\, sin(\varphi_2) \qquad (3-4)$$

The following expression can thereby be derived:

$$cos(\varphi_2) = \frac{x_f^2 + y_f^2 - l_1^2 - l_2^2}{2 l_1 l_2} \qquad (3-5)$$

Notably, the cosine function has a value range of $[-1,1]$; thus, verifying the result of $\cos(\varphi_2)$ is crucial during the computation. If the right-hand-side of equation (3-5) exceeds the value range of cos(x), it indicates that the expected foot position is beyond the accessible range of the current leg structure. Therefore, it is essential to respond appropriately to this situation, such as reporting an error and then terminating the program or selecting the closest value in the value range of $\cos(\varphi_2)$. The closest value of the operation's result on the right side of equation (3-5) can be -1 or 1, which corresponds to the fully extended and fully flexed state of the two-link structure's legs, respectively.

If the above issues do not persist, the solution process can be further pursued utilizing equation (3-5):

$$sin(\varphi_2) = \pm\sqrt{1 - cos^2(\varphi_2)} \qquad (3-6)$$

As the aim of the current calculation is to obtain the joint rotation angle of the leg link, the occurrence of the anti-joint phenomenon is not anticipated. Therefore, it is assumed that the value of $\varphi_2$ is greater than 0, which implies that the sign of $sin(\varphi_2)$ must be positive:

$$sin(\varphi_2) = \sqrt{1 - cos^2(\varphi_2)} \qquad (3-7)$$

The tangent of $\varphi_2$ can be obtained via formulas (3-6) and (3-7):

$$tan(\varphi_2) = \frac{sin(\varphi_2)}{cos(\varphi_2)} \qquad (3-8)$$

Subsequently, the value of $\varphi_2$ can be determined by the arctangent function. Notably, both MATLAB and Python provide two methods for calculating the arctangent function: $atan()$ and $atan2()$. The value range of $atan()$ is $(-\frac{\pi}{2}, \frac{\pi}{2})$, while the value range of $atan2()$ is $(-\pi, \pi]$. The feasible range of $\varphi_2$ in Figure 3.2 should be $[0, \pi]$, therefore, $atan2()$ should be selected to solve $\varphi_2$. In MATLAB, $atan2()$ can be directly called, whereas in Python, the "math" library must be called to use $atan2()$.

$$\varphi_2 = atan2(sin(\varphi_2), cos(\varphi_2)) \qquad (3-9)$$

After obtaining the value of $\varphi_2$, the calculation of $\varphi_1$ is also necessary. By applying the algebraic method, equations (3-1) and (3-2) can be rewritten as follows:

$$x_f = k_1\, cos(\varphi_1) - k_2\, sin(\varphi_1) \tag{3-10}$$

$$z_f = k_1\, sin(\varphi_1) + k_2\, cos(\varphi_1) \tag{3-11}$$

where

$$k_1 = l_1 + l_2\, cos(\varphi_2) \tag{3-12}$$

$$k_2 = l_2\, sin(\varphi_2) \tag{3-13}$$

Subsequently, variable substitution techniques will be utilized to replace variables for $k_1$ and $k_2$:

$$k_1 = r\, cos(\gamma) \tag{3-14}$$

$$k_2 = r\, sin(\gamma) \tag{3-15}$$

where

$$r = \sqrt{k_1^2 + k_2^2} \tag{3-16}$$

$$\gamma = atan2(k_2, k_1) \tag{3-17}$$

With equations (3-12) through (3-17), the equations (3-10) and (3-11) can be rewritten as follows:

$$\frac{x_f}{r} = cos(\gamma)\, cos(\varphi_1) - sin(\gamma)\, sin(\varphi_1) \tag{3-18}$$

$$\frac{z_f}{r} = cos(\gamma)\, sin(\varphi_1) + sin(\gamma)\, cos(\varphi_1) \tag{3-19}$$

Through the sum and difference angle formulas, (3-18) and (3-19) can then be simplified as follows:

$$\frac{x_f}{r} = cos(\gamma + \varphi_1) \tag{3-20}$$

$$\frac{z_f}{r} = sin(\gamma + \varphi_1) \tag{3-21}$$

By using the function $atan2()$, the following equation can be obtained:

$$\gamma + \varphi_1 = atan2\left(z_f, x_f\right) \tag{3-22}$$

The first joint rotation angle $\varphi_1$ can then be calculated by introducing equations (3-17), (3-12) and (3-13) into (3-22):

$$\varphi_1 = atan2\left(z_f, x_f\right) - atan2(l_2\, sin(\varphi_2)\,, l_1 + l_2\, cos(\varphi_2)) \tag{3-23}$$

In summary, equations (3-9) and (3-23) can be employed to accomplish the solution from the given relative position of the foot to the demanded joint angle, namely inverse kinematics.

### 3.1.3 General Dynamics

Prior to initiating the derivation of the dynamics of the two-link leg, this section presents an introduction to the Lagrangian dynamics modeling methodology in general.

For a multi-component system, the position of the COM of each component $i$ can be expressed as $\boldsymbol{p}_i \in \mathbb{R}^{2\ or\ 3}$. The length of $\boldsymbol{p}_i$ corresponds to the system's dimension, in Figure 3.2 there are only two dimensions, namely $\boldsymbol{p}_i = [p_{i,x}, p_{i,z}]^T$. The independent DOFs of the system (i.e., the generalized coordinates) can be represented by a vector $\boldsymbol{q}$.

In the case of a two-link leg system, the two links constitute the mass, and there are no further masses at the joints. Therefore, $i = 1{,}2$ and $\boldsymbol{p}_i$ represents the position of the COM of the two links. Two joints can move independently with $\boldsymbol{q} = [q_1, q_2]^{\boldsymbol{T}}$, namely the shoulder or hip joint and the knee joint. Thus, the corresponding generalized coordinates are joint rotation angles, namely $q_1 = \varphi_1$ and $q_2 = \varphi_2$.

Based on the above information, the mapping from the generalized coordinates to the positions of the i-th COM (i.e., the position Jacobian matrix) can be expressed as follows:

$$\boldsymbol{J_{p,i}} = \left[\frac{\partial \boldsymbol{p}_i}{\partial q_1}, \frac{\partial \boldsymbol{p}_i}{\partial q_2}, \dots, \frac{\partial \boldsymbol{p}_i}{\partial q_k}\right] \in \mathbb{R}^{2\times k\ or\ 3\times k} \qquad (3-24)$$

The relationship from the generalized coordinates to the i-th COM position can be expressed as follows:

$$\boldsymbol{p_i} = \boldsymbol{J_{p,i}} \cdot \boldsymbol{q} \qquad (3-25)$$

In addition, the position Jacobian matrix is the same as the velocity Jacobian matrix:

$$\boldsymbol{J_{v,i}} = \boldsymbol{J_{p,i}} \qquad (3-26)$$

The relationship between the linear velocity of the i-th COM and the velocity of the generalized coordinates can then be expressed as follows:

$$\boldsymbol{v_i} = \boldsymbol{J_{v,i}} \cdot \dot{\boldsymbol{q}} \qquad (3-27)$$

As the generalized coordinate of the two-link leg is the joint rotation angle, its variation affects not only the translation but also the rotation of the links. Therefore, it is necessary to establish the Jacobian matrix of angular velocity $\boldsymbol{J_{\omega,i}}$, which represents the relationship between the rotational angular velocity of the i-th COM of the links and the velocity of the generalized coordinate.

For translational joints, the corresponding entries in the angular velocity Jacobian matrix are zero as they do not affect the rotation of the links. Conversely, for rotary joints, their angular velocity corresponds to the rotational angular velocity of the links. The construction and size of the angular velocity Jacobian matrix depend on the specific system. In the case of the two-link leg, the following expression represents its angular velocity Jacobian matrix:

$$\boldsymbol{J_{\omega,1}} = [1 \quad 0], \qquad \boldsymbol{J_{\omega,2}} = [1 \quad 1] \qquad (3-28)$$

Therefore, the relationship between the angular velocity of the COM and the velocity of the generalized coordinates can be formulated as follows:

$$\boldsymbol{\omega}_i = \boldsymbol{J_{\omega,i}} \cdot \dot{\boldsymbol{q}} \qquad (3-29)$$

By utilizing the pertinent velocity information, it is possible to subsequently derive the kinetic energy of each constituent within the system:

$$\begin{aligned} k_i &= \frac{1}{2} m_i \boldsymbol{v_i}^T \boldsymbol{v_i} + \frac{1}{2} I_i \boldsymbol{\omega_i}^T \boldsymbol{\omega_i} \\ &= \frac{1}{2} m_i \dot{\boldsymbol{q}}^T \boldsymbol{J_{v,i}}^T \boldsymbol{J_{v,i}} \dot{\boldsymbol{q}} + \frac{1}{2} I_i \dot{\boldsymbol{q}}^T \boldsymbol{J_{\omega,i}}^T \boldsymbol{J_{\omega,i}} \dot{\boldsymbol{q}} \end{aligned} \quad (3-29)$$

Thus, the total kinetic energy of the system is as follows:

$$\begin{aligned} \boldsymbol{K} = \Sigma_i k_i &= \frac{1}{2} \dot{\boldsymbol{q}}^T \left[ \Sigma_i \left( m_i \boldsymbol{J_{v,i}}^T \boldsymbol{J_{v,i}} + I_i \boldsymbol{J_{\omega,i}}^T \boldsymbol{J_{\omega,i}} \right) \right] \dot{\boldsymbol{q}} \\ &= \frac{1}{2} \dot{\boldsymbol{q}}^T M \dot{\boldsymbol{q}} \end{aligned} \quad (3-30)$$

with

$$M(\boldsymbol{q}) = \Sigma_i \left( m_i \boldsymbol{J_{v,i}}^T \boldsymbol{J_{v,i}} + I_i \boldsymbol{J_{\omega,i}}^T \boldsymbol{J_{\omega,i}} \right) \quad (3-31)$$

$M$ is the mass matrix in the standard kinetic equation. Similarly, the total gravitational potential energy of the system can be represented as follows:

$$\boldsymbol{U} = \Sigma_i u_i = -m_1 \boldsymbol{g}^T \boldsymbol{P}_i(\boldsymbol{q}) \quad (3-32)$$

The gravitational acceleration vector $\boldsymbol{g}$ has a structure of $\boldsymbol{g} = [0\ g]^T or\ [0\ 0\ g]^T$, its length is dependent upon the dimension of the analyzed system, and $\boldsymbol{P}_i$ represents the position of the COM of the i-th component, which is also a function of the generalized coordinates.

Subsequently, the Lagrangian dynamic equation is introduced:

$$\boldsymbol{\mathcal{L}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{K}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{U}(\boldsymbol{q}) \quad (3-33)$$

$$\frac{d}{dt} \frac{\partial \boldsymbol{\mathcal{L}}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial \boldsymbol{\mathcal{L}}}{\partial \boldsymbol{q}} = \frac{d}{dt} \frac{\partial \boldsymbol{K}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial \boldsymbol{K}}{\partial \boldsymbol{q}} - \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{q}} = \boldsymbol{\tau} \quad (3-34)$$

Among these, $\boldsymbol{\tau}$ represents the external forces acting on the generalized coordinates. For instance, in the case of a two-link leg, $\boldsymbol{\tau}$ represents the torque applied by the motor to the two joints.

By substituting equations (3-30) and (3-32) into the terms in equation (3-34), the following can be obtained:

$$\frac{\partial \boldsymbol{K}}{\partial \dot{\boldsymbol{q}}} = \frac{\partial}{\partial \dot{\boldsymbol{q}}} \left( \frac{1}{2} \dot{\boldsymbol{q}}^T M \dot{\boldsymbol{q}} \right) = M \dot{\boldsymbol{q}} \quad (3-35)$$

$$\frac{d}{dt} \frac{\partial \boldsymbol{K}}{\partial \dot{\boldsymbol{q}}} = M \ddot{\boldsymbol{q}} + \dot{M} \dot{\boldsymbol{q}} \quad (3-36)$$

$$\frac{\partial \boldsymbol{K}}{\partial \boldsymbol{q}} = \frac{1}{2} \begin{bmatrix} \dot{\boldsymbol{q}}^T \dfrac{\partial M}{\partial q_1} \dot{\boldsymbol{q}} \\ \vdots \\ \dot{\boldsymbol{q}}^T \dfrac{\partial M}{\partial q_n} \dot{\boldsymbol{q}} \end{bmatrix} \quad (3-37)$$

Subsequently, several matrices can be defined:

$$V(\boldsymbol{q},\dot{\boldsymbol{q}}) = \dot{M}\dot{\boldsymbol{q}} - \frac{1}{2}\begin{bmatrix} \dot{\boldsymbol{q}}^T \dfrac{\partial M}{\partial q_1}\dot{\boldsymbol{q}} \\ \vdots \\ \dot{\boldsymbol{q}}^T \dfrac{\partial M}{\partial q_n}\dot{\boldsymbol{q}} \end{bmatrix} \quad (3-38)$$

$$G(\boldsymbol{q}) = -\frac{\partial \boldsymbol{U}}{\partial \boldsymbol{q}} \quad (3-39)$$

Thus, the final expression of the Lagrangian dynamic equation is as follows:

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + V(\boldsymbol{q},\dot{\boldsymbol{q}}) + G(\boldsymbol{q}) = \boldsymbol{\tau} \quad (3-40)$$

### 3.1.4 Foot Jacobian Matrix of the Two-DOF Two-Link Leg

Before deriving the dynamic equations of the two-link leg, another important Jacobian matrix, namely the foot Jacobian matrix, must first be introduced thereby enabling the mapping from the GRF to the joint torques.

Firstly, the coordinate position of the foot end $P_f(X_f, Z_f)$ is obtained as follows:

$$X_f = l_1\cos(q_1) + l_2\cos(q_1+q_2) \quad (3-41)$$

$$Z_f = l_1\sin(q_1) + l_1\sin(q_1+q_2) \quad (3-42)$$

Approximate to the derivation process of formulas (3-24) through (3-27), the Jacobian matrix from the generalized coordinates to the foot position is as follows:

$$\boldsymbol{J_{p,f}} = \boldsymbol{J_{v,f}} = \begin{bmatrix} \dfrac{\partial X_f}{\partial q_1} & \dfrac{\partial X_f}{\partial q_2} \\ \dfrac{\partial Z_f}{\partial q_1} & \dfrac{\partial Z_f}{\partial q_2} \end{bmatrix} = \begin{bmatrix} -l_1\sin(q_1) - l_1\sin(q_1+q_2) & -l_1\sin(q_1+q_2) \\ l_1\cos(q_1) + l_2\cos(q_1+q_2) & l_2\cos(q_1+q_2) \end{bmatrix} \quad (3-43)$$

The differential form expressing the relationship between the velocity of the foot end and the velocity of the generalized coordinate can be calculated via the following:

$$\delta\boldsymbol{p_f} = \boldsymbol{J_{v,f}} \cdot \delta\boldsymbol{q} \quad (3-44)$$

Based on the principle of virtual work and the conservation of momentum (i.e., when the mass of the leg is less than 10% of the mass of the entire body) the leg's energy variation can be neglected [16]. In this case, the energy input from the motor though leg joint to the leg system is equal to the energy transferred by the leg system to the ground, and the following equation can be obtained:

$$\boldsymbol{\tau}^T \cdot \delta\boldsymbol{q} + (-\boldsymbol{F})^T \cdot \delta\boldsymbol{p_f} = 0 \quad (3-45)$$

Among them, $\boldsymbol{F}$ refers to the force of the foot against the ground or the reaction force of GRF. The relationship between GRF and the joint torque can then be obtained:

$$\boldsymbol{\tau}^T = \frac{\delta\boldsymbol{p_f}}{\delta\boldsymbol{q}} \cdot \boldsymbol{F}^T = \boldsymbol{J_{v,f}} \cdot \boldsymbol{F}^T \quad (3-46)$$

and

$$\boldsymbol{\tau} = \boldsymbol{J}_{v,f} \cdot \boldsymbol{F} \tag{3-47}$$

### 3.1.5 Dynamics of the Two-DOF Two-Link Leg

This section introduces concrete dynamic equation of the two-link leg based on the schematic diagram presented in Figure 3.2.

In the case of the two-link leg, the generalized coordinates are denoted as $q_1 = \varphi_1$ and $q_2 = \varphi_2$. The system is composed of two parts, namely the upper link and the lower link, and the position of the COM for each part, namely $P_1(X_1, Z_1)$ and $P_2(X_2, Z_2)$, can be calculated as follows:

$$X_1 = \frac{1}{2} l_1 \cos(q_1) \tag{3-48}$$

$$Z_1 = \frac{1}{2} l_1 \sin(q_1) \tag{3-49}$$

$$X_2 = l_1 \cos(q_1) + \frac{1}{2} l_2 \cos(q_1 + q_2) \tag{3-50}$$

$$Z_2 = l_1 \sin(q_1) + \frac{1}{2} l_1 \sin(q_1 + q_2) \tag{3-51}$$

According to the formulas (3-24) and (3-26), the linear velocity Jacobian matrix can be obtained as follows:

$$\boldsymbol{J}_{v,1} = \begin{bmatrix} \frac{\partial X_1}{\partial q_1} & \frac{\partial X_1}{\partial q_2} \\ \frac{\partial Z_1}{\partial q_1} & \frac{\partial Z_1}{\partial q_2} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} l_1 \sin(q_1) & 0 \\ \frac{1}{2} l_1 \cos(q_1) & 0 \end{bmatrix} \tag{3-52}$$

$$\boldsymbol{J}_{v,2} = \begin{bmatrix} \frac{\partial X_2}{\partial q_1} & \frac{\partial X_2}{\partial q_2} \\ \frac{\partial Z_2}{\partial q_1} & \frac{\partial Z_2}{\partial q_2} \end{bmatrix} = \begin{bmatrix} -l_1 \sin(q_1) - \frac{1}{2} l_1 \sin(q_1 + q_2) & -\frac{1}{2} l_1 \sin(q_1 + q_2) \\ l_1 \cos(q_1) + \frac{1}{2} l_2 \cos(q_1 + q_2) & \frac{1}{2} l_2 \cos(q_1 + q_2) \end{bmatrix} \tag{3-53}$$

The angular velocity Jacobian matrix has already been calculated via formular (3-28), and the gravitational potential energy of the system is the following:

$$\boldsymbol{U} = -m_1 g \left( \frac{1}{2} l_1 \sin(q_1) \right) - m_2 g \left( \frac{1}{2} l_2 \sin(q_1 + q_2) + l_1 \sin(q_1) \right) \tag{3-54}$$

Based on the equations (3-52), (3-53) and (3-28), the mass matrix of the system is as follows:

$$\begin{aligned} M(\boldsymbol{q}) &= m_1 \boldsymbol{J}_{v,1}{}^T \boldsymbol{J}_{v,1} + I_1 \boldsymbol{J}_{\omega,1}{}^T \boldsymbol{J}_{\omega,1} + m_2 \boldsymbol{J}_{v,2}{}^T \boldsymbol{J}_{v,2} + I_2 \boldsymbol{J}_{\omega,2}{}^T \boldsymbol{J}_{\omega,2} \\ &= m_1 \boldsymbol{J}_{v,1}{}^T \boldsymbol{J}_{v,1} + m_2 \boldsymbol{J}_{v,2}{}^T \boldsymbol{J}_{v,2} + \begin{bmatrix} I_1 + I_2 & I_2 \\ I_2 & I_2 \end{bmatrix} \end{aligned} \tag{3-55}$$

Subsequently, the $V(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $G(\boldsymbol{q})$ vectors can be computed by applying formulas (3-38) and (3-39), respectively. The entire Lagrangian dynamic equation (3-40) for the two-link leg

can then be obtained. It is noteworthy that the derivation of intricate formulas can be facilitated by utilizing the "syms" function offered by MATLAB.

Until now, all of the parameters in equation (3-40) have been based on generalized coordinates and its velocity and acceleration, which correspond to the state of the two-link joints, also known as "joint space". However, in quadruped robot control, particularly under central dynamics models, the commonly available information is the spatial position of the foot end in Cartesian space. Therefore, the dynamic equation expressed based on the foot end position in Cartesian space should also be derived.

Equation (3-44) can be rewritten as follows:

$$\boldsymbol{v_f} = \boldsymbol{J_{v,f}} \cdot \dot{\boldsymbol{q}} \qquad (3-56)$$

By continuing to differentiate both sides of formula (3-56) with respect to time, the relationship between foot end acceleration and generalized coordinate acceleration can be obtained:

$$\boldsymbol{a_f} = \boldsymbol{J_{v,f}} \cdot \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{\boldsymbol{v,f}} \cdot \dot{\boldsymbol{q}} \qquad (3-57)$$

Through formula (3-57), the generalized coordinate acceleration is calculated as follows:

$$\ddot{\boldsymbol{q}} = \boldsymbol{J_{v,f}}^{-1}\left(\boldsymbol{a_f} - \dot{\boldsymbol{J}}_{\boldsymbol{v,f}} \cdot \dot{\boldsymbol{q}}\right) \qquad (3-58)$$

By substituting formula (3-58) with dynamic formula (3-40), the dynamic equation in Cartesian space can be obtained:

$$M(\boldsymbol{q})\boldsymbol{J_{v,f}}^{-1}\left(\boldsymbol{a_f} - \dot{\boldsymbol{J}}_{\boldsymbol{v,f}} \cdot \dot{\boldsymbol{q}}\right) + V(\boldsymbol{q}, \dot{\boldsymbol{q}}) + G(\boldsymbol{q}) = \boldsymbol{\tau} \qquad (3-59)$$

If the force on the foot must be solved, the dynamic equation can be further manipulated by substituting the formula (3-47), producing the following expression:

$$\boldsymbol{J_{v,f}}^{-1} M(\boldsymbol{q}) \boldsymbol{J_{v,f}}^{-1}\left(\boldsymbol{a_f} - \dot{\boldsymbol{J}}_{\boldsymbol{v,f}} \cdot \dot{\boldsymbol{q}}\right) + \boldsymbol{J_{v,f}}^{-1} V(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{J_{v,f}}^{-1} G(\boldsymbol{q}) = \boldsymbol{F} \qquad (3-60)$$

## 3.2 Two-DOF Four-Link Leg

This section presents the leg structure employed on the Nermo solid robot. As depicted in Figure 3.3, it features a more intricate four-link structure compared to the two-link leg, and the motor that imparts torque to it is also positioned on the robot's torso. However, in the case of the four-link leg, the two motors are situated in different locations, namely joint A and B in Figure 3.4.

In Figure 3.3 and Figure 3.4, the origin of the leg coordinate system is established at joint A. Joint B sits directly above Joint A, and both joints are attached to the body. These two joints are immovable and are driven directly by motors on the robot's body. Furthermore, joints D, E, and F are connected via the same links.

In addition, $q_1$ and $q_2$ denote the angles between the link AE and the $\boldsymbol{x}$ -axis and between the link BC and the $\boldsymbol{z}$ -axis, respectively. Notably, the movements of these two angles are independent of each other. Or in other words, these two angles can be used as the generalized coordinates of the system.

In this chapter, the geometric method is used to solve the forward and inverse kinematics of the four-link leg.

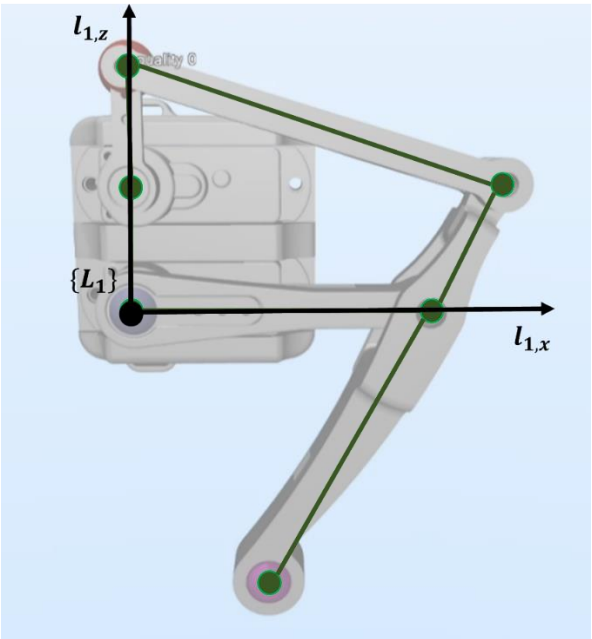


*Figure 3.3:MuJoCo simulation model of the four-link leg structure currently used by Nermo. In contrast to the two-link leg, the four-link leg has two joints connected to the body, namely the joint at the origin position and the first joint directly above the origin position. These two joints are connected to the motor and receive torque.*

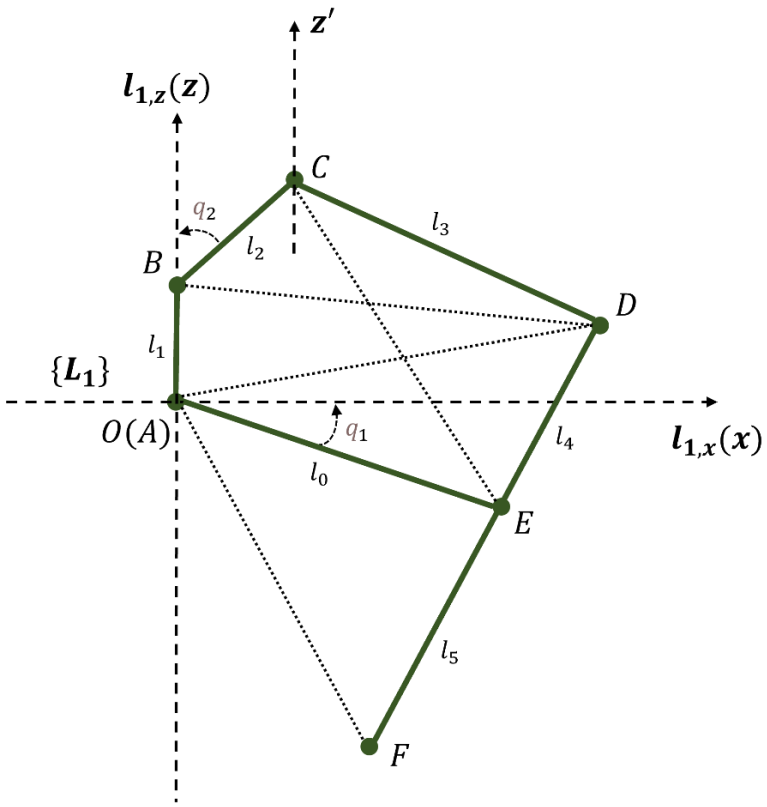


*Figure 3.4: An abstract diagram of Figure 3.3. Several auxiliary lines are added to depict the*

*interrelationship among various components and the corresponding mathematical expressions more accurately. In the diagram, the symbol $\boldsymbol{z}$ denotes $\boldsymbol{l_{1,z}}$, $\boldsymbol{x}$ represents $\boldsymbol{l_{1,x}}$, and $\boldsymbol{z}'$ refers to a non-collinear straight line that is parallel to $\boldsymbol{z}$ and pointing in the same direction.*

### 3.2.1 Law of Cosine

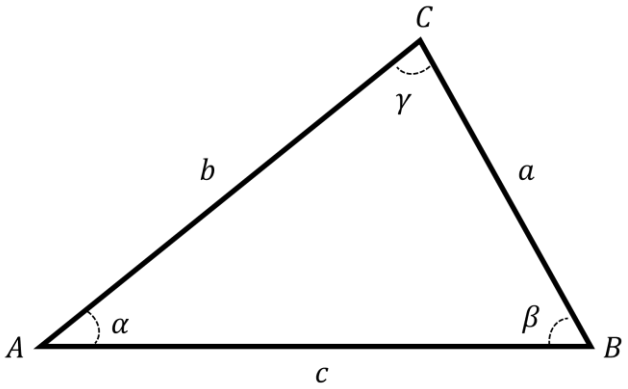


*Figure 3.5: An ordinary triangle has sides a, b and c, vertices A, B and C, and interior angles α, β and γ.*

For the triangle in Figure 3.5, the theorem called the "Law of Cosine" to expresses the relationship between its sides and angles, the expression is as follows:

$$c = f_L(a, b, \gamma) = \sqrt{a^2 + b^2 - 2ab\cos\gamma} \qquad (3-61)$$

$$\gamma = f_A(a, b, c) = arccos\frac{a^2 + b^2 - c^2}{2ab} \qquad (3-62)$$

Among them, the subscript $L$ is the abbreviation of "Line", and the subscript $A$ is the abbreviation of the "Angle".

### 3.2.2 Forward Kinematics of the Two-DOF Four-Link Leg

The purpose of forward kinematics is to determine the position of the foot $F(x_F, z_F)$ given the known generalized coordinates $q_1, q_2$.

The position of joint C can be obtained from the position of joint B and the link BC that connects joint B and joint C:

$$C(x_c, z_c) = (l_2\sin(-q_2), l_1 + l_2\cos(q_2)) \qquad (3-63)$$

Similarly, through the position of joint A and the link AE that connects joint A and joint E, the position of joint E can be calculated as follows:

$$E(x_E, z_E) = (l_0\cos(q_1), l_0\sin(q_1)) \qquad (3-64)$$

Therefore, the length of link CE can be obtained from the positions of joints C and E:

$$|CE| = \sqrt{(x_E - x_C)^2 + (z_E - z_C)^2} \qquad (3-65)$$

The angle between CE and $\boldsymbol{z}$ axis can also be obtained as follows:

$$\angle EC\boldsymbol{z}' = \pi - \frac{x_E - x_C}{|x_E - x_C|}\arccos\frac{z_C - z_E}{|CE|} \quad (3-66)$$

For the triangle $\Delta CDE$, the lengths of the three sides are currently known, and the size of $\angle ECD$ can be obtained according to formula (3-61):

$$\angle ECD = arccos\frac{|CE|^2 + {l_3}^2 - {l_4}^2}{2 \cdot l_3 \cdot |CE|} \quad (3-67)$$

Based on formulas (3-66) and (3-67), $\angle DC\boldsymbol{z}'$can also be obtained:

$$\angle DC\boldsymbol{z}' = \angle EC\boldsymbol{z}' - \angle ECD \quad (3-68)$$

The position of joint D can then also be calculated:

$$D(x_D, z_D) = (x_C + l_3\sin(\pi - \angle DC\boldsymbol{z}'), z_C - l_3\cos(\pi - \angle DC\boldsymbol{z}')) \quad (3-69)$$

Since joints D, E, and F are collinear, the position of joint F, namely the foot end position can be obtained:

$$F(x_F, z_F) = (x_E - \frac{l_5}{l_4}(x_D - x_E), z_E - \frac{l_5}{l_4}(z_D - z_E)) \quad (3-70)$$

In summary, the forward kinematics $F(q_1, q_2)$ is a function of the generalized coordinates $q_1$, $q_2$.

### 3.2.3 Inverse Kinematics of the Two-DOF Four-Link Leg

The purpose of inverse kinematics is to determine the generalized coordinates $q_1, q_2$ given the known position of the foot $F(x_F, z_F)$.

Given the positions of joints A and F, the length of the link AF can be obtained:

$$|AF| = \sqrt{{x_F}^2 + {z_F}^2} \quad (3-71)$$

The angle $\angle FA\boldsymbol{x}$ can then be calculated as follows:

$$\angle FA\boldsymbol{x} = \frac{z_F}{|z_F|}\arccos\frac{x_F}{|AF|} \quad (3-72)$$

Meanwhile, based on the law of cosine, the angle $\angle FAE$ can be calculated:

$$\angle FAE = \arccos\frac{|AF|^2 + {l_0}^2 - {l_5}^2}{2 \cdot l_0 \cdot |AF|} \quad (3-73)$$

By utilizing formulas (3-72) and (3-73), the magnitude of the generalized coordinate $q_1$ can be determined:

$$q_1 = \angle EA\boldsymbol{x} = \angle FA\boldsymbol{x} - \angle FAE \quad (3-74)$$

Subsequently, another generalized coordinate $q_2$ should be solved.

The expression of joint E has been obtained in formula (3-64) and does not change in inverse kinematics. Since joints D, E, and F are collinear, the position of joint D is as follows:

$$D(x_D, z_D) = (x_E + \frac{l_4}{l_5}(x_E - x_F), z_E + \frac{l_4}{l_5}(z_E - z_F)) \quad (3-75)$$

The lengths of links AD and BD can then be obtained:

$$|AD| = \sqrt{{x_D}^2 + {z_D}^2} \quad (3-76)$$

$$|BD| = \sqrt{{x_D}^2 + (z_D - z_B)^2} \quad (3-77)$$

For triangles $\Delta ABD$ and $\Delta BCD$, the law of cosines can be utilized to determine the values of $\angle ABD$ and $\angle DBC$, given that their side lengths have been determined:

$$\angle ABD = arccos\frac{|BD|^2 + {l_1}^2 - |AD|^2}{2 \cdot l_1 \cdot |BD|} \quad (3-78)$$

$$\angle DBC = arccos\frac{|BD|^2 + {l_2}^2 - {l_3}^2}{2 \cdot l_2 \cdot |BD|} \quad (3-79)$$

Thus, the second generalized coordinate $q_2$ can be calculated as follows:

$$q_2 = \angle ABC - \pi = \angle ABD + \angle DBC - \pi \quad (3-80)$$

Thus far, the computations for both forward and inverse kinematics of the four-link leg have been completed. For the calculation of the foot Jacobian matrix and the dynamics of the four-link leg, Chapters 3.1.4 and 3.1.5 can be used as a reference.

## 3.3 Spine of Nermo

This section elucidates the kinematics of Nermo's flexible spine, which exhibits two DOFs, namely, horizontal and vertical extension and flexion. These movements are controlled by two servos placed on the shoulder and hip of the robot, respectively. The servo is connected by a wire to pull the spine and to bend it in a certain direction.

### 3.3.1 Kinematics of the Spine with Horizontal DOF

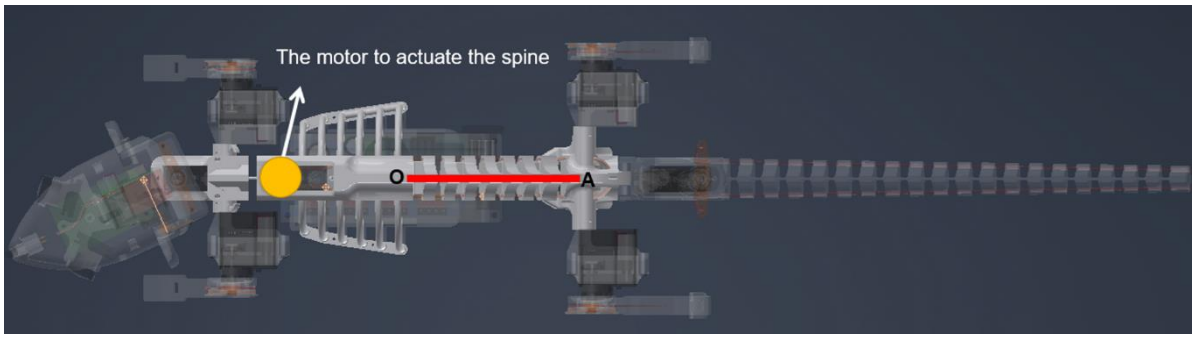


*Figure 3.6: A top view of the real quadruped robot Nermo. Neromo's spine is designed to be flexible, and its movement can be controlled by wire connections.*

The calculation of the spinal kinematics is also performed via geometric methods. Here I need

to thank my colleague Yuhong Huang for providing pictures and formula derivation ideas in the sections about the four-link leg kinematics and in this section about the spine's horizontal motion.

The principal objective of spine kinematics is to establish the correlation between the deflection angle $q_s$ of the servo that controls the spine and the rotation angle $\theta_s$ of the spine. This correlation will facilitate the determination of the distance between the front shoulder joints and rear hip joints, as well as the position of the COM and the moment of inertia, which are critical parameters for research following this paper concerning the effect of the spine of the robot motion.

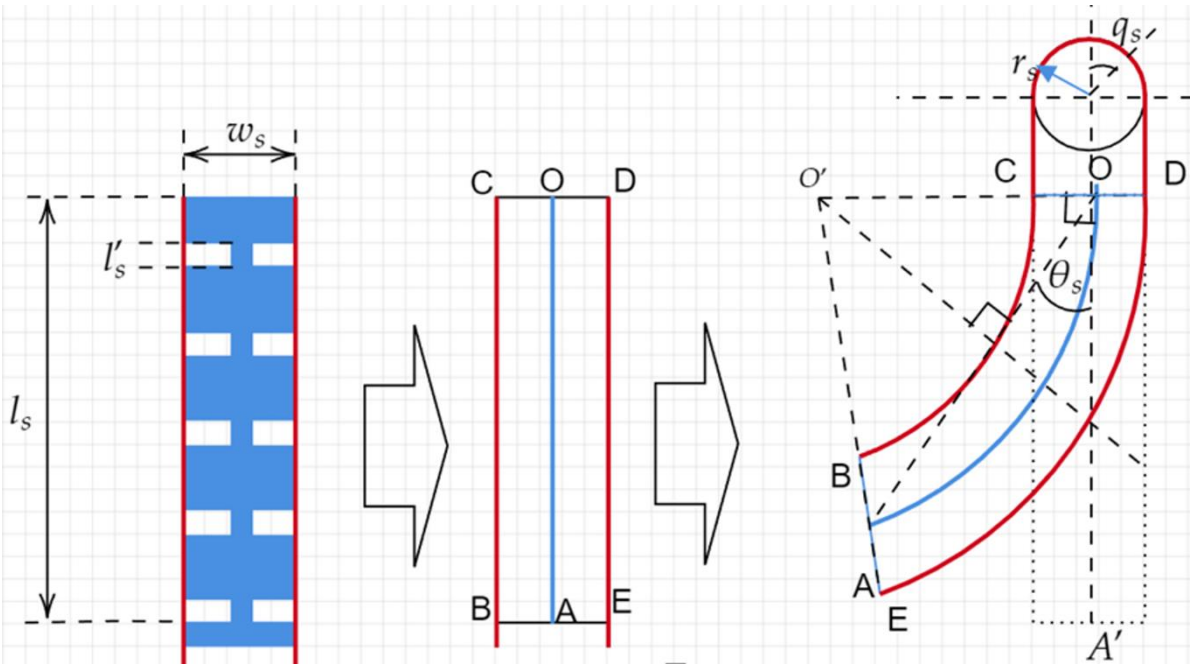


*Figure 3.7: Illustration of the spine's motion. From left to right, the first image depicts the actual structure of the spine, while the second image simplifies it as a blue line segment OA. The tendons, represented by two red lines BC and DE, are manipulated by the same servo to achieve controlled movement. The final image illustrates the configuration of the spine when it is horizontally flexed.*

Figure 3.7 illustrates the process of abstraction and simplification of the spine, in which the two lines BE and CD denote the spine's width. The spine has a constant width of $\omega_s$ throughout the entire spine. Link OA is always in the middle of links BC and DE, i.e.,$|OC| = |OD| = |AB| = |AE| = \frac{\omega_s}{2}$. In the unflexed state, the length of the spine is expressed as $l_s$. The steering wheel radius is expressed as $r_s$.

In the rightmost portion of Figure 3.7, the arcs $\widehat{OA}$, $\widehat{BC}$, and $\widehat{DE}$ are isometric curves, where the line segments BE and CD intersect at point $O'$. It is assumed in this study that the tension force applied by the tendon is uniformly distributed and causes a consistent deformation of the spine, such that arcs $\widehat{OA}$, $\widehat{BC}$, and $\widehat{DE}$ are segments of the circles with point $O'$ as their common center.

Finally, a crucial assumption must be made, specifically, that the tendon is a rigid structure and that its overall length remains unaffected by the magnitude of the applied force.

Based on the above assumptions, the relationship between the length of the arc $\widehat{BC}$ and the

length of the arc $\widehat{OA}$ in the rightmost sub-graph in Figure 3.7 can be obtained as follows:

$$\left|\widehat{BC}\right| = l_s - r_s q_s = \left|\widehat{OA}\right| - q_s r_s \qquad (3-81)$$

Equation (3-81) is based on changes in tendon length.

Denote the size of the instantaneous angle $\angle AO'O$ as $\theta_t$, and denote the instantaneous radius $O'B$ as $r_t$, based on the geometric relationship, another equation can then be obtained based on the equation (3-81):

$$q_s r_s = \left|\widehat{OA}\right| - \left|\widehat{BC}\right| = \theta_t\left(r_t + \frac{\omega_s}{2}\right) - \theta_t r_t = \frac{\theta_t \omega_s}{2} \qquad (3-82)$$

By transposing the position of the parameters in formula (3-83), the relational expression can be obtained:

$$\theta_t = \frac{2q_s r_s}{\omega_s} \qquad (3-83)$$

Since line segments $O'O$ and $O'A$ are both radii of a circle, their lengths are equal, and the following applies:

$$\angle O'OA = \angle O'AO = \frac{2\pi - \theta_t}{2} = \pi - \frac{\theta_t}{2} \qquad (3-84)$$

The rotation angle $\theta_s$ of the spine in Figure 3.7 is as follows:

$$\theta_s = \pi - \angle O'OA \qquad (3-85)$$

Thus based on the equations (3-83), (3-84) and (3-85), the relationship between the deflection angle $q_s$ of the servo and the rotation angle $\theta_s$ of the spine can be determined as follows:

$$\theta_s = \frac{q_s r_s}{\omega_s} \qquad (3-86)$$

In addition, due to the limitations of the physical structure of the spine, there is a gap $l_s{}'$ between every two adjacent spines described in the leftmost sub-figure within Figure 3.7 and a total of N gaps on one spine side, and the length change of the unilateral spine must be less than or equal to the total gap of the spine:

$$q_s r_s \leq \mathrm{N} \cdot l_s{}' \qquad (3-87)$$

Thus, the following can easily be calculated:

$$-\frac{\mathrm{N} \cdot l_s{}'}{r_s} \leq q_s \leq \frac{\mathrm{N} \cdot l_s{}'}{r_s} \qquad (3-88)$$

### 3.3.2 Kinematics of the Spine with Vertical DOF

As the spinal section has a similar structure both vertically and horizontally, the approach to solving the kinematic equations for the vertical DOF of the spine is fundamentally the same as the approach used to solve the kinematic equations for the horizontal DOF discussed in section 3.3.1. Hence, the relationship between the deflection angle $q_s$ of the servo and the

rotation angle $\theta_s$ of the spine is the same as in equation (3-86).

A specific aspect must be considered in the cross-sectional structure of the spine, as depicted in the top sub-figure within Figure 3.9. The upper and lower sides of the spine have distinct gap lengths, which are respectively denoted as $l_u$ and $l_d$.

Therefore, the optional value range of $q_s$ is different from the previous section with equation (3-88), and its neu expression is as follows

$$-\frac{\mathrm{N} \cdot l_u}{r_s} \leq q_s \leq \frac{\mathrm{N} \cdot l_d}{r_s} \qquad (3-89)$$

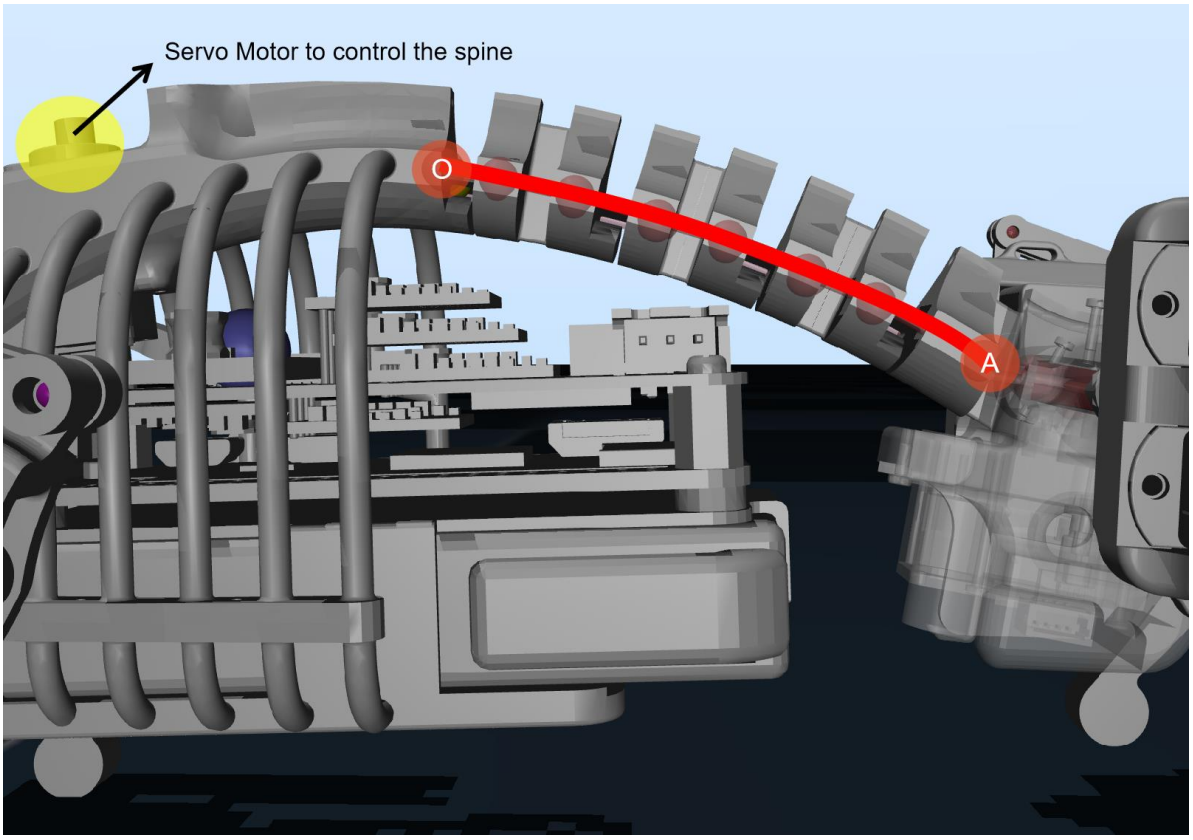


*Figure 3.8: Lateral view of the Nermo robot in MuJoCo: The rat robot exists a soft actuated spine and controls the vertical DOF of it using two lines driven by a motor.*

When applying the kinematic equation of the spine to actual simulations, it is important to consider additional factors. Firstly, the movement of the spine in the horizontal DOF is different from that of the vertical DOF, where gravity affects the movement when the spine moves vertically, resulting in a different bending effect. Secondly, the tendon cannot achieve complete rigidity, leading to deviation between the actual and theoretical results of the kinematic equation. Especially when the motion of the spine is near the boundary of the $q_s$ value range (equation (3-88) and (3-89)), due to greater force, the error will be larger. Lastly, the output torque of the servo controlling the tendon movement must be considered. For horizontal movement, the servo must overcome the elastic potential energy generated by tendon stretching and spine deformation. For vertical movement, the servo must additionally overcome or gain gravitational potential energy.

However, the aforementioned issue does not affect the linear relationship between the deflection angle $q_s$ of the servo and the rotation angle $\theta_s$ of the spine. Therefore, a feasible solution is to determine the original equation (3-86) and to scale it using a suitable coefficient $a$. For the horizontal movement of the spine, a single coefficient $a_h$ can be applied to both left and right movements. Conversely, for the vertical movement of the spine, distinct coefficients are required for upward and downward movements, namely $a_{v,up}$ and $a_{v,down}$.

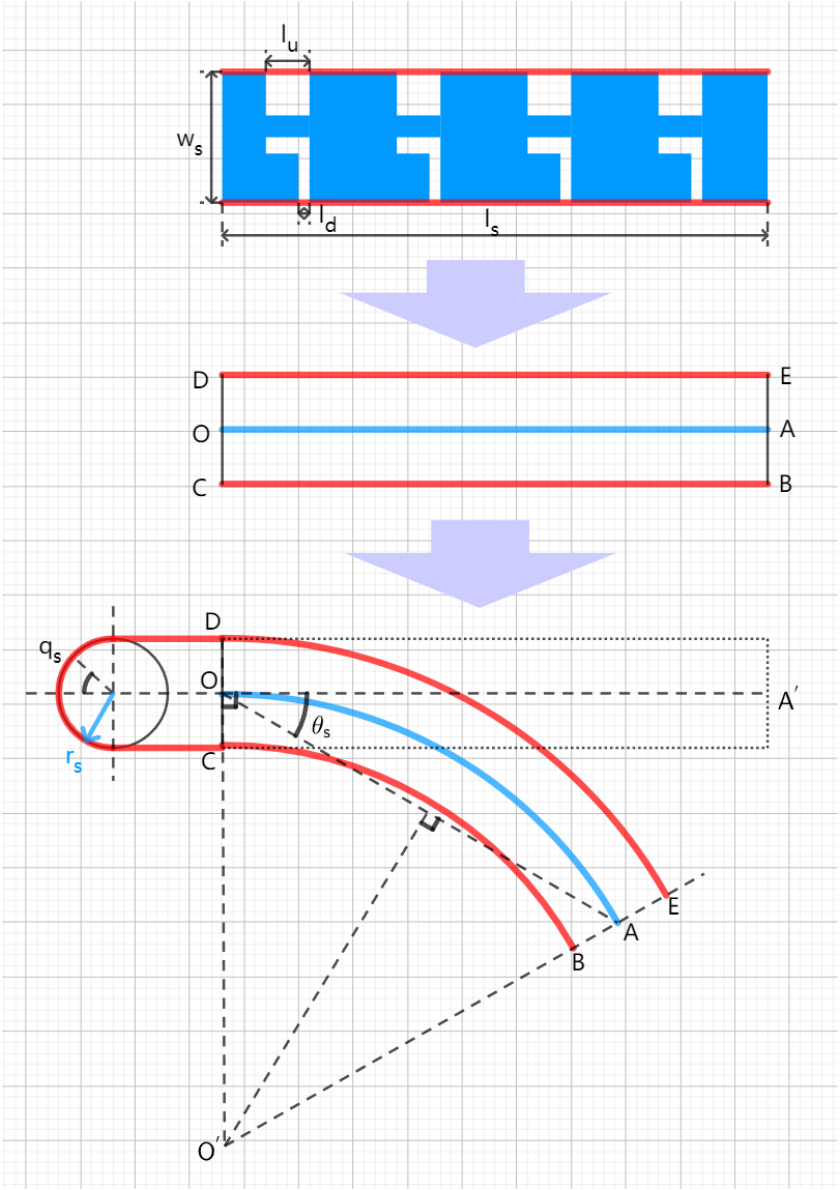


*Figure 3.9: Schematic of the spinal flexion in a vertical direction. From top to bottom, the first model is the structure of a real robot spine. The second abstracts the spine as a single wire (the blue one) that is driven by two tendons in a red line. The third is the schematic diagram of a vertical flexing spine. In addition, the two red lines are tendons that drive the robot spine and are controlled by the same servo. To control the spine, the problem is deriving the control parameters of the spine motor $q_s$ from the target turn angle of the spine $\theta_s$. It is assumed that, when the motor rotates clockwise, $q_s$ is positive and corresponds $\theta_s$ is positive in clockwise.*

After the above changes, the formula (3-86) will become the following:

$$\theta_s = \frac{q_s r_s}{\omega_s} * a_i \; with \; i \in \{"h", "v, up", "v, down"\} \qquad (3-90)$$

## 3.4 Three-DOF Two-Link Leg

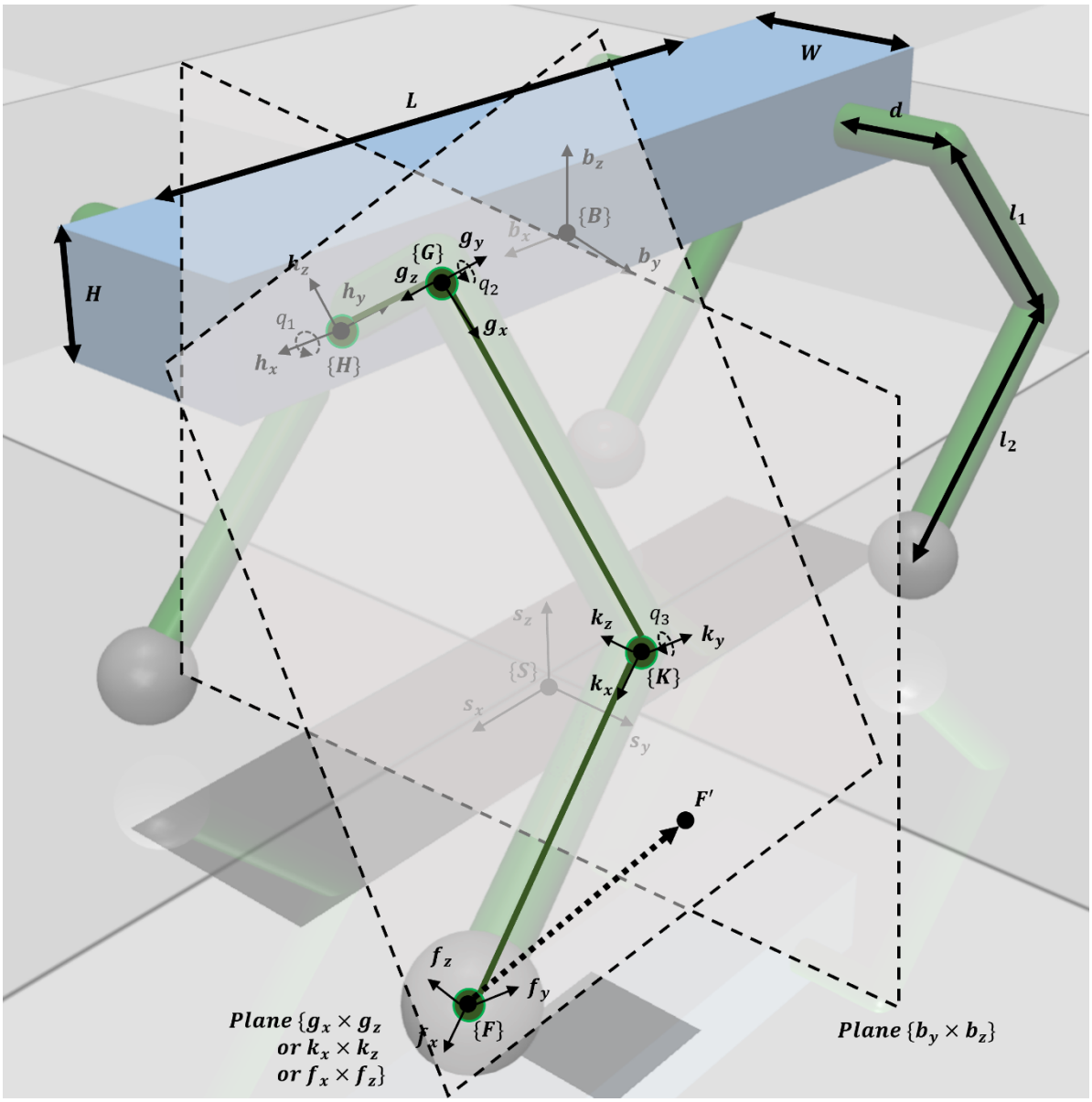


*Figure 3.10: Schematic diagram of the quadruped robot with cross-sectional views of the LF leg with three DOFs. These three DOFs are positioned at the origin of the {**H**}, {**G**}, and {**K**} coordinate systems, with joint angles $q_1$, $q_2$, and $q_3$, respectively. To facilitate the derivation of the forward and inverse kinematics of the leg via matrix methods, a coordinate system is introduced at each joint and also at the foot {**F**}.The body of the robot is SRB, and its length between the LF and RF hip joint, the total width and the total height are **L**, **W** and **H** respectively, and the lengths of the three links of the legs are $d$, $l_1$ and $l_2$ respectively. The parameters of the four legs are designed to be the same.*

Based on the analysis of the two-DOF two-link leg in section 3.1, this paper also presents forward and inverse kinematics analyze regarding the three-DOF two-link leg to realize the

motion control of a quadruped robot with totally 12 leg DOFs.

In contrast to the 2-DOF leg design, the 3-DOF leg design has DOFs in two directions. Specifically, the first joint angle $q_1$ is oriented towards the x-axis of the body coordinate system $\boldsymbol{b_x}$, while the second and third joint angles $q_2$ and $q_3$, are oriented towards the same direction that is orthogonal to $q_1$. The included angle between $q_2$ and $q_3$ with the y-axis direction of the body coordinate system $\boldsymbol{b_y}$ is affected by the value of $q_1$. Therefore, to facilitate a more rigorous exposition of the forward and inverse kinematics derivation process, it is beneficial to introduce two auxiliary planes. Specifically, Plane $\{\boldsymbol{b_y} \times \boldsymbol{b_z}\}$ is positioned orthogonally to the direction of $q_1$, while Plane $\{\boldsymbol{g_x} \times \boldsymbol{g_z}\ \boldsymbol{or}\ \boldsymbol{k_x} \times \boldsymbol{k_z}\ \boldsymbol{or}\ \boldsymbol{f_x} \times \boldsymbol{f_z}\}$ is oriented orthogonally to the directions of $q_2$ and $q_3$.

### 3.4.1 Forward Kinematics of the Three-DOF Two-Link Leg

This section performs the forward kinematics derivation via the method of the transformation matrix with the following general formula:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{p} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{3-91}$$

where $\boldsymbol{R}$ represents the rotation matrix from the first coordinate system to the second coordinate system, $\boldsymbol{p}$ represents the vector from the origin of the first coordinate system to the origin of the second coordinate system, and $\boldsymbol{0}$ is a zero vector with a size of (1, 3).

Therefore, in reference to the coordinate system annotation in Figure 3.10, the transformation matrices from the initial position, namely the position of the hip joint $\{\boldsymbol{H}\}$, along the connecting links to the foot end frame $\{\boldsymbol{F}\}$ can be expressed as follows:

$$\boldsymbol{T_{H,H}} = \begin{bmatrix} \boldsymbol{R_x}(q_1) & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{3-92}$$

$$\boldsymbol{T_{H,G}} = \begin{bmatrix} \boldsymbol{R_y}(q_2) & [0\ sign_d \cdot d\ 0]^T \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{3-93}$$

$$\boldsymbol{T_{G,K}} = \begin{bmatrix} \boldsymbol{R_y}(q_3) & [l_1\ 0\ 0]^T \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{3-94}$$

$$\boldsymbol{T_{K,F}} = \begin{bmatrix} \boldsymbol{I} & [l_2\ 0\ 0]^T \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{3-95}$$

Among them, $\boldsymbol{R_x}(\cdot)$ and $\boldsymbol{R_y}(\cdot)$ are the rotation matrices around the x and y axes; they refer to formulas (2-6) and (2-7). $d$, $l_1$ and $l_2$ are the lengths of the three connecting links of the leg respectively, and $\boldsymbol{I}$ is the identity matrix. $sign_d$ is a binary number whose value is -1 or 1, where 1 refers to the two legs LF and LR on the left side, and -1 refers to the two legs RF and RR on the right side. Thus, the transformation matrix from the hip joint to the foot is as follows:

$$\boldsymbol{T_{H,F}}(q_1, q_2, q_3) = \boldsymbol{T_{H,H}} \cdot \boldsymbol{T_{H,G}} \cdot \boldsymbol{T_{G,K}} \cdot \boldsymbol{T_{K,F}} = \begin{bmatrix} \boldsymbol{R_{H,F}} & \boldsymbol{p_{H,F}} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{3-96}$$

Among these, $\boldsymbol{p_{H,F}}(q_1, q_2, q_3)$ is the three-dimensional position vector from the hip joint to the foot, assuming that $\boldsymbol{p_{H,F}} = (X_f, Y_f, Z_f)$.

### 3.4.2 Inverse Kinematics of the Three-DOF Two-Link Leg

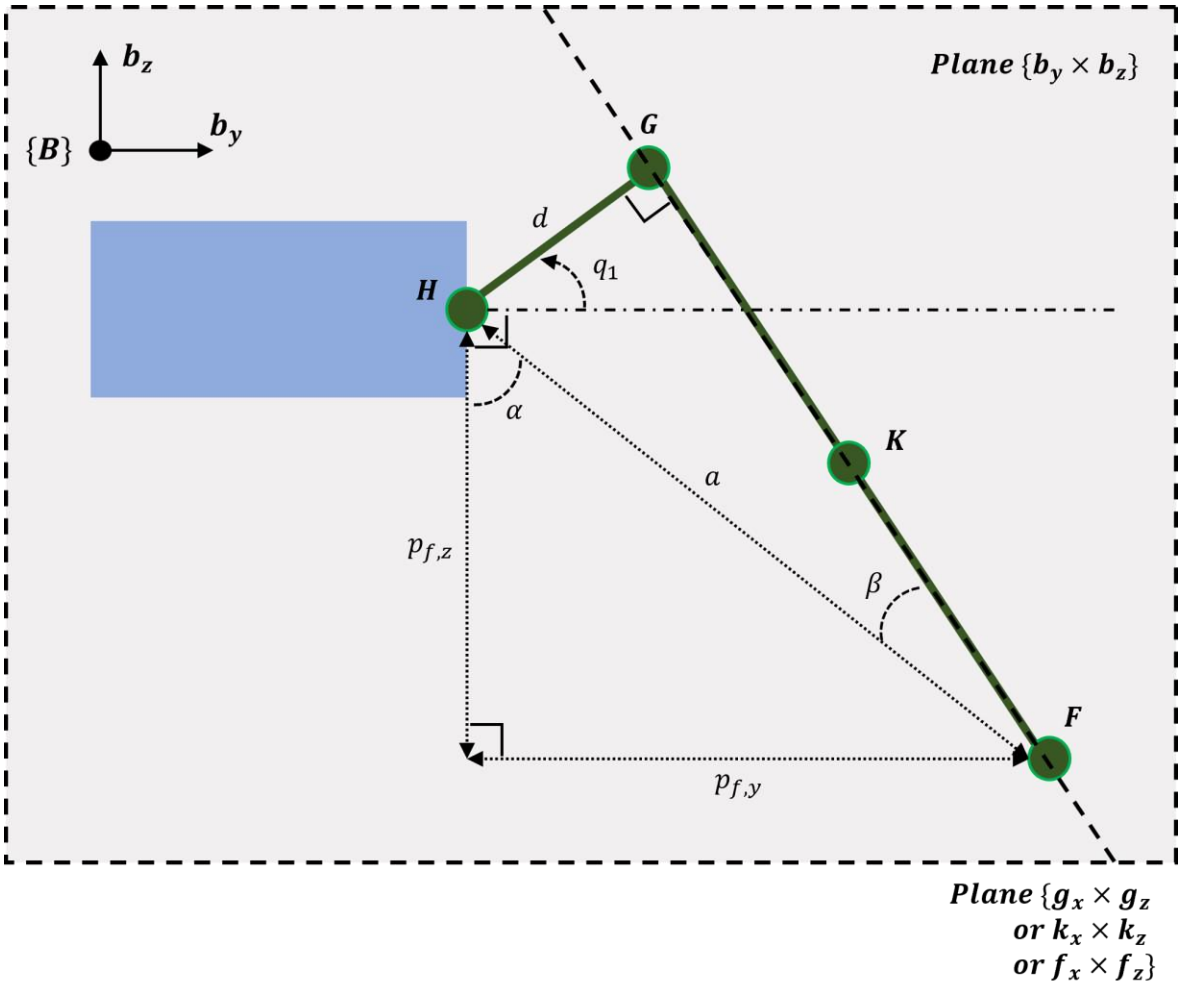


*Figure 3.11: Front view of plane* $\{\boldsymbol{b_y} \times \boldsymbol{b_z}\}$*. The blue square is the front view of SRB. The spatial foot position relative to the hip joint position is expressed as* $p_f = (p_{f,x}, p_{f,y}, p_{f,z})$.

The solution of the inverse dynamics of the three-DOF two-link leg is performed via geometric methods. As illustrated in Figure 3.11, the length of line segment $a$ can be expressed as follows:

$$a = \sqrt{{p_{f,y}}^2 + {p_{f,z}}^2} \qquad (3-97)$$

Then the angle $\alpha$ is then calculated as follows:

$$\angle\alpha = \arcsin\left(\frac{p_{f,y}}{a}\right) \qquad (3-98)$$

The angle $\beta$ is the following:

$$\angle\beta = \arcsin\left(\frac{d}{a}\right) \qquad (3-99)$$

The angle of $q_1$ can be obtained by combining the two formulas above:

$$q_1 = \angle\alpha + \left(\frac{\pi}{2} - sign_d \cdot \angle\beta\right) - \frac{\pi}{2} = \angle\alpha - sign_d \cdot \angle\beta \qquad (3-100)$$

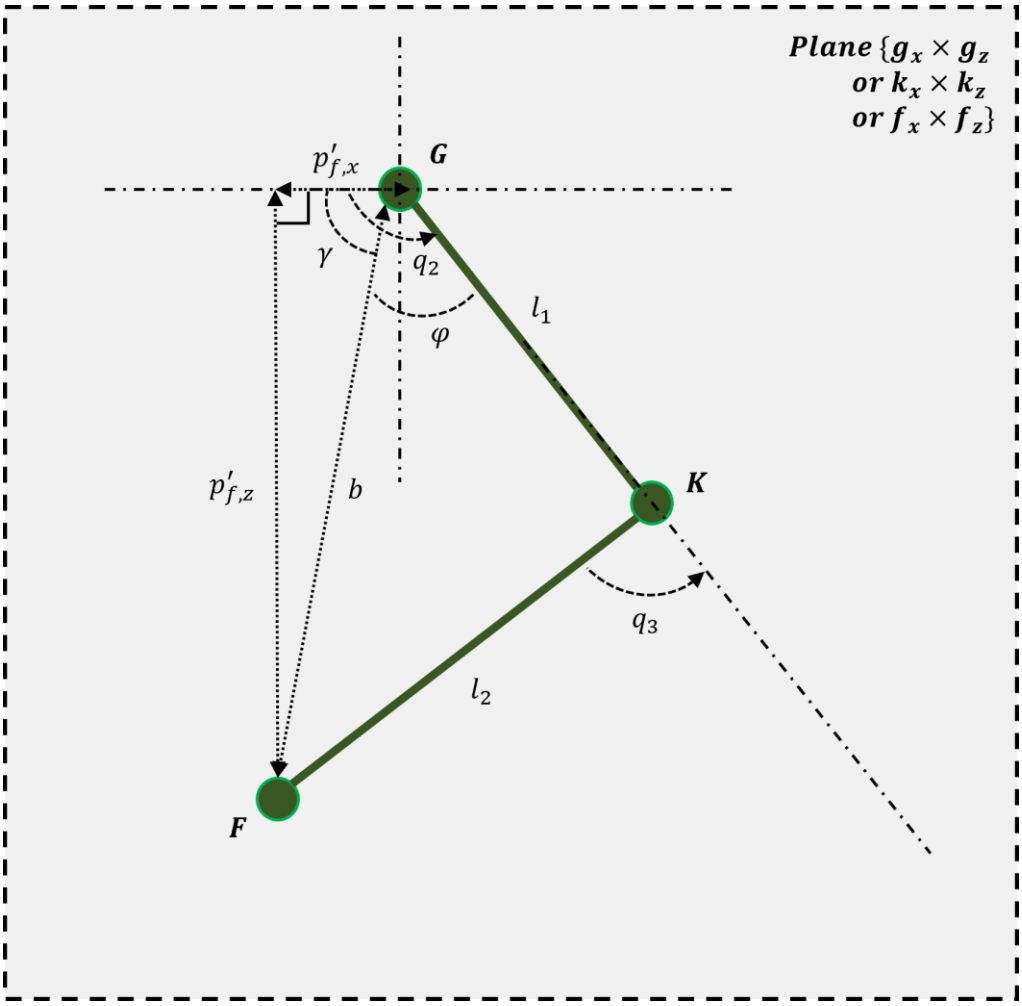


*Figure 3.12: Front view of plane* $\{\boldsymbol{g_x \times g_z}$ ***or*** $\boldsymbol{k_x \times k_z}$ ***or*** $\boldsymbol{f_x \times f_z}\}$. *Since this plane is not parallel to plane* $\{\boldsymbol{b_x \times b_z}\}$ *all the time, the foot position relative to joint* ***G*** *in this plane is the converted position of* $p_f$, *expressed as* $p'_f$.

The position vector pointing from joint ***H*** to joint ***G*** can be expressed through angle $q_1$ as follows:

$$\overrightarrow{\boldsymbol{HG}} = [0 \quad d \cdot \cos(q_1) \quad d \cdot \sin(q_1)]^T \qquad (3-101)$$

The position vector pointing from joint ***G*** to joint ***F*** can be expressed as follows:

$$\overrightarrow{\boldsymbol{GF}} = \overrightarrow{\boldsymbol{HF}} - \overrightarrow{\boldsymbol{HG}} = [p_{f,x} \quad p_{f,y} - d \cdot \cos(q_1) \quad p_{f,z} - d \cdot \sin(q_1)]^T \qquad (3-102)$$

Subsequently, let $\overrightarrow{\boldsymbol{GF}}$ rotate the angle $-q_1$ around the joint H to obtain the position vector $\overrightarrow{\boldsymbol{GF}}$ described in the plane depicted in Figure 3.12, namely $p'_f$:

$$p'_f = \boldsymbol{R_x}^{\boldsymbol{T}}(q_1) \cdot \overrightarrow{\boldsymbol{GF}} = (p'_{f,x}, p'_{f,y}, p'_{f,z}) \qquad (3-103)$$

Among them, $p'_{f,y}$ should be 0.

The length of line segment $b$ can then be expressed as follows:

$$b = \sqrt{{p'_{f,x}}^2 + {p'_{f,z}}^2} \qquad (3-104)$$

The angle $\gamma$ can also be calculated via the following equations:

$$\begin{cases} \angle\gamma = \arccos\left(\dfrac{p'_{f,x}}{b}\right) \ when\ p'_{f,z} \le 0 \\ \angle\gamma = -\arcsin\left(\dfrac{p'_{f,z}}{b}\right) \ when\ p'_{f,z} > 0, p'_{f,x} \ge 0 \\ \angle\gamma = \pi + \arcsin\left(\dfrac{p'_{f,z}}{b}\right) \ when\ p'_{f,z} > 0, p'_{f,x} < 0 \end{cases} \quad (3-105)$$

Regarding the Law of Cosine, as explained in Section 3.2.1, the angle $\varphi$ can be obtained through the following:

$$\cos(\angle\varphi) = \frac{l_1^2 + b^2 - l_2^2}{2 \cdot l_1 \cdot b} \quad (3-106)$$

and

$$\angle\varphi = \arccos(\cos(\angle\varphi)) \quad (3-107)$$

The angle $q_2$ can be obtained via the following:

$$q_2 = \angle\gamma + \angle\varphi \quad (3-108)$$

Similarly, the angle $q_2$ can be obtained as follows:

$$q_3 = -\left(\pi - \arccos\left(\cos\left(\frac{l_1^2 + l_2^2 - b^2}{2 \cdot l_1 \cdot l_2}\right)\right)\right) \quad (3-109)$$

### 3.4.3 Foot Jacobian Matrix of the Three-DOF Two-Link Leg

Using the same method as in section 3.1.4, the foot Jacobian matrix of the three-DOF two-link leg can be solved based on the forward kinematics presented in section 3.4.1.

$$\boldsymbol{J_{p,f}} = \boldsymbol{J_{v,f}} = \begin{bmatrix} \dfrac{\partial X_f}{\partial q_1} & \dfrac{\partial X_f}{\partial q_2} & \dfrac{\partial X_f}{\partial q_3} \\ \dfrac{\partial Y_f}{\partial q_1} & \dfrac{\partial Y_f}{\partial q_2} & \dfrac{\partial Y_f}{\partial q_3} \\ \dfrac{\partial Z_f}{\partial q_1} & \dfrac{\partial Z_f}{\partial q_2} & \dfrac{\partial Z_f}{\partial q_3} \end{bmatrix} \quad (3-110)$$

The full formula is transcribed in Appendix 2.

# 4 Control Loop of Gaits

In Chapter 2, the methodology for determining the optimal output of a system $\boldsymbol{u_{optimal}}$, under the known desired and current system states, namely $(\boldsymbol{x_d}, \boldsymbol{u_d})$ and $(\boldsymbol{x_{op}}, \boldsymbol{u_{op}})$, was presented through the application of MPC and QP solvers. However, the process of defining the desired state has not been introduced, and the optimal system output obtained from MPC represents the GRF required at the foot end, rather than the corresponding servo motor torque value that the robot can accept. Additionally, the movement of the robot's legs while in the air requires control algorithms. This chapter addresses these issues and establishes a complete multi-gait control loop for quadruped robots including MPC and QP solvers.

## 4.1 Reference Trajectory Generation

This section describes how to generate the desired state and input vectors $\boldsymbol{x_d}, \boldsymbol{u_d}$, see equation (2-24) and Figure 2.6, to generate different motion gaits based on the same MPC architecture.

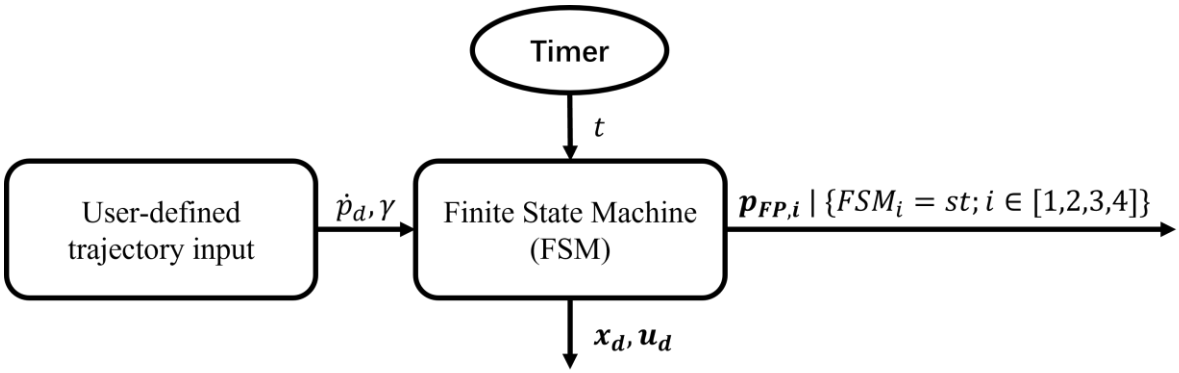


*Figure 4.1: The high-level control parameters necessary for the quadruped robot, namely the robot COM velocity $\dot{p}_d$ and the yaw angle $\gamma$, are obtained through manual input by the use. The Finite State Machine (FSM) generates the desired state and input vector, namely $\boldsymbol{x_d}$ and $\boldsymbol{u_d}$, for the robot using $\dot{p}_d$ and $\gamma$, as well as the current time $t$ and intended gait. Additionally, it utilizes another algorithm to generate the desired foot-end trajectory for the legs in the air phase.*

The task of reference trajectory generation is mainly performed via Finite State Machine (FSM). There are two main trajectories that must be generated, namely the desired state and input vector of the robot torso, specifically $\boldsymbol{x_d}$ and $\boldsymbol{u_d}$, and the foot end trajectory of the legs in the stance (air) phase. The inner operating process of the FSM is illustrated in Figure 4.2.

### 4.1.1 Finite State Machine

The desired state vector of the robot $\boldsymbol{x_d}$ can be directly obtained via the given robot COM speed $\dot{p}_d$ and yaw angle $\gamma$:

$$\begin{cases} \dot{\boldsymbol{p}}_{\boldsymbol{d}} = \ddot{\boldsymbol{p}}_{\boldsymbol{d}} \cdot t;\ \boldsymbol{p}_{\boldsymbol{d}} = \frac{1}{2} \cdot \ddot{\boldsymbol{p}}_{\boldsymbol{d}} \cdot t^2; \quad with\ t < \min\left(\frac{\dot{\boldsymbol{p}}_{\boldsymbol{d}}}{\ddot{\boldsymbol{p}}_{\boldsymbol{d}}}\right) \\ \dot{\boldsymbol{p}}_{\boldsymbol{d}} = \dot{\boldsymbol{p}}_{\boldsymbol{d}};\ \boldsymbol{p}_{\boldsymbol{d}} = \dot{\boldsymbol{p}}_{\boldsymbol{d}} \cdot t - \frac{1}{2} \cdot \frac{\dot{\boldsymbol{p}}_{\boldsymbol{d}}^{\boldsymbol{2}}}{\ddot{\boldsymbol{p}}_{\boldsymbol{d}}}; \quad with\ t > \min\left(\frac{\dot{\boldsymbol{p}}_{\boldsymbol{d}}}{\ddot{\boldsymbol{p}}_{\boldsymbol{d}}}\right) \end{cases} \quad (4-1)$$

and

$$\boldsymbol{R}_{\boldsymbol{d}} = expm\left([0\ 0\ \widehat{\gamma}]^T\right);\ \boldsymbol{\omega}_{\boldsymbol{d}} = [0\ 0\ 0]^T \quad (4-2)$$

$\ddot{\boldsymbol{p}}_{\boldsymbol{d}}$ refers to the given maximum COM acceleration to prevent the robot from losing control due to excessive acceleration at the beginning of the robot's operation.

The calculation of the desired input vector $\boldsymbol{u}_{\boldsymbol{d}}$ and the foot trajectory of the leg in the air phase must refer to the different phases of the four legs. In Figure 4.2 the flow chart within the FSM is introduced, through which the specific phase of the four legs at certain times can be determined.

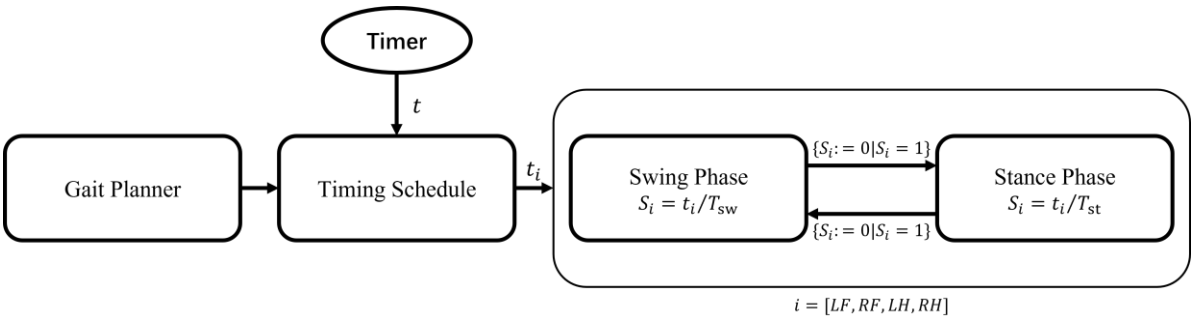


*Figure 4.2: Flowchart within the FSM. The set of timing schedules to use is determined by pre-selecting gait. The timing schedule determines the current state of the four legs according to the current time, that is, swing phase (air phase) or stance phase. Finally, the variable s is used to determine the specific timing position of each leg in its current state.*

The timing schedule in FSM refers to a specific gait cycle of a certain gait, see Chapter 2.2. It has a specific cycle length value and duty cycle value, which are provided by the user in advance. Through the current time $t$ indicated by the timer, the current phase of each leg (swing phase or stance phase) and the cumulative duration of the legs under the current phase $t_i$ can be determined according to the timing schedule. The percentage timing position $s$ of the leg at the current phase can then also be calculated via $s_i = t_i \,/\, T_{sw|st}$.

The desired input vector of the robot $\boldsymbol{u}_{\boldsymbol{d}}$ can be calculated based on the phase of the legs:

$$u_{d,i}^{x|y} = 0;\ u_{d,i}^{z} = \frac{b_i}{\sum_{i=1}^{4} b_i} \cdot Mg \quad (4-3)$$

The superscript $x, y, z$ indicates the direction of the GRF, $i$ indicates the index of the leg, and $b_i$ is a binary value, that indicates whether the i-th leg is in the stance phase. For the leg in the stance phase, the corresponding $b_i$ has the value 1, otherwise it has a value 0. This equation ensures that the legs in contact with the ground can bear the weight of the robot without causing the robot to fall to the ground.

Notably, the results of the equation (4-3) are only reference values and serve as the input of MPC. The optimal GRF obtained through MPC will also consider the requirement of the

COM forward velocity, rotary velocity etc.

The calculation method of the foot trajectory is based on the literature [61], and its composition is related to the current velocity of the shoulder or hip joint and the projected position of the joint on the ground, as well as the required velocity of the shoulder or hip joint and the projected position of the joint on the ground:

$$p_{step}^{f} = p^{h} + \frac{T_{st}}{2} \cdot \dot{p}_{d}^{h} + \sqrt{\frac{z_0^h}{g}} \cdot \left(\dot{p}^{h} - \dot{p}_{d}^{h}\right) \qquad (4-4)$$

$p_{step}^{f}$ is the desired foot position projected on the ground plane. $p^h$ and $\dot{p}^h$ are the projections of the shoulder or hip joint on the ground plane and its corresponding velocity; $\dot{p}_d^h$ is the desired shoulder or hip velocity projected on the ground plane; $T_{st}$ is the stance time in one gait cycle; $g$ is the gravitational acceleration constant; $z_0^h$ is the nominal shoulder or hip height, namely the height, that the joints and body COM are expected to maintain during the robot‘s motion.

For SRB, the relative positional relationship between the shoulder or hip joint and the body COM remains unchanged; thus, the above joint-related position and velocity parameters can be solved based on the position and velocity of the COM.

The desired foot height is related to a predetermined Bezier curve and the variable $s$ in Figure 4.2, in which the calculation formula of the Bezier curve is as follows:

$$b = \sum_{k=0}^{M} \alpha_k \cdot \frac{M!}{k! \cdot (M-k)!} \cdot s^k \cdot (1-s)^{M-k} \qquad (4-5)$$

Above $\alpha$ represents a vector whose length is $(M+1)$, and $s$ is a number between [0, 1]. The output $b$ of the Bezier curve has the value $\alpha_1$ when $s = 0$, and starts to approach $\alpha_2$, $\alpha_3$, etc. as $s$ increases. Ultimately $b$ has the value $\alpha_{M+1}$ when $s = 1$.

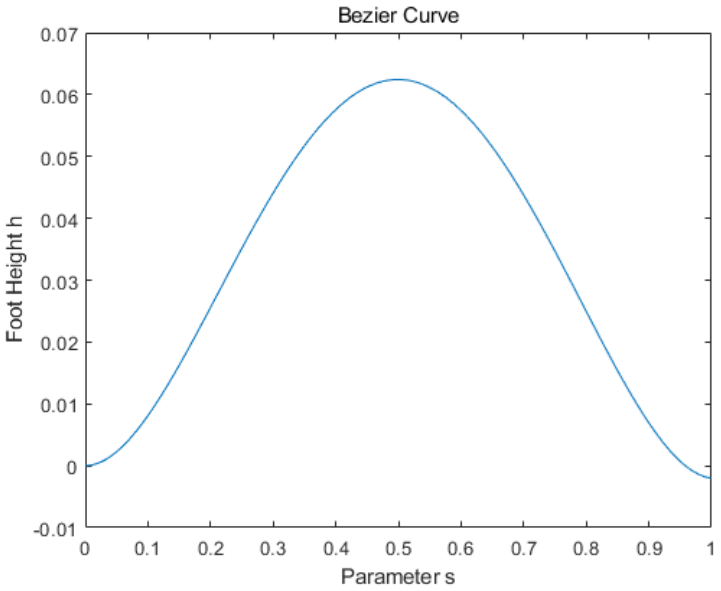


*Figure 4.3: The Bezier curve of the relationship between variable s and foot height.*

An example of the desired foot height is presented in Figure 4.3. When the variable $s$ is close to 0.5, that is, when the current phase of the corresponding leg is located at the center of the swing phase, the height of the foot from the ground is the highest, specifically about 0.06 meters. When the variable $s$ is close to 0 or 1, that is, when the current phase of the leg is at the border point of the stance phase and the swing phase, this means that the foot is in contact with the ground.

Based on the desired foot height and the desired foot position projected on the ground plane $p_{step}^{f}$, the foot-end trajectory of the legs in the stance (air) phase $\boldsymbol{p_{FP,i}}$ in Figure 4.1 can be determined.

### 4.1.2 Impulse Principle for Periodic Bounding

Through the formula introduced in the previous section, namely from (4-1) through (4-3), and the various gait cycles introduced in section 2.2, the desired state and input vectors for realizing various gaits can already be calculated. However this algorithm does not provide the accurate desired values for MPC. For example, only the rough value of the vertical component of the GRF in the desired input vector is calculated in formula (4-3). Although such an algorithm can finally achieve the desired gait, the control effect is relatively poor, for example it cannot enable the stable, high-speed and long-term movement of the quadruped robot. This problem is particularly obvious when realizing a dynamic gait, such as a bounding gait.

To solve such a problem, this section introduces a more advanced algorithm that provides the desired state and input vectors for MPC to better realize a bounding gait. This algorithm was proposed by the literature [62] and is called the Impulse Principle.

Regarding the bounding gait, due to the synchronous movement of its two front legs and the synchronous movement of its two rear legs, it is possible to simplify the robot's motion as a two-dimensional movement on the x-z plane. The equation for the robot‘s motion can be expressed as a function of the GRFs of both the front and rear legs in two directions, namely $F_x, F_z$, as well as the alteration of the COM pitch angle $\theta$. The equations for motion can be expressed as follows:

$$m\ddot{x} = F_x = \frac{xF_z + \tau}{z} \; with \; z > 0 \qquad (4-6)$$

$$m\ddot{z} = F_z - mg \qquad (4-7)$$

$$I\ddot{\theta} = \tau \qquad (4-8)$$

Above, $\ddot{x}$ and $\ddot{z}$ represent the acceleration of the COM in the horizontal and vertical dimensions, while $x$ and $z$ indicate the horizontal and vertical distances between the COM and the front or rear foot. Additionally, $m$ denotes the weight of the robot, while $\tau$ represents the current pitch angle of the robot. The torque acquired currently is equivalent to the GRF experienced by the front or rear foot multiplied by the moment arms $x$ and $z$ corresponding to the COM. $g$ denotes the gravity acceleration and $I$ denotes the moment of inertia.

In addition, to ensure sustained stability during the gait, it becomes necessary to impose certain constraints upon the aforementioned equations:

$$\int_0^T \left(-mg + F_z(t)\right)dt = \int_0^T m\ddot{z} = 0 \qquad (4-9)$$

$$\int_0^T \tau(t)dt = \int_0^T I\ddot{\theta} = 0 \qquad (4-10)$$

$$F_z(t) \geq 0, \forall t \in [0, T] \qquad (4-11)$$

Above, $T$ designates the duration of a gait cycle. Formulas (4-9) and (4-10) serve to guarantee that throughout a gait cycle, the cumulative alteration in vertical velocity and angular velocity of the pitch angle of the COM equals zero. This means that, no cumulative error exists after each gait cycle that could cause the robot to lose control after a series of gait cycles. Formula (4-11) ensures that the GRF in the vertical dimension consistently points upward.

As indicated by the aforementioned motion equations, it can be observed that the entire motion of the robot can be controlled by two variables over time, namely $F_z(t)$ and $\tau(t)$. It is necessary to predetermine these two time-varying parameters, but during a cycle their alteration over time is closely associated with the robot's condition. Hence, before defining the values of these two parameters, it is pertinent to introduce the state division of the bounding gait cycle.

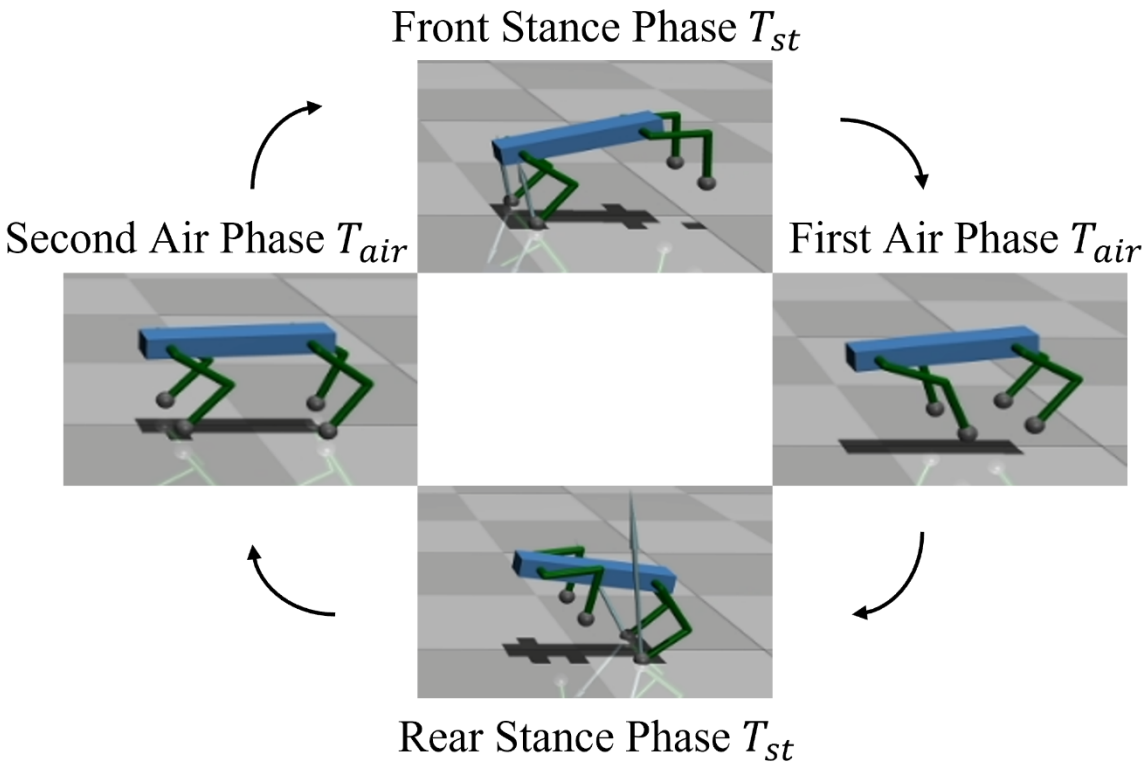


*Figure 4.4: Quadrupedal bounding gait cycle. By assuming massless legs, the quadruped is abstracted through a planar rigid body, which evolves under the impact of GRFs and gravity. A fixed horizontal foot placement relative to the hip at the onset of the stance is utilized to establish the point of application for the GRF.*

Based on Figure 4.2, this section considers the phase matching of the front and rear feet and divides a bounding gait cycle into four distinct phases, including the “Front stance phase”,

"first air phase", "rear stance phase" and "second air phase" in Figure 4.4. Both stance phases are of equal duration, represented as $T_{st}$, and both air phases are similarly of identical length, denoted as $T_{air}$. The duration of the swing phase for both the front and rear legs is equal to $T_{sw} = T_{st} + 2T_{air}$, while the complete gait cycle has a total duration of $T = 2T_{st} + 2T_{air}$.

Subsequently, the trajectory of the parameters $F_z(t)$ and $\tau(t)$ throughout a gait cycle can be described as follows:

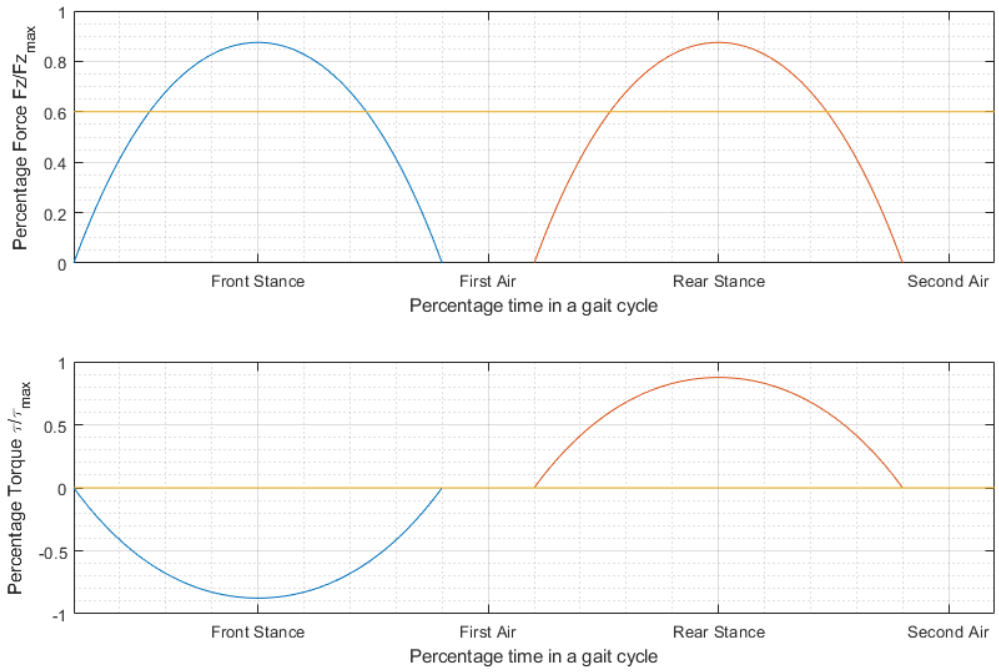


*Figure 4.5: The profile shape of the generalized vertical component of GRF $F_z(t)$ and torque $\tau(t)$, which is calculated though the formula (4-5) concerning the Bezier curve and satisfies the requirements of formulas (4-10) and (4-11). During the "Front stance phase" the blue curve illustrates the vertical component of the GRF of the front leg and the torque of the GRF acting on the body COM over time. In the "First air phase" neither leg experiences stress. The "Rear stance phase" corresponds to the stage of stress on the rear leg. The shape of the curve in the figure is determined, but the actual value of the parameter $F_z(t)$ and $\tau(t)$ depends on the maximum $F_{z,max}$ and $\tau_{max}$ prescribed by the user. Furthermore, the yellow line in the upper subgraph represents the ratio of the robot's gravity to the $F_{z,max}$, and the yellow line in the lower subgraph represents the desired torque received by the robot, namely 0.*

To satisfy formula (4-9), the following relationship can be obtained:

$$2 \cdot c \cdot F_{z,max} \cdot T_{st} = mgT \qquad (4-12)$$

which

$$c = mean(\alpha) \qquad (4-13)$$

Among them, $\alpha$ is a vector comprised of a series of target points along the Bezier curve, see equation (4-5), and its mean value is equivalent to the integral of the curve. With the given value of $T_{st}$ and $T$, $F_{z,max}$ can then be expressed as follows:

$$F_{z,max} = \frac{mgT}{2 \cdot c \cdot T_{st}} \qquad (4-12)$$

In contrast, the determination of $\tau_{max}$ is not certain. Any value of $\tau_{max}$ within a certain range can achieve stable bounding gait. The value of $\tau_{max}$ only affects the swing amplitude of the robot in the direction of the pitch angle.

## 4.2 Force Control

The preceding part of MPC within the control loop was discussed in section 4.1. This section elaborates upon how to utilize the optimization outcomes of MPC to control the robot.

The results of the MPC algorithm are illustrated in Figure 2.6, comprising the optimal state and input vectors, namely $\boldsymbol{x_{optimal}}, \boldsymbol{u_{optimal}}$. $\boldsymbol{u_{optimal}}$ represents the GRF required by the four legs, which must be transformed into the joint torque required by both the simulation and actual robots using the kinematic model of the legs. The calculation method for this step was already introduced in section 3.1.4. For legs with varying structures, one can adopt the same approach outlined in section 3.1.4 to determine the relationship between the force exerted on the foot and the joint torque.


*Figure 4.6: Flow chart from optimal GRF to joint torque.*

## 4.3 Swing Leg Control

Formula (4-3) reveals that when the i-th leg loses contact with the ground, i.e., during the swing phase, the corresponding desired GRF $\boldsymbol{u_{d,i}}$ is 0. Therefore, the optimal GRF obtained via MPC optimization $\boldsymbol{u_{optimal,i}}$ is also 0. Nevertheless, during the swing phase, the legs must still conform to the foot trajectory $\boldsymbol{p_{FP,i}}$ specified by the FSM (see Figure 4.1). Specifically, the legs must extend in the direction of the motion during the swing phase to meet the requirement of propelling the body forward by kicking the ground backward during the next stance phase.

In scenarios in which the desired position of the foot $\boldsymbol{p_{FP,i}}$ is known, it is possible to utilize the inverse kinematics method discussed in section 3.1.2 to convert the desired foot‘s position in the desired rotary angle of the joint $\boldsymbol{\varphi_d}$. $\boldsymbol{\varphi_d}$ can then be compared with the current joint angle $\boldsymbol{\varphi}$ to derive an error term $(\boldsymbol{\varphi_d} - \boldsymbol{\varphi})$. Furthermore, the joint angle at the preceding moment (last iteration) can be recorded in the control program, enabling the calculation of the current joint angular velocity $\dot{\boldsymbol{\varphi}}$. Subsequently, the desired joint torque can be determined based on the above information:

$$\boldsymbol{\tau_{i,ST}} = K_P \cdot (\boldsymbol{\varphi_d} - \boldsymbol{\varphi}) + K_d \cdot (\mathbf{0} - \dot{\boldsymbol{\varphi}}) \qquad (4-13)$$

which

$$\dot{\boldsymbol{\varphi}} = \frac{\boldsymbol{\varphi} - \boldsymbol{\varphi_{pre}}}{timestep} \qquad (4-14)$$

Notably, multiple joints provide torque on each leg (commonly two to three joints); thus, the variables in formulas (4-13) and (4-14) are in the form of vectors.

The values of gain parameters (PD control parameter) $K_P$ and $K_d$ vary according to different leg structures and joint parameters (stiffness or damping). Hence, it is essential to conduct actual experimental tests to determine the appropriate magnitudes of $K_P$ and $K_d$ after constructing the robot platform.

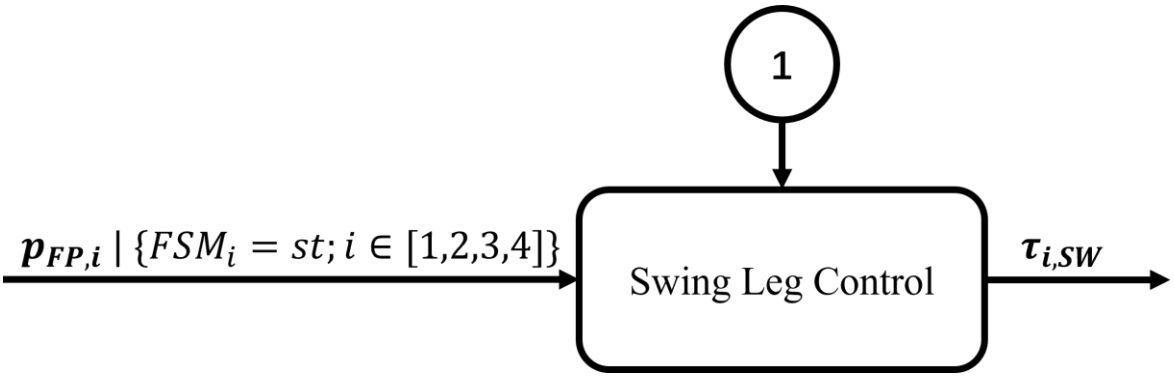


*Figure 4.7: Flow chart from reference foot trajectory to swing leg joint torque.*

Given the elliptical form of the trajectory of the quadruped robot's foot during the robot‘s motion, the ability of the foot to track a prescribed elliptical trajectory was evaluated in the actual experimental tests.

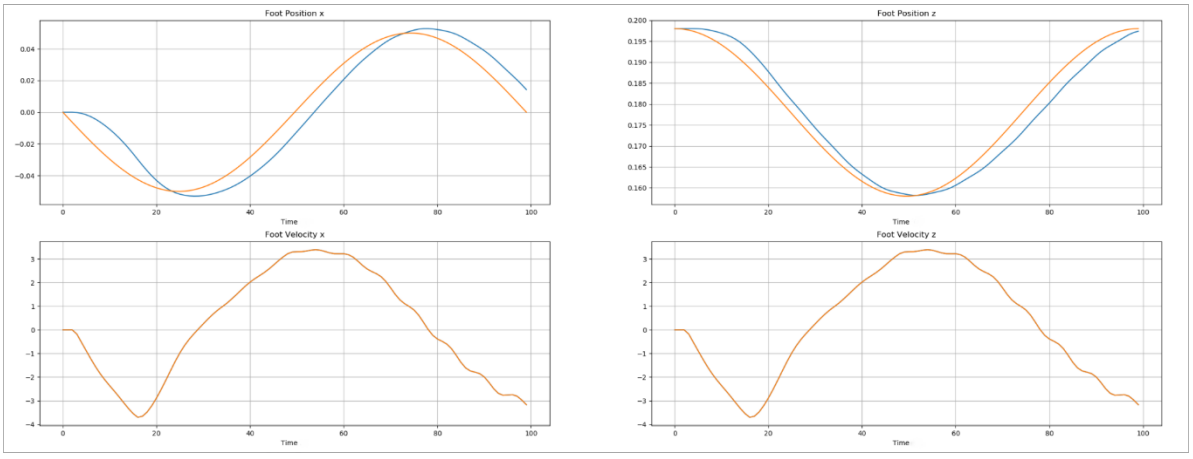


*Figure 4.8: Above are position and velocity diagrams that depict the results of the foot-end trajectory tracking experiment. The orange line illustrates the desired position and velocity of the foot-end following the prescribed elliptical trajectory along the x and z axes over time. Conversely, the blue line portrays the real position and velocity of the foot-end under the PD control along the x and z axes over time. The horizontal axis corresponds to the number of timesteps, each of which lasts for 0.001 seconds. Compared with a swing phase of the leg of 0.1s-0.4s normally, that is, 100 to 400 timesteps, the current PD control results are acceptable.*

Herein, an illustration of a PD control parameter test is executed in MuJoCo. Notably, the presented example serves solely as an instance of PD control parameter testing. Further experimentation about tracking the elliptical trajectory to re-evaluate the PD control parameters whenever modifications are made to the robot parameters.

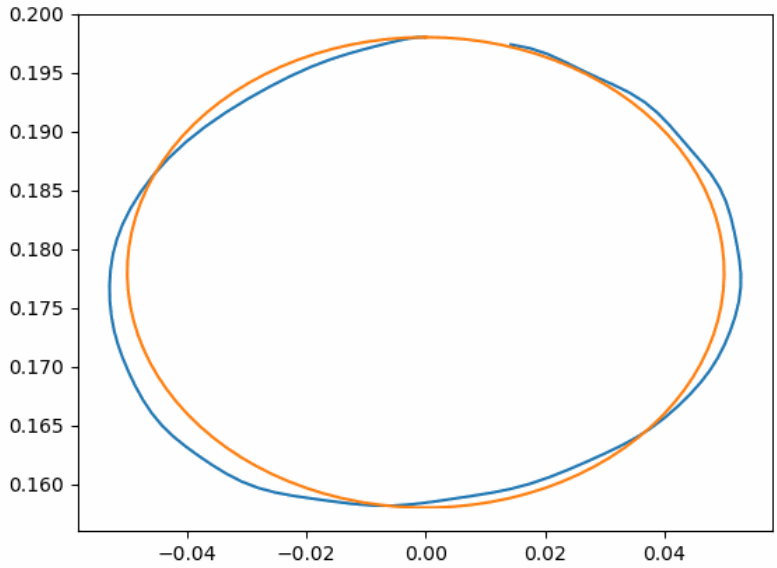


*Figure 4.9: A x, z-axis plan view of the foot-end trajectory. The orange line illustrates the desired foot-end prescribed elliptical trajectory. Conversely, the blue line portrays the real foot-end trajectory under the PD control.*

As evidenced by the results depicted in Figures 4.8 and 4.9, while the trajectory tracking performance is not flawless, the legs are capable of accomplishing most of the trajectory tracking tasks and of maintaining a stable running trajectory without vibrations under the control of the present PD controller. This observation supports the view that the currently employed PD control parameters are acceptable.

## 4.4 Initial Standing State

Thus, various components of the complete control loop have been elucidated. However, an algorithm that governs the quadruped robot's stable stance and that prevents it from toppling to the ground under the influence of gravity in the absence of joint torque is still missing. Achieving a stable standing posture is considered the precursor to subsequent gait control.

The task of achieving stable standing can be accomplished via the PD joint angle control algorithm, as expounded upon in section 4.3. Nevertheless, an alternative and more intuitive algorithm referred to as Virtual Machines Control (VMC) is employed in this study and its expression is as follows:

$$\boldsymbol{\tau_i} = J^T \cdot (K_{P,init} \cdot (\boldsymbol{p_d} - \boldsymbol{p}) + K_{d,init} \cdot (\mathbf{0} - \dot{\boldsymbol{p}})) \qquad (4-15)$$

Among them, $\boldsymbol{p_d}$ denotes the desired position of the COM, $\boldsymbol{p}$ represents the present COM

position, and $\dot{\boldsymbol{p}}$ denotes the current velocity of the COM. $J^T$ corresponds to the foot Jacobian matrix, as elucidated in section 3.1.4, while $\boldsymbol{\tau_i}$ represents the joint torque of the legs. As emphasized in Chapter 4.3, the values of $K_{P,init}$ and $K_{d,init}$ also must be adjusted in the experiment.

An additional issue that warrants attention is the inability of the robot to precisely attain the desired COM position due to the impact of gravity under the control of VMC. In general, the actual COM position of the robot under the control of VMC along the z-axis is slightly lower than the desired value. To circumvent this difficulty, the $\boldsymbol{p_d}$ value in the z-axis can be suitably increased to counterbalance the impact of gravity.

## 4.5 Complete Control Loop

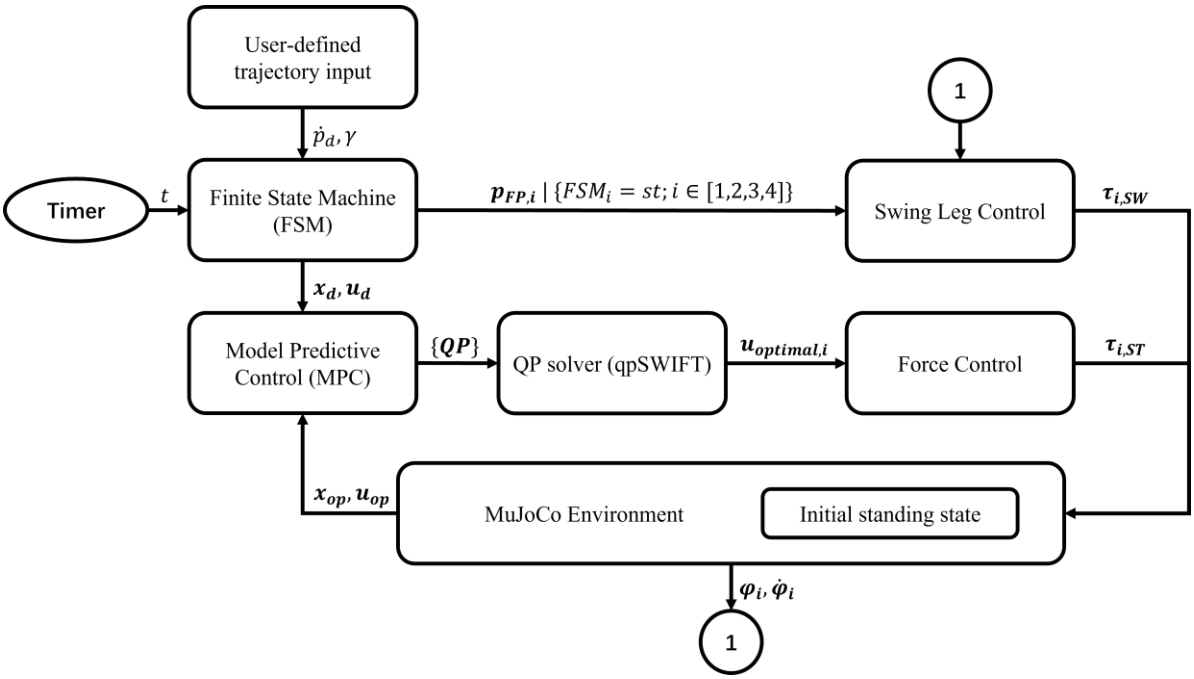


*Figure 4.10: Flowchart of the complete control loop connecting to the MuJoCo environment.*

# 5 Experiments

Drawing upon the foundational knowledge and formulae presented in earlier chapters, this chapter aims to realize some gaits of quadruped robots within the MuJoCo simulation environment.

## 5.1 Basic parameters of quadruped robot

| | |
|---|---|
| Body Length | $0.301\ m$ |
| Body Width | $0.088\ m$ |
| Body Height | $0.05\ m$ |
| Body Mass | $5.5\ kg$ |
| Body Moment of Inertia (x-axis) | $4.695e-3\ kg \cdot m^2$ |
| Body Moment of Inertia (y-axis) | $7.632e-2\ kg \cdot m^2$ |
| Body Moment of Inertia (z-axis) | $7.873e-2\ kg \cdot m^2$ |
| | |
| Upper Leg Length | $0.14\ m$ |
| Upper Leg Mass | $0.055\ kg$ |
| Lower Leg Length | $0.14\ m$ |
| Lower Leg Mass | $0.055\ kg$ |
| Leg Link Hip Length (for 3 DOFs) | $0.05\ m$ |
| Leg Link Hip Mass (for 3 DOFs) | $0.02\ kg$ |
| | |
| Simulation Timestep | $0.001s$ |
| Gravity acceleration | $9.81\ m/s^2$ |
| Friction Coefficient | 1 |
| Nominal COM Height | $0.2\ m$ |
| Leg Joint Stiffness | 0 |
| Leg Joint Damping | 0.1 |
| Upper Leg Joint Torque | $50\ Nm$ |
| Lower Leg Joint Torque | $50\ Nm$ |
| Hip Leg Joint Torque (for 3 DOFs) | $150\ Nm$ |
| | |

| | |
|---|---|
| Gain Value $K_{P,init}$ | 1000 |
| Gain Value $K_{d,init}$ | 40 |
| Gain Value $K_P$ | 1600 |
| Gain Value $K_d$ | 5 |
| Gain Value $K_P$ (for the 3rd DOF) | 6400 |
| Gain Value $K_d$ (for the 3rd DOF) | 20 |

*Table 5.1: Parameter configurations for both the simulated robot and MuJoCo environment. Specifically, the parameter settings are mainly derived from the MIT Panther robot's physical specifications but have been suitably adapted to conform to the requirements of the simulation.*

## 5.2 Walk Trot for Robot with Two-DOF Leg

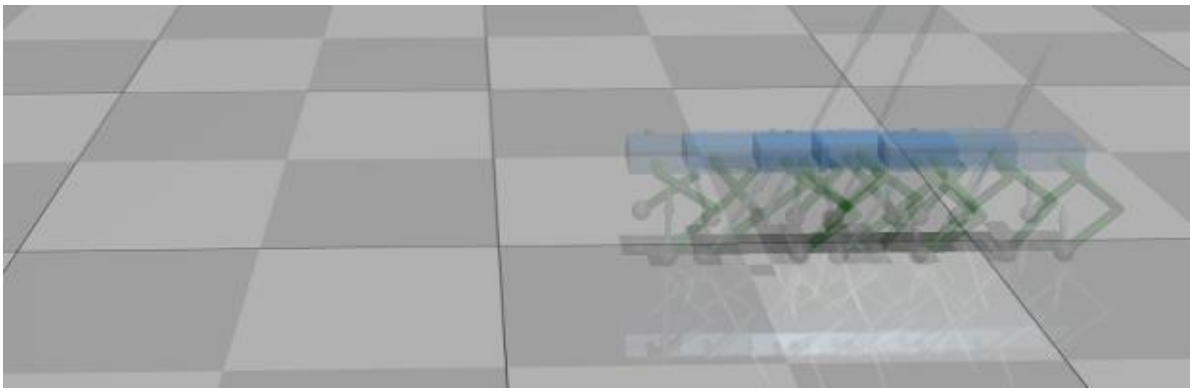

*Figure 5.1: Exposure image of the Walk Trot gait in the simulation environment.*

To achieve the walk trot gait of a quadruped robot equipped with 2-DOF legs, a target COM velocity of 0.2m/s is selected, with a maximum COM acceleration of 1m/s^2. Given that the leg joints of the 2-DOF share the same rotation direction, whereby the legs move solely within the x-z plane, the variation in the robot's yaw angle is deemed negligible. Consequently, only the robot's horizontal forward movement is taken into account. The simulation results are illustrated in figures 5.2 to 5.4.

Among them, the most important results are located in the upper left subplot in Figure 5.2, which showcase the COM position of the robot along the x-axis, namely the forward movement direction, within the world coordinate system. As observed from the graph, the robot's capacity to closely track the target COM position with minimal variance is readily apparent. This result serves as compelling evidence of the veracity of the formulaic derivations and control loop construction detailed in previous chapters. Moreover, it attests to the robot's ability to sustain a stable gait and desired velocity.

Furthermore, it should be noted that certain parameters have a desired value of 0, but during actual motion, their corresponding real values cannot be maintained at precisely 0. Such parameters include the COM Euler angle and COM Euler angular velocity, see the third and fourth column subplots of Figure 5.2.

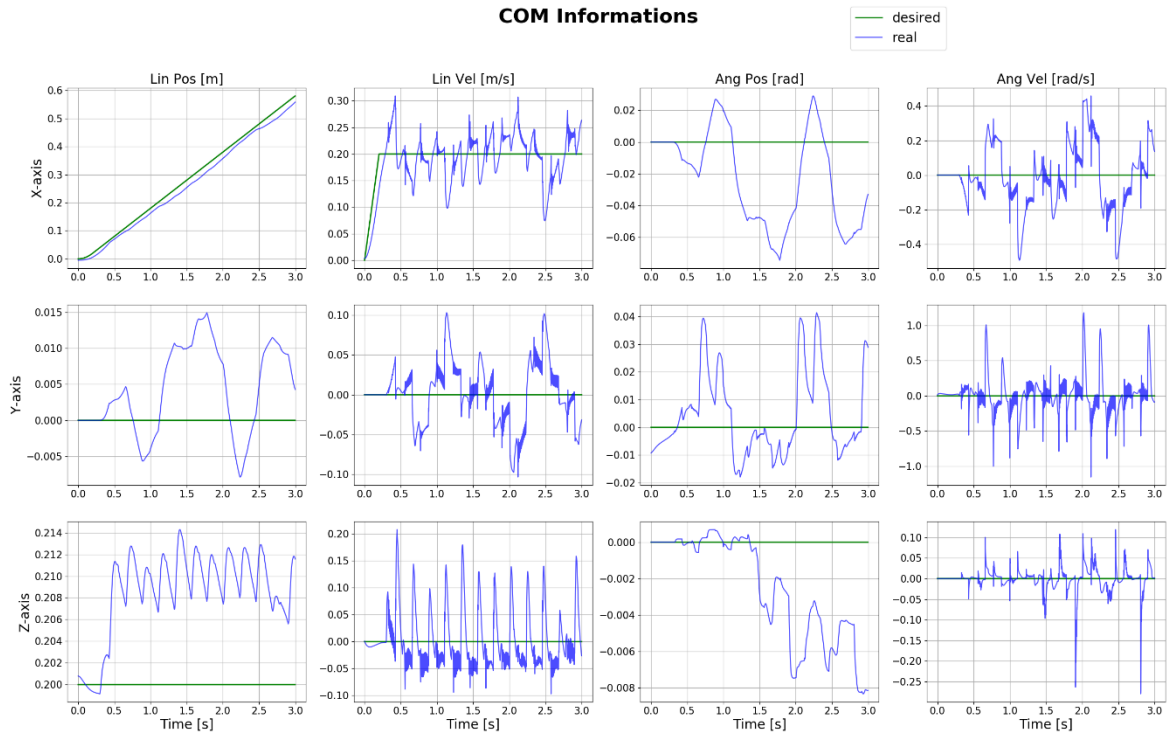


*Figure 5.2: Results about the robot COM. "Lin Pos" for COM linear position in global frame; "Lin Vel" for COM linear velocity in global frame; "Ang Pos" for COM Euler angle in global frame; "Ang Vel" for COM angular velocity in body frame. The results in 3-dimension are plotted in 3 figures from top to bottom respectively.*

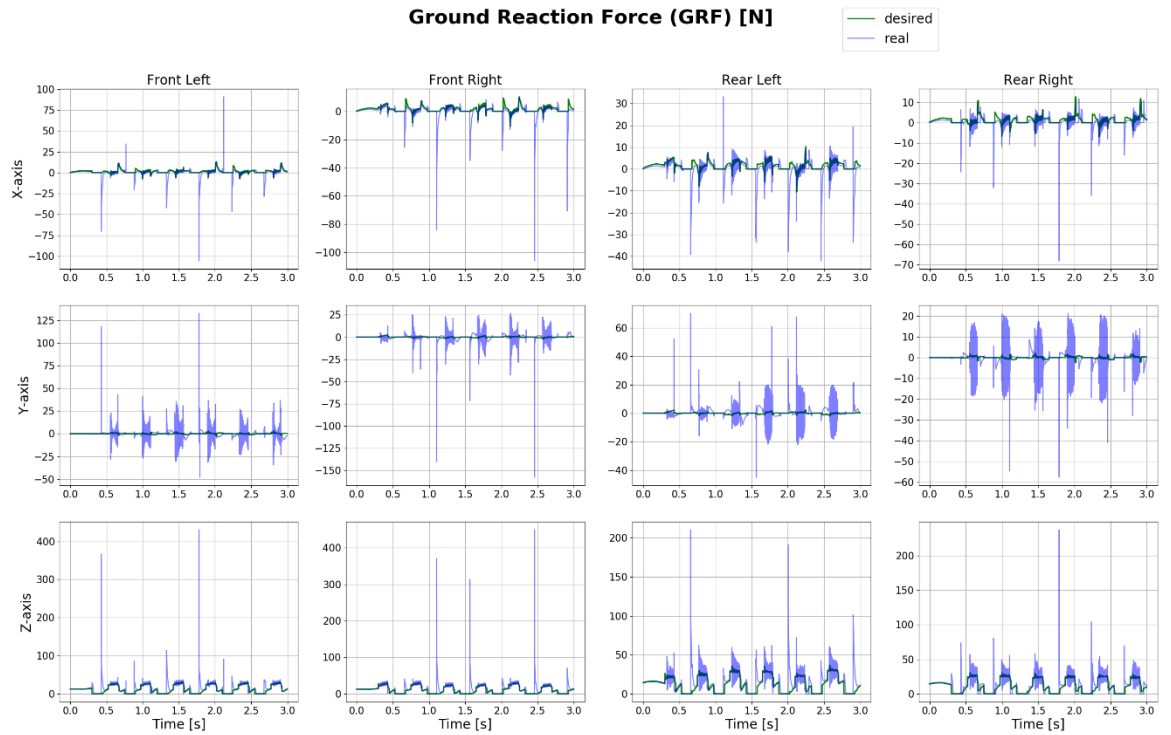


*Figure 5.3: Results about the GRFs of the four legs. The results in 3-dimension are plotted in 3 figures from top to bottom respectively.*

In addition, in some subfigures, although the fluctuation of real value seems to be relatively large, attention should be paid to the numerical range of the ordinate, the absolute error is actually very small, such as subfigure at position (2, 1), (3, 1) etc. of Figure 5.2.

Figure 5.3 demonstrates that the error of the GRF is minute, which verifies the viability of calculating the joint torque that corresponds to the desired GRF via the foot Jacobian matrix. And the results depicted in Figure 5.4 evince that the error in the desired foot end position is quite small, confirming the feasibility of tracking a desired foot end trajectory via PD control.

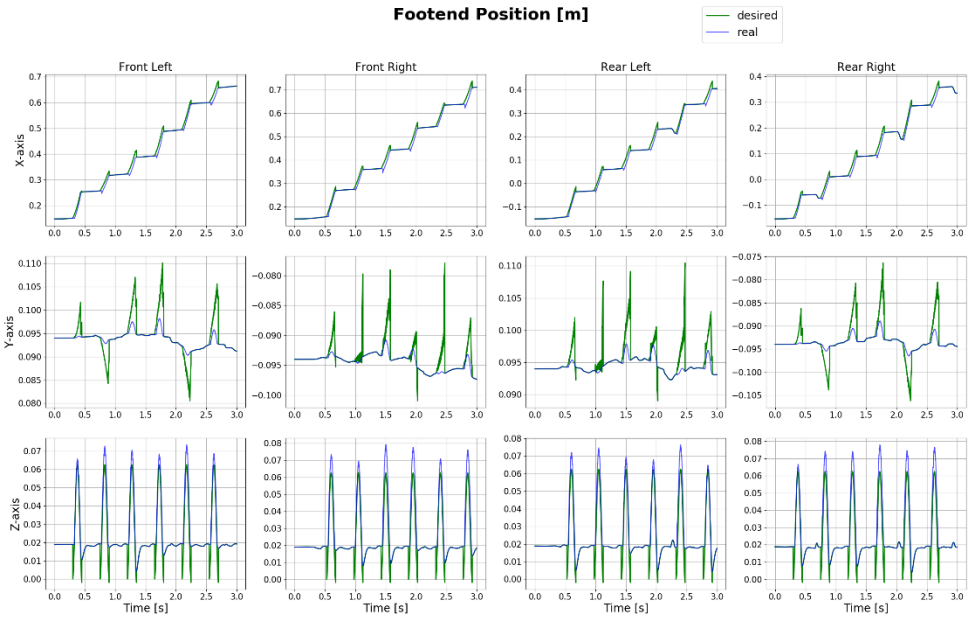


*Figure 5.4: Results about the foot position related to respective hip joint. The results in 3-dimension are plotted in 3 figures from top to bottom respectively. From left to right are the results of 4 legs.*

## 5.3 Bounding for Robot with Two-DOF Leg

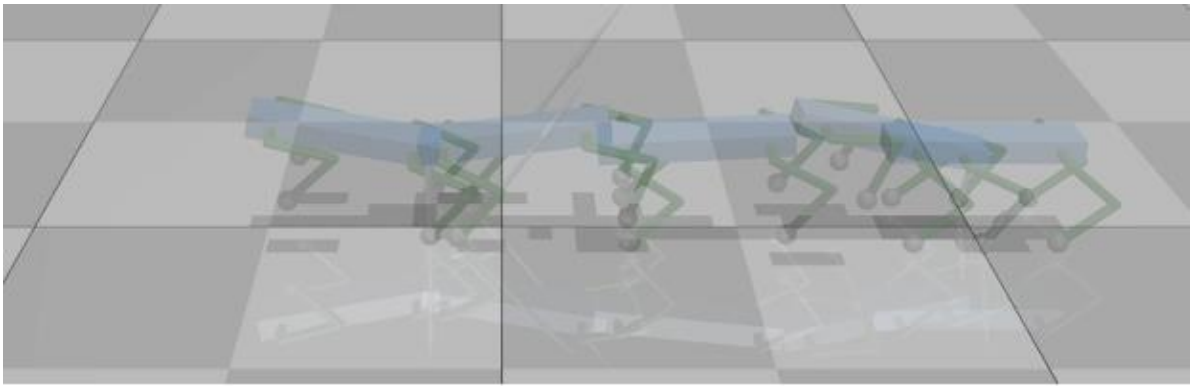

*Figure 5.5: Exposure image of the Bounding gait in the simulation environment.*

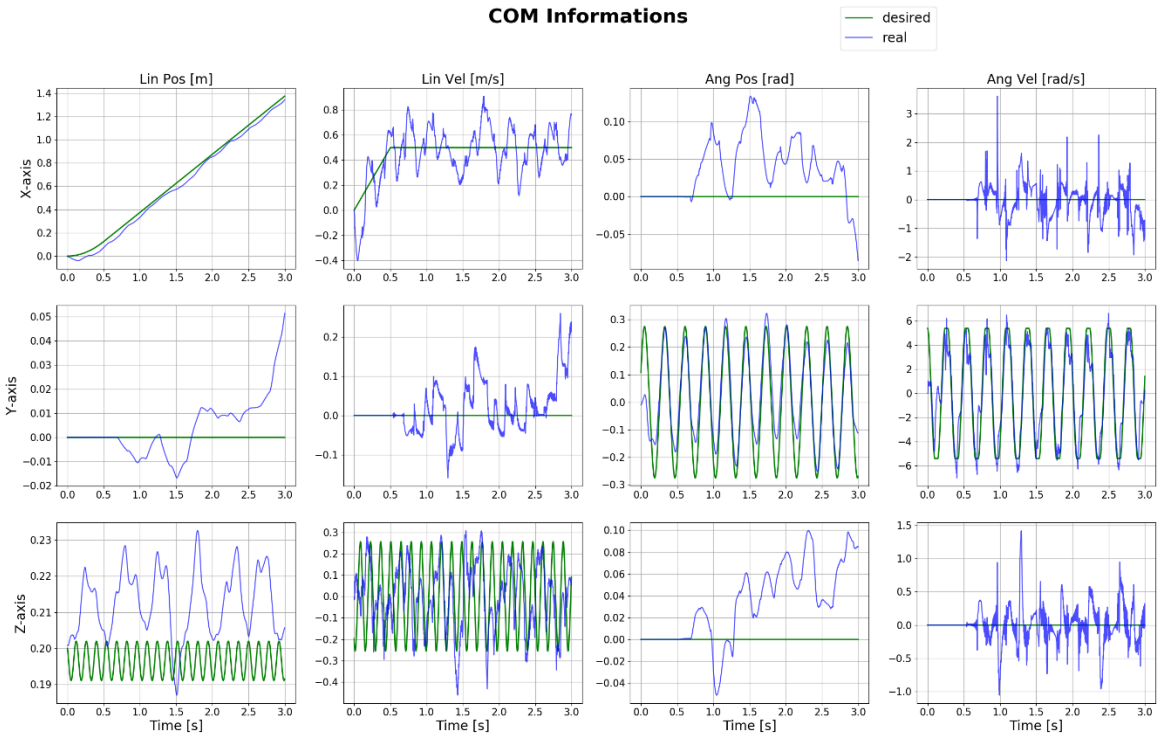


*Figure 5.6: Results about the robot COM. "Lin Pos" for COM linear position in global frame; "Lin Vel" for COM linear velocity in global frame; "Ang Pos" for COM Euler angle in global frame; "Ang Vel" for COM angular velocity in body frame. The results in 3-dimension are plotted in 3 figures from top to bottom respectively.*

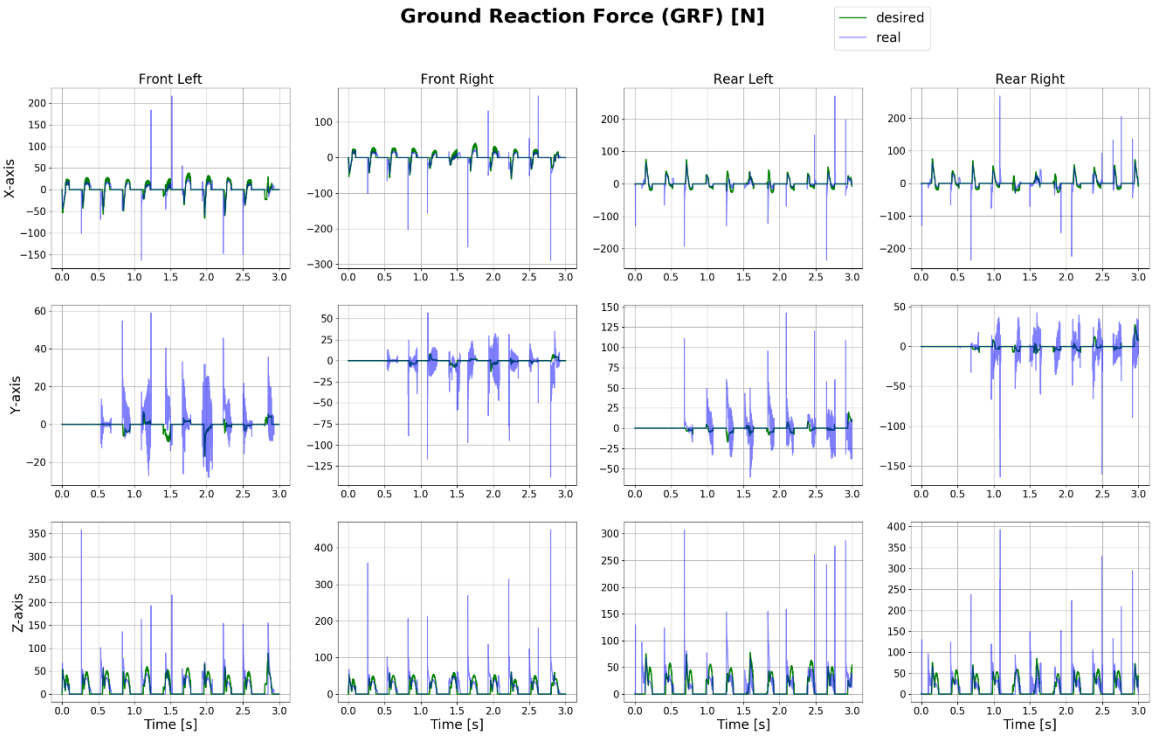


*Figure 5.7: Results about the GRFs of the four legs. The results in 3-dimension are plotted in 3 figures from top to bottom respectively.*

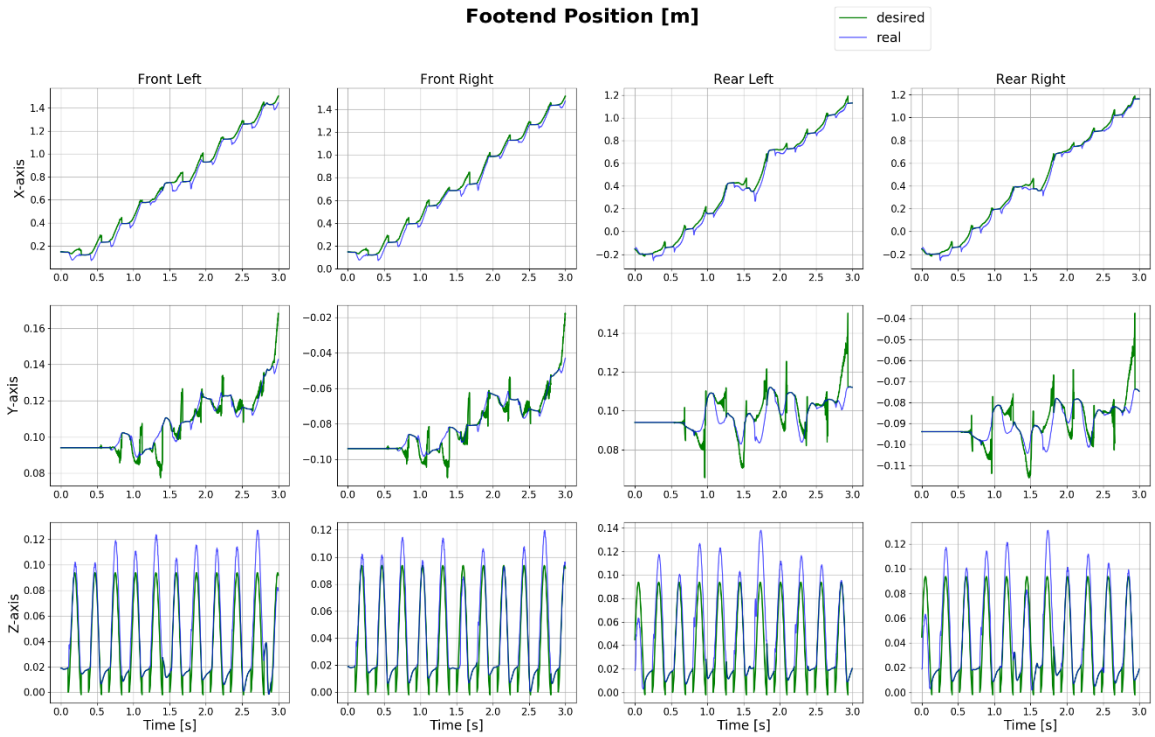


*Figure 5.8: Results about the foot position related to respective hip joint. The results in 3-dimension are plotted in 3 figures from top to bottom respectively. From left to right are the results of 4 legs.*

This thesis has also implemented bounding gait control through MPC. Compared to walk trot, the challenging aspect of bounding gait is that it is a dynamic gait featuring air phases in the gait cycle. Consequently, the joints of the robot experience relatively larger torque and more impact. But the results displayed in Figures 5.6 to 5.8 demonstrate that the quadruped robot can still maintain a stable gait and follow the desired trajectory.

# 6 Discussion

Based on the experimental findings presented in the preceding chapter, it is evident that the same structure of MPC control algorithms can effectively regulate various robot structures and different gaits, which is a significant advantage when compared to conventional quadruped robot control algorithms. A conventional quadruped robot control algorithm can generally realize only one gait of a specifical robot design. By inputting a rough gait feature and the corresponding high-level motion trajectory, MPC can derive the values of detailed parameter for the targeted gait with considering of the current robot situation. For instance, as highlighted in chapter 4, only the ground reaction force required for vertical support and the forward speed and acceleration of the robot are needed for MPC. Then MPC can calculated accurate ground reaction forces in all directions of the four legs. This obviates the need for exact kinematic and dynamic calculations to control gait movements.

Furthermore, the MPC framework offers modular flexibility, enabling the addition or removal of arbitrary constraints by simply introducing or eliminating a constraint equation in the optimization equation. In addition, more precise gait control can be achieved by conducting more accurate precomputation, as mentioned in section 4.1. With this method a bounding gait characterized by more stable motion and higher operating speeds can be realized. Moreover, the swing leg control and force control logic can be modified, for instance, by replacing PD control with more accurate dynamic calculation. This modular modification feature renders the MPC control becomes a basic and general framework that can be improved to accommodate any additional requirements without necessitating the redesign of a whole new control algorithm.

Furthermore, the experimental results revealed that some gait cycles during simulation exhibited less stable robot postures, such as asynchronous contact of the hind legs with the ground during bounding gait, or premature contact of a leg with the ground during the flight phase. These instabilities may result in unexpected robot postures in the next gait cycle. However, MPC can compensate for these instabilities in the subsequent timesteps or gait cycles, leading the robot to return to a stable state after one or two gait cycles.

Finally, it should be noted that the MPC algorithm contains a large number of gain values, such as those listed in Appendix 1. The adjustment of these parameters significantly impacts the final control performance. Compared to traditional optimization algorithms, the MPC algorithm requires more time and attention to parameter tuning.

# 7 Conclusion and Outlook

This paper presents an integration of the popular Model Predictive Control (MPC) algorithm for quadruped robot control within the MuJoCo simulation platform and enable the control of quadruped gaits through MPC in MuJoCo. This provides basic control codes and valuable experience for controlling the bionic mouse robot Nermo developed by the chair through MPC.

Furthermore, this paper presents a diverse range of calculation techniques for kinematic and dynamic formulas of the legs and spine. These formulas can be incorporated into the MPC framework in future studies to address more intricate motion control demands.

About the outlook, the present MPC-based control framework still presents challenges in achieving high-speed gait control of quadruped robots, primarily owing to the lack of an accurate and stable leg control algorithm. The incorporation of leg dynamics equations into swing leg control can offer the potential for attaining higher-speed motion for quadruped robots.

Moreover, the Nermo robot exhibits additional degrees of freedom, including a flexible spine, a flexible tail, and a rotatable head. The influence of these degrees of freedom on motion and their control can also be integrated into the MPC framework, thereby enhancing the efficacy of motion control.

During the implementation of the MPC algorithm to control the Nermo robot within the MuJoCo simulation environment, certain control issues can be encountered due to the robot's extremely low weight. Then when applying the MPC control to a physical Nermo robot, it is noted that the optimization algorithm places high demands on computing resources, which can lead the synchronization of the output frequency of the MPC control algorithm with the motor control frequency becoming a challenge.

In conclusion, the control of quadruped robots is a complex and promising field. The movement of real quadruped animals is influenced not only by their body state and desired gait, but also by sensory signals such as vision, hearing, touch, and even smell. Moreover, the final performance of animal movement does not adhere to a single gait but frequently switches between multiple gaits, indicating that current motion control methods based on kinematics and optimization algorithms is still far away from completely simulating the motion of real animals. The development of control algorithms has a long road ahead, and such research will contribute to a better understanding of animal movement and cognitive processes in nature.

# 8 Reference

**[1] A. De and D. E. Koditschek,** "Vertical hopper compositions for preflexive and feedback-stabilized quadrupedal bounding, pacing, pronking, and trotting," The International Journal of Robotics Research, vol. 37, no. 7, pp. 743–778, 2018.

**[2] C. Gehring, S. Coros, et. al.,** "Control of dynamic gaits for a quadrupedal robot," Robotics and Automation (ICRA), 2013 IEEE International Conference, pp. 3287–3292, 2013.

**[3] C. Semini, N. G. Tsagarakis, et. al.,** "Design of HyQ: a hydraulically and electrically actuated quadruped robot," Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, vol. 225, no. 6, pp. 831–849, 2011.

**[4] M. Raibert, K. Blankespoor, et. al.,** "Bigdog, the rough/terrain quadruped robot", Proceedings of the 17th World Congress, pp. 10823-10825, 2008.

**[5] Boston Dynamics,** Cheetah robot runs 28.3 mph; a bit faster than usain bolt, Youtube Videos, online available: http://youtu.be/chPanW0QWhAa.

**[6] S. Seok, A. Wang, et. al.,** "Actuator design for high force proprioceptive control in fast legged locomotion", in IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1970–1975, 2012.

**[7] S. Seok, A. Wang, et. al.,** "Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot", in IEEE International Conference on Robotics and Automation, 2013.

**[8] D. J. Hyun, S. Seok, et. al.,** "High speed trot running: Implementation of a hierarchical controller using proprioceptive impedance control on the MIT cheetah", 2014.

**[9] H. -W. Park and S. Kim,** "Quadrupedal galloping control for a wide range of speed via vertical impulse scaling", 2015 Bioinspir. Biomim. 10 025003, 2015.

**[10] H.-W. Park, M. Y. Chuah,et. al.,** "Quadruped bounding control with variable duty cycle via vertical impulse scaling", Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference, 2014.

**[11] Q. Nguyen, M. J. Powell, B. Katz, J. D. Carlo, and S. Kim,** "Optimized jumping on the mit cheetah 3 robot," in 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 7448–7454.

**[12] B. Katz, J. Di Carlo, and S. Kim,** "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 6295–6301.

**[13] M. H. Raibert,** "Trotting, pacing and bounding by a quadruped robot", Journal of Biomechanics, vol. 23, Supplement 1, no. 0, pp. 79 – 98, 1990.

**[14] L. D. Maes, M. Herbin, et. al.,** "Steady locomotion in dogs: temporal and associated spatial coordination patterns and the effect of speed", Journal of Experimental Biology, vol. 211, no. 1, pp. 138–149, 2008.

**[15] P. E. Hudson, S. A. Corr, et. al.,** "High speed galloping in the cheetah (acinonyx jubatus) and the racing greyhound (canis familiaris): spatio-temporal and kinetic characteristics", Journal of Experimental Biology, vol. 215, no. 1, pp. 2425–2434, 2012.

**[16] H. -W. Park, S. Park, et. al.,** "Variable-speed quadrupedal bounding using impulse planning: Untethered high- speed 3D Running of MIT Cheetah 2," 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 5163-5170, 2015.

**[17] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter,** "Learning agile and dynamic motor skills for legged robots," Science Robotics, vol. 4, no. 26, 2019.

**[18] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl,** "Online walking motion generation with automatic footstep placement," Advanced Robotics, vol. 24, no. 5-6, pp. 719–737, 2010.

**[19] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard,** "Whole-body model-predictive control applied to the HRP-2 humanoid," in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2015, pp. 3346–3351.

**[20] M. Neunert, M. St¨auble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli,** "Whole-body nonlinear model predictive control through contacts for quadrupeds," IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 1458–1465, 2018.

**[21] R. Grandia, F. Farshidian, A. Dosovitskiy, R. Ranftl, and M. Hutter, "**Frequency-aware model predictive control," IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1517–1524, 2019.

**[22] A. Rohregger,** "Digital Nermo - Development of a Digital Twin forthe Exploration & Evaluation of Tendon Driven Compliant Legs & Spine on the Locomotion Behaviour of the Bio-Inspired Robotic Mouse Nermo," Informatics 6 - Chair of Robotics, Artificial Intelligence and Real-Time Systems at the School of Information of TUM, 2021

**[23] J. Collins, S. Chand, A. Vanderkop, and D. Howard,** "A review of physics simulators for robotic applications," IEEE Access, 2021.

**[24] M. Hildebrand,** "Symmetrical gaits of horses: Gaits can be expressed numerically and analyzed graphically to reveal their nature and relationships", Science, vol. 150, no. 3697, pp. 701–708, 1965.

**[25] S. Coros, A. Karpathy, et al.,** "Locomotion skills for simulated quadrupeds", ACM Trans. Graph. 30, 4, Article 59, 2011.

**[26] K. Clarke and J. Still,** "Gait analysis in the mouse," Physiology & behavior, vol. 66, no. 5, pp. 723–729, 1999.

**[27] C. S. Mendes, I. Bartos, et. al.,** "Quantification of gait parameters in freely walking rodents," BMC biology, vol. 13, no. 1, pp. 1–11, 2015.

**[28] R. McNeill Alexander,** "Energetics and optimization of human walking and running: The 2000 raymond pearl memorial lecture," American journal of human biology, vol. 14, no. 5, pp. 641–648, 2002.

**[29] H.-W. Park, P. M. Wensing, S. Kim, et al.** Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. Robotics: Science and Systems, 2015.

**[30] H. -W. Park and S. Kim,** "Quadrupedal galloping control for a wide range of speed via vertical impulse scaling", 2015 Bioinspir. Biomim. 10 025003, 2015.

**[31] N. C. Heglund, C. R. Taylor, et. al.,** "Scaling stride frequency and gait to animal size: Mice to horses," Science, vol. 186, no. 4169, pp. 1112–1113, 1974.

**[32] M. S. Fischer, N. Schilling, et. al.,** "Basic limb kinematics of small therian mammals," Journal of Experimental Biology, vol. 205, no. 9, pp. 1315–1338, 2002.

**[33] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson,** "Optimization based full body control for the darpa robotics challenge," Journal of Field Robotics, 32(2):293–312, 2015.

**[34] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "**Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," Autonomous Robots, 40(3):473–491, 2016.

**[35] M. Mistry, J. Buchli, and St. Schaal,** "Inverse dynamics control of floating base systems using orthogonal decomposition," In 2010 IEEE international conference on robotics and automation, pages 3406–3412. IEEE, 2010.

**[36] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and St. Schaal,** "Optimal distribution of contact forces with inverse-dynamics control," The International Journal of Robotics Research, 32(3):280–298, 2013.

**[37] B. Henze, C. Ott, and M. A. Roa,** "Posture and balance control for humanoid robots in multi-contact scenarios based on model predictive control," In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pages 3253–3258. IEEE, 2014.

**[38] S. Kajita,** "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," In Proc. IEEE Int. Conf. on Robotics and Automation, Sacramento, CA, 1991, pages 1405–1411, 1991.

**[39] X. Xiong and A. D. Ames,** "Bipedal hopping: Reduced-order model embedding via optimization-based control," In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3821–3828. IEEE, 2018.

**[40] D. E. Orin, A. Goswami, and S.-H. Lee,** "Centroidal dynamics of a humanoid robot," Autonomous robots, 35(2-3):161–176, 2013.

**[41] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter,** "Feedback mpc for torque-controlled legged robots," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 4730– 4737.

**[42] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini,** " Motion

planning for quadrupedal locomotion: coupled planning, terrain mapping and whole-body control," IEEE Transactions on Robotics, 2020.

**[43] S. Fahmi, C. Mastalli, M. Focchi, and C. Semini,** "Passive whole-body control for quadruped robots: Experimental validation over challenging terrain," IEEE Robotics and Automation Letters, vol. 4, no. 3, pp. 2553– 2560, 2019.

**[44] M. Focchi, R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell, and C. Semini,** "Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality," Cham: Springer International Publishing, 2020, pp. 165–209.

**[45] M. D. Shuster,** "A survey of attitude representations," Navigation, vol. 8, no. 9, pp. 439–517, 1993.

**[46] B. Siciliano and O. Khatib,** Springer handbook of robotics. Springer, 2016.

**[47] S. P. Bhat and D. S. Bernstein,** "A topological obstruction to global asymptotic stabilization of rotational motion and the unwinding phenomenon," in American Control Conference, 1998. Proceedings of the 1998, vol. 5. IEEE, 1998, pp. 2785–2789.

**[48] F. Bullo and A. D. Lewis,** "Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems," Springer Science & Business Media, 2004, vol. 49.

**[49] Y. Ding, A. Pandala, C. Li, Y. -H. Shin and H. -W. Park,** "Representation-Free Model Predictive Control for Dynamic Motions in Quadrupeds," in IEEE Transactions on Robotics, vol. 37, no. 4, pp. 1154-1171, Aug. 2021, doi: 10.1109/TRO.2020.3046415.

**[50] G. Wu and K. Sreenath,** "Variation-based linearization of nonlinear systems evolving on SO(3) and S2,," IEEE Access, vol. 3, pp. 1592– 1604, 2015.

**[51] T. Lee, M. Leok, and N. H. McClamroch,** "Stable manifolds of saddle equilibria for pendulum dynamics on S2 and SO(3)," in Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC). IEEE, 2011, pp. 3915–3921.

**[52] H. G. Bock and K.-J. Plitt,** "A multiple shooting algorithm for direct solution of optimal control problems," IFAC Proceedings Volumes, 17(2):1603–1608, 1984.

**[53] O. von Stryk,** "Numerical Solution of Optimal Control Problems by Direct Collocation," pages 129–143. Birkhäuser Basel, Basel, 1993.

**[54] J. E. Marsden and T. S. Ratiu,** "Introduction to mechanics and symmetry," Physics Today, 48(12):65, 1995.

**[55] A. Graham,** "Kronecker products and matrix calculus with applications," Courier Dover Publications, 2018.

**[56] T. A. Davis,** "Algorithm 849: A concise sparse cholesky factorization package," ACM Trans. Math. Softw., vol. 31, no. 4, pp. 587–591, 2005.

**[56] J. C. Trinkle, J-S Pang, S. Sudarsky, and G. Lo,** "On dynamic multi-rigid-body contact problems with coulomb friction," ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik, 77(4):267–279, 1997.

**[57] Y. Wang and S. Boyd,** "Fast model predictive control using online optimization," IEEE Transactions on Control Systems Technology, 18(2):267–278, March 2010.

**[58] S. Boyd and L. Vandenberghe,** "Convex optimization," Cambridge university press, 2004.

**[59] A. G. Pandala, Y. Ding and H. -W. Park,** "qpSWIFT: A Real-Time Sparse Quadratic Program Solver for Robotic Applications," in IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 3355-3362, Oct. 2019, doi: 10.1109/LRA.2019.2926664.

**[60] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing and S. Kim,** "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018, pp. 2245-2252, doi: 10.1109/IROS.2018.8593885.

**[61] J. Pratt, J. Carff, S. Drakunov, and A. Goswami,** "Capture point: A step toward humanoid push recovery," in Humanoid Robots, 2006 6th IEEE-RAS International Conference on. IEEE, 2006, pp. 200–207.

**[62] H.-W. Park, P. M. Wensing, and S. Kim,** "High-speed bounding with the MIT cheetah 2: Control design and experiments," The International Journal of Robotics Research, vol. 36, no. 2, pp. 167–192, 2017.

# 9 List of abbreviations

| | |
|---|---|
| SLIP | Spring-Loaded Inverted Pendulum |
| CPG | Central Pattern Generator |
| MPC | Model Predictive Control |
| PD | Proportional Derivative (Control) |
| TUM | Technical University Munich |
| DOF | Degrees of Freedom |
| COM | Center of Mass |
| RF-MPC | Representation-Free Model Predictive Control |
| ZMP | Zero Moment Point |
| LF | Left Forelimb |
| LH | Left Hindlimb |
| RF | Right Forelimb |
| RH | Right Hindlimb |
| LIPM | Linear Inverted Pendulum |
| QP | Quadratic Program |
| GRF | Ground Reaction Force |
| OP | Operating Point |
| NLP | Nonlinear Programming |
| SRB | Single Rigid Body |
| KKT | Karush-Kuhn-Tucker |
| FSM | Finite State Machine |
| VMC | Virtual Machines Control |

# Appendix

# A1 Appendix 1

This section introduces reference gain values used in RF-MPC.

| | Walk trot | Bound | Pacing | Gallop | Run trot | Crawl |
|---|---|---|---|---|---|---|
| $Q_{P_x}$ | 1e5 | 2e5 | 5e3 | 3e3 | 1e5 | 5e5 |
| $Q_{P_y}$ | 2e5 | 5e4 | 5e3 | 3e3 | 1.5e5 | 5e5 |
| $Q_{P_z}$ | 3e5 | 5e6 | 9e4 | 4e6 | 2e4 | 9e5 |
| $Q_{\dot{P}_x}$ | 5e2 | 8e3 | 5e2 | 5e2 | 1.5e3 | 5 |
| $Q_{\dot{P}_y}$ | 1e3 | 5e2 | 5e2 | 1e3 | 1e3 | 5 |
| $Q_{\dot{P}_z}$ | 1e3 | 5e2 | 5e2 | 150 | 100 | 5 |
| $Q_{R_x}$ | 1e3 | 1e4 | 7e3 | 1e4 | 2e3 | 3e3 |
| $Q_{R_y}$ | 1e4 | 5e4 | 7e3 | 1e4 | 2e3 | 3e3 |
| $Q_{R_z}$ | 800 | 5e3 | 7e3 | 800 | 800 | 3e3 |
| $Q_{\omega_x}$ | 40 | 1e2 | 5e1 | 1e2 | 100 | 3 |
| $Q_{\omega_y}$ | 40 | 1e2 | 5e1 | 5e1 | 60 | 3 |
| $Q_{\omega_z}$ | 10 | 1e2 | 5e1 | 5e1 | 10 | 3 |
| $R_{u_x}$ | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| $R_{u_y}$ | 0.2 | 0.1 | 0.2 | 0.2 | 0.18 | 0.2 |
| $R_{u_z}$ | 0.1 | 0.1 | 0.1 | 0.1 | 0.08 | 0.1 |
| $T_{st}$ | 0.3 | 0.1 | 0.12 | 0.08 | 0.12 | 0.3 |
| $T_{sw}$ | 0.15 | 0.18 | 0.12 | 0.2 | 0.2 | 0.1 |
| $N_{hor}$ | 6 | 7 | 6 | 6 | 6 | 6 |
| $\gamma$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $T_{pred}$ | 0.008 | 0.002 | 0.004 | 0.004 | 0.003 | 0.004 |
| $f_{MPC}$ | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 |

Note: $T_{st}$, $T_{sw}$ and $T_{pred}$ have the unit of [s]; $N_{hor}$ is the MPC prediction horizon; $T_{pred}$ is the prediction time step; $f_{MPC}$ is the MPC control frequency with the unit of [Hz]. All values are just reference values, and specific debugging is required.

# A2 Appendix 2

This section introduces the full foot Jacobian matrix of 3-DOF two-link leg.

$$\boldsymbol{J_{p,f}} = \boldsymbol{J_{v,f}} = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} \quad (A2-1)$$

Among them,

$$J_{11} = 0 \quad (A2-2)$$

$$J_{12} = -l_2(cq_2sq_3 + sq_2cq_3) - l_1sq_2 \quad (A2-3)$$

$$J_{13} = -l_2(cq_2sq_3 + sq_2cq_3) \quad (A2-4)$$

$$J_{21} = l_1cq_1sq_2 + l_2(cq_1cq_2sq_3 + cq_1sq_2cq_3) - sign_d \cdot d \cdot sq_1 \quad (A2-5)$$

$$J_{22} = l_1sq_1cq_2 - l_2(sq_1sq_2sq_3 - sq_1cq_2cq_3) \quad (A2-6)$$

$$J_{23} = -l_2(sq_1sq_2sq_3 - sq_1cq_2cq_3) \quad (A2-7)$$

$$J_{31} = l_1sq_1sq_2 + l_2(sq_1cq_2sq_3 + sq_1sq_2cq_3) - sign_d \cdot d \cdot cq_1 \quad (A2-8)$$

$$J_{32} = -l_1cq_1cq_2 - l_2(cq_1cq_2cq_3 - cq_1sq_2sq_3) \quad (A2-9)$$

$$J_{33} = -l_2(cq_1cq_2cq_3 - cq_1sq_2sq_3) \quad (A2-10)$$

which

$$c = \cos(\cdot)\,;\ s = \sin(\cdot) \quad (A2-11)$$