ROBUST LOCOMOTION CONTROL FOR A QUADRUPED ROBOT WITH HIGH PAYLOAD CAPACITY

A Thesis

by

Erim Can Özçınar

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Mechanical Engineering

Özyeğin University
December 2022

ROBUST LOCOMOTION CONTROL FOR A QUADRUPED ROBOT WITH HIGH PAYLOAD CAPACITY

Approved by:

Asst. Prof. Barkan Uğurlu, Advisor
Dept. of Mechanical Eng.
Özyeğin University

Asst. Prof. Özkan Bebek
Dept. of Mechanical Eng.
Özyeğin University

Prof. Dr. Duygun Erol Barkana
Dept. of Electrical and Electronics Eng.
Yeditepe University

Date Approved: 27 December 2022

To my family

ABSTRACT

This thesis presents a robust locomotion control framework for a quadruped robot. The locomotion control framework consisted of a trajectory planer and feedback controller based on CTC. Trajectory planning algorithm based on the ZMP concept to generate dynamically balanced locomotion and CTC method was implemented to the robot to follow generated trajectories. To implement the CTC, the inverse dynamic model of the robot is constructed through a recursive Newton-Euler method. Additionally, capture point method was implemented for the robot to preserve its balance against external perturbation. This method computes the capture point and generates a recovery trajectory for the robot to step in order to preserve its balance. Furthermore, novel algorithm is proposed to distribute contact forces for the purpose of eliminating undesired torso orientation fluctuations and regulate heading during dynamic quadruped trot-walking motion. The proposed method makes use of centroidal momentum, an important physical quantity in characterizing the whole-body behavior and overall balance of the robot. The contact forces are computed by means of centroidal momentum feedback and injected to the locomotion controller via Virtual Model Control (VMC) which can render virtual forces to regulate robot-environment interaction. The proposed controllers were validated via walking experiments in both a realistic simulation environment and by using an actual quadruped robot. Moreover, endurance and the payload capacity of the robot were validated through endurance test and walking experiment with 20 kg payload.

ÖZETÇE

Bu tez, dört ayaklı bir robot için sağlam lokomosyon kontrol algoritması sunmaktadır. Lokomosyon kontrol algoritması, yörünge planlayıcısından ve Hesaplamalı Tork Kontrolcüsüne (HTK) dayalı geri besleme denetleyicisinden oluşmaktadır. Dinamik olarak dengeli lokomoson oluşturmak için ZMP konseptine dayalı yörünge planlama algoritması ve bacakların oluşturulan yörüngeleri takip edebilmeleri için robot üzerinde HTK yöntemi uygulanmıştır. HTK'yı uygulamak için, robotun ters dinamik modeli özyinelemeli Newton-Euler yöntemiyle oluşturulmuştur. Ayrıca robotun dış pertürbasyonlara karşı dengesini koruması için yakalama noktası yöntemi uygulanmıştır. Bu yöntem, yakalama noktasını hesaplar ve robotun adım atarak dengesini korumak için bir kurtarma yörüngesi oluşturur. Buna ek olarak, istenmeyen gövde oryantasyon dalgalanmalarını ortadan kaldırmak ve dinamik dört ayaklı tırıs yürüme hareketi sırasında istikameti düzenlemek amacıyla yer tepki kuvvetlerini dağıtmak için yeni bir algoritma sunulmuştur. Önerilen yöntem, robotun tüm vücut davranışını ve genel dengesini karakterize etmede önemli bir fiziksel nicelik olan merkezsel momentumu kullanır. Temas kuvvetleri, merkezsel momentum geri bildirimi aracılığıyla hesaplanır ve robot-çevre etkileşimini düzenlemek için sanal kuvvetler sağlayabilen Sanal Model Kontrolü (SMK) yoluyla lokomosyon kontrolcüsüne beslenir. Önerilen kontrolcüler hem gerçekçi bir simülasyon ortamında hem de gerçek bir dört ayaklı robot kullanılarak yürüme deneyleri yoluyla doğrulanmıştır. Ayrıca, robotun dayanıklılığı ve faydalı yük taşıma kapasitesi, dayanıklılık testi ve 20 kg faydalı yük ile yürüme deneyi yapılarak doğrulanmıştır.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my thesis advisor Asst. Prof. Barkan Uğurlu for his guidance and support throughout my M.Sc. study. Also, I would like to thank my professor Asst. Prof. Özkan Bebek, Asst. Prof. Polat Şendur and Asst. Prof. Ramazan Ünal for their support during this project.

I wish to thank my fellow co-workers in this project; Barış Balcı, Sinan Emre and Barış Akın. Also, I want to thank my labmates for their support; Eyüp Turan Karasu, Deniz Yılmaz, Burak Özkaynak, Sinan Coruk, Mustafa Derman, Barış Baysal, Ege Gediksiz, Omid Khosrowshahi, Ahmed Fahmy Soliman, Alihan Kuru, Dilay Yeşildağ.

Finally, I would like to thank my family, and my friends for their consistent support during my M.Sc. study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Abbreviations

COM	Center of Mass
AA (A/A)	Abduction-Adduction
FE (F/E)	Flexion-Extension
DOF	Degrees of Freedom
DC	Direct Current
CAD	Computer Aided Design
ZMP	Zero Moment Point
PD	Proportional-Derivative
ROS	Robot Operating System
IMU	Inertial Measurment Unit
SSI	Synchronous Serial Interface
TCP	Transmission Control Protocol
IP	Internet Protocol
RMS	Root Mean Square
RMSE	Root Mean Square Error

CHAPTER I

INTRODUCTION

1.1 Literature Review

Until a few years ago, wheeled robots dominated the robotic systems. Wheeled robots are advantageous due to their simplistic mechanisms and lightweight, and these robots are highly effective on smooth surfaces. However, irregular terrains may be inaccessible for wheeled robots. Even wheeled robots can surpass some small obstacles and surface unevenness, they could not be efficient for a general class of uneven terrains [1]. On the contrary, legged robots may be more suitable on uneven terrains compared to wheeled robots [2]. The most distinct advantage of the legged robots is the ability to overcome large obstacles by lifting their legs [3]. Thus, nowadays, legged robots are becoming more prevalent and started to draw much attention.

The most extensive types of legged robots are biped or quadruped robots. Quadruped robots are four-legged robots that are advanced in dynamic locomotion abilities [4]. These robots are designed via the inspiration of quadrupedal animals, which are dynamically versatile over challenging terrains [5].

There are several commercialized and product-level quadruped robots. For instance, the most known example is the Boston Dynamics' Spot robot which descended from Boston Dynamics' early success BigDog [6]. Spot allows operators to automate routine inspection tasks and data capture safely [7]. Moreover, various research groups also achieved locomotion performance similar to the Spot robot. Istituto Italiano di Tecnologia (IIT) developed a hydraulically powered quadruped robot named HyQ [8]. MIT developed a quadruped robot Mini Cheetah that is capable of displaying agile motions, e.g., back flip [9]. ETH developed a compliant quadruped named StarlETH to study versatile locomotion [10]. All these robots

achieved dynamic locomotion via various control algorithms.

The overall locomotion framework of quadruped robots varies according to the desired task. The overall locomotion controller can be divided into two sections,

- Trajectory planning and pattern generation.
- Whole body control with real-time feedback.

In the literature, different types of trajectory planners are tested on quadruped robots. For instance, Central Pattern Generator (CPG) is a neural network to generate complex control signals for rhythmic movements inspired by animals [11]. Moro et al. made use of walking, trotting, and galloping data obtained from horses via motion capture cameras and proposed a method to transfer these data to quadruped locomotion [12]. Another pattern generation method for legged robots is a ZMP-based trajectory generation. ZMP concept enables the generation of COM trajectory while theoretically preserving dynamic balance [13]. ZMP-based pattern generation is achieved by moving COM in a way that keeps ZMP inside support polygon [14]. There are several legged robots in the literature that uses the ZMP concept to generate dynamically balanced COM trajectory [15, 16, 17].

In addition to trajectory planning and pattern generation, control algorithms for tracking the generated patterns are necessary for the execution of tasks. Barasuol et al. proposed a reactive controller for robust quadrupedal locomotion to handle tracking errors and terrain irregularities [18]. Barasuol's reactive controller framework consists of two main blocks, which are motion generation and motion control. In motion generation, they generate feet trajectories based on CPG, which are converted into joint trajectories via inverse kinematics. Then, desired joint trajectories were tracked by PD position, torque controllers, and feed-forward inverse dynamics. Moreover, the locomotion framework of the HyQ robot for uneven terrain is proposed by Mastalli et al. Locomotion control framework of HyQ consists of three modules which are motion planning, whole-body control, and mapping

and estimation [19]. The mapping and estimation module handles sensor feedback and terrain mapping. In the motion planning module, COM trajectory and footholds satisfy robot stability and deal with terrain conditions computed by the horizon optimization. In addition to the horizon optimization, the attitude planner adapts torso orientation. In order to achieve compliant tracking, they designed a whole-body controller composed of a virtual model, a joint impedance controller, and a whole-body optimization.

The different example, Kim et al. proposed a locomotion control framework that combines whole-body impulse control (WBIC) and Model Predictive Control (MPC) for MIT's mini-cheetah [20]. The main idea of this framework is to reduce complexity. MPC computes optimal reaction force distribution. Then, computed reaction forces and robot states that are calculated from the dynamic model are fed to the WBIC which computes desired joint states and required joint torques. Another example of locomotion control is proposed by Kalakrishnan et al. [21]. For robust and fast locomotion, they applied state-of-the-art learning, planning, optimization, and control techniques. They used a template learning algorithm to compute optimal foot placements. Then, they planned a body path roughly which guides the robot through suitable foothold regions. According to the planned body path and terrain rewards the footstep planner plans the following four footsteps. Pose finder optimizes the robot's pose to avoid collisions and maximize kinematic reachability. For the next four steps, smooth body trajectory and swing leg trajectories are generated. To track planned trajectories, they implemented a PD controller with inverse dynamics and contact force control. They used a joint PD controller to track joint trajectories, and inverse dynamics torques were fed forward to provide better tracking. They added a contact force controller via desired contact forces obtained from inverse dynamics. Locomotion control proposed by Kalakrishnan et al. achieves locomotion over rough terrain [21].

1.2 Contributions

In the literature, various types of robots with various types of locomotion controllers are presented. Trajectory planning and control algorithms for tracking those trajectories are essential for achieving dynamically balanced locomotion.

In this thesis, the locomotion control problem of the quadruped robot that was designed and assembled in the Biomechatronics laboratory of Ozyegin University. In order to achieve dynamically balanced locomotion, ZMP based trajectory planner is designed [14]. Moreover, a joint space position controller was implemented to track generated trajectories. Furthermore, a computed torque controller was implemented to improve tracking. The inverse dynamic of the robot is obtained through a novel Newton-Euler algorithm. Furthermore, the capture point method is used to achieve dynamic balancing for external forces. The contributions of this thesis are as follows:

i The development of state-of the art quadruped that can dynamically carry payloads up to 20 kg.

ii Modification of the trajectory generator proposed in [14] such that the robot could be commanded with a wireless terminal.

iii Modifications at the Newton-Euler algorithm such that it is computed inexpensively, approximately within 8 ms.

iv A novel algorithm to achieve momentum-aware contact force distribution that could be implemented in real-time.

The locomotion control framework is supported by both simulation and experimental studies. Additionally, experimental studies of locomotion with a high payload and endurance test with a 5.5 kg payload are presented. Apart from that, contact force distribution via a combination of centroidal momentum feedback and the virtual model controller is presented. Force distribution study supported with simulation.

CHAPTER II

HARDWARE DEVELOPMENT

2.1 Brief Overview on Mechanical Design

In this section, the mechanical design of the quadruped robot, which is developed at the Biomechatronics Laboratory of Ozyegin University, is briefly introduced. Hardware development of the robot can be divided into three parts. These parts are the actuator unit design, leg unit design and torso design. Arguably, the most essential component of the quadruped robot is the actuator unit. There are three DOFs on each leg for feasible walking on 3-D thus twelve actuators were used in total. The primary goal of the actuator design is to develop a lightweight actuator unit with high torque output. This unit consist of three main elements. These elements are listed as fallows,

- Frameless Brushless Motor (Kollmorgen TBM-7631)
- Reduction Gear (Harmonic Drive CSG-25, 1:30)
- High Resolution (18-bit) encoders (FENAC FNC AS40HBK3)

Emre et al. proposed a compact and lightweight design for actuator unit. The CAD design of this actuator can be seen in Fig.1. The motor was placed in a costume built frame. The output hollow shaft of the motor was directly connected to the harmonic bearing of the gearbox module. The high resolution encoder is tightly placed at the backside of the motor shaft to measure motor the angular position of the motor shaft. In order to develop a lightweight actuator unit, a composite material based on a onyx-carbon fiber combination is used for the main frame of the actuator unit. Al-7075 was selected for motor shaft and output shaft of the actuator unit to preserve rigidity.

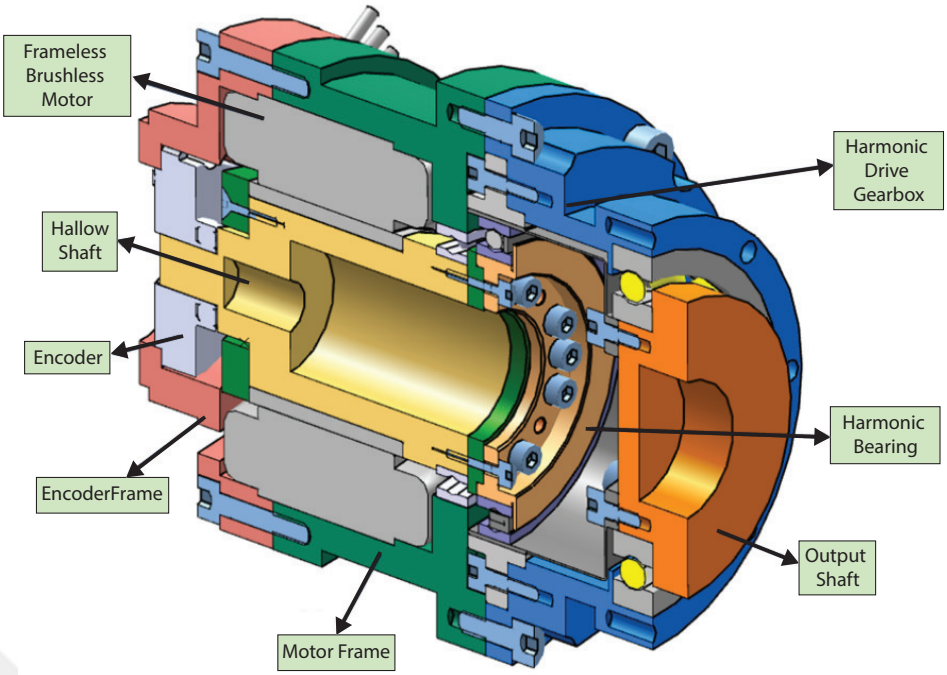

Figure 1: CAD design of actuator unit.

For the leg unit design, we aimed to minimize the leg inertia. Thus, actuator units were place as close as possible to torso. This design strategy requires a mechanism to actuate knee joint as the knee motor is bedded close to the hip. Emre et al. conducted dynamic simulations for 3 different mechanisms. As a result of these simulations, a four-bar mechanism found to be the most suitable in terms of torque generation and range of motion. A composite material based on a onyx-carbon fiber combination is also used in leg unit design. Final design of the composite leg unit of the robot and the one four-bar mechanism is shown in Fig. 2.

All the electronic hardware and HIP A/A joint actuators are located at the torso of the robot. Therefore, majority of the weight of the system is gathered around torso. In order to achieve lightweight design of torso, carbon fiber tubes were used in skeletal structure and wall of this skeletal structure were manufactured from ABS plastic material. The model analysis and static structural analyses of the walls and torso were conducted. and results showed that Von-Mises stresses are below the yield stresses [22].

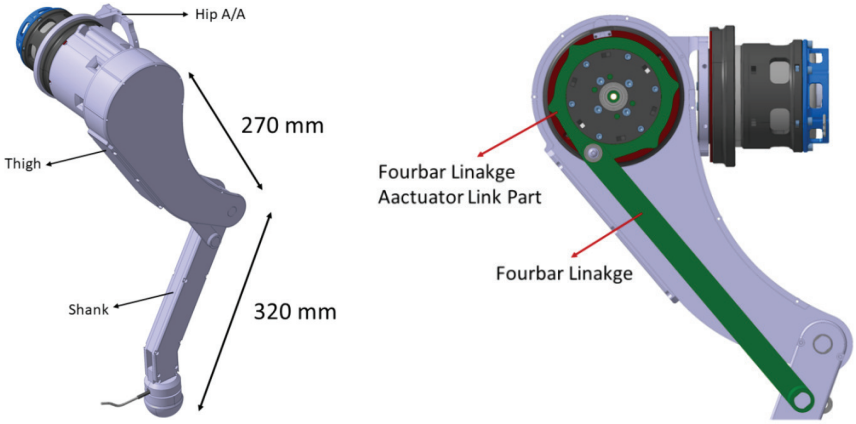

Figure 2: CAD design of composite leg unit and four-bar mechanism.

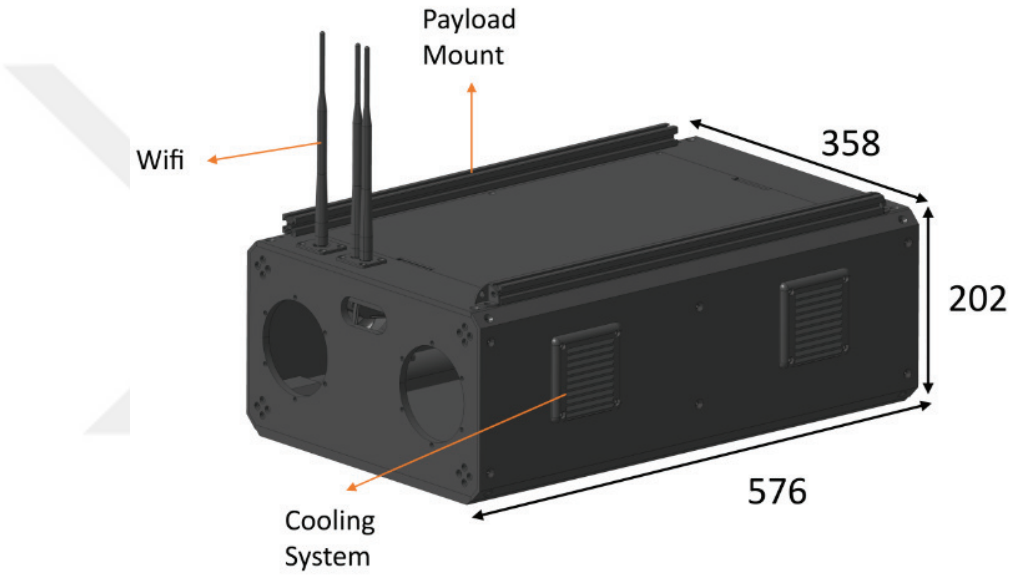

Figure 3: CAD design of torso.

2.2 Electronics Design and Communication

While determining the electronic systems of the robot, both the power requirements of the electromechanical components of the robot and the requirements of the robot's control system were taken into account. In Fig. 4, the essential electronic components of the robot and the connections between these components are shown. In this figure, connections indicating voltage value represent power system connections, non-valued connections represent motor-driver-specific connections, and controller systems connections. The connections of the motor driver are given in Fig. 5 on the drawing taken from the data sheet of the motor drivers.

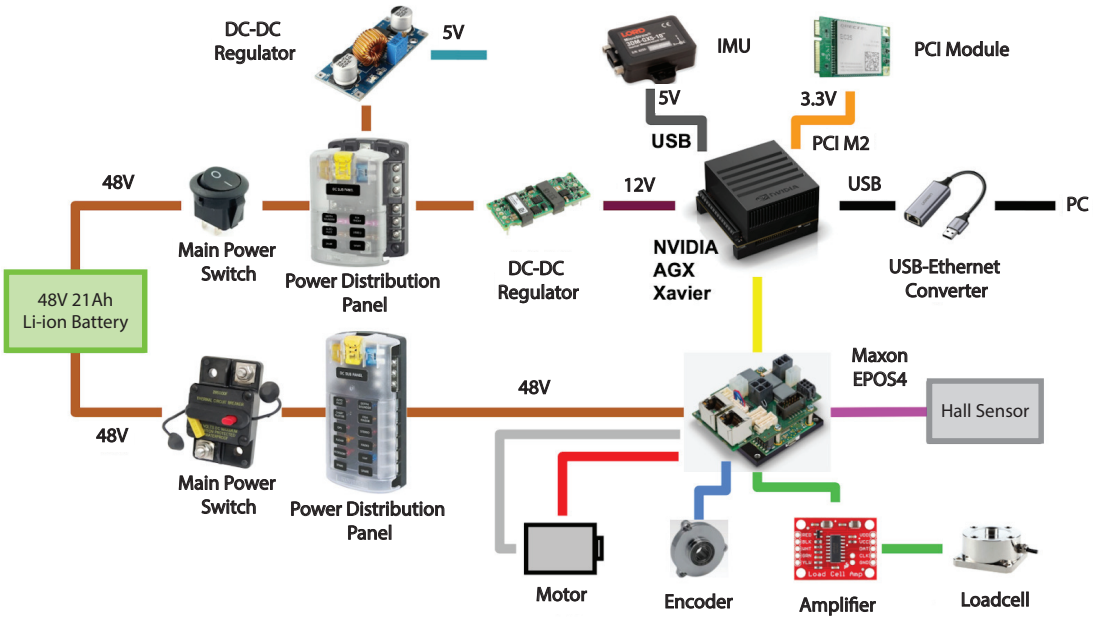

Figure 4: Electronic system scheme.

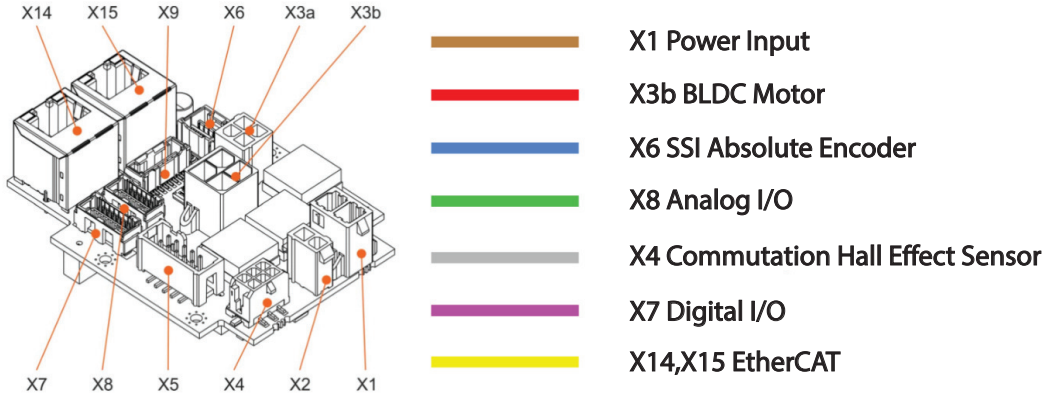

Figure 5: Connections between motor drives and control system equipment.

The electronic hardware of the robot consists of three major parts; motor drivers, sensors, and computational unit. For this robot, NVIDIA Jetson AGX Xavier was employed as the main computational unit. Its high performance computational capabilities, energy efficiency, and AI inference capabilities ensure fast and reliable computations for complex calculations in real-time. Besides, it provides the required computational power for future research such as machine learning, computer vision and so on. In order to drive brushless motors, Maxon EPOS4 Compact 50/15 EtherCAT motor drivers were used. The main purpose of the motor driver is to transfer the battery power to the motor via a control system.

Moreover, these motor drivers are capable of incorporating many sensors into the control system without the need for additional equipment. Motor drivers can receive the energy required for their own operations and actuator unit over power distribution panel via a pair of cables. While the nominal operating voltage of the motor drives is 8-50 V, the nominal current value is 15 A. Motor drives can also continue to operate for up to one minute at a peak current of 30 A.

For the precise position control application, high-resolution encoders are required. Thus, FENAC custom-made SSI absolute encoders with 18-bit resolution were utilized. These encoders were mounted on the motor shaft of the actuator. EPOS4 motor drivers can communicate with SSI encoders without requiring any intermediary circuit. Hence, every encoder in the actuator unit is directly connected to the motor driver that controls the actuator unit. While this connection carries energy to the encoder, it also transfers the encoder data to the main control system.

A Hall effect sensor was used in each actuator unit in order to achieve the main configurations of the actuator units in the home positioning sequence. These Hall Effect sensors were mounted on the actuators, and magnets were placed on the joints. Similar to encoders, Hall effect sensors were connected directly to the EPOS4 motor drives. Thus, these sensors receive their power over the motor drivers and perform the data transfers over the motor drives.

In order to keep the torso of the robot in the desired orientation and to implement the planned trajectories with high precision, a Parker Lord 3DM-GX5-25 IMU sensor was mounted on the torso of the robot. This IMU provides triaxial acceleration and angular velocity data as well as torso orientation information. Unlike other sensors, the IMU sensor is connected to the control system by directly connecting it to the host computer via a RS232-USB converter.

The control system is the layer that connects the robot's power system to the program block. Components in both the power and control system, such as motor drives, act as a bridge between these two systems. Simplicity has been

given importance in the control system, so it was aimed to obtain maximum efficiency from these components by using the minimum number of components. This objective was mainly achieved by the use of EtherCAT technology [23].

2.3 Programming Environment

The control system creates the necessary structure for the electronic equipment to communicate with the main robot computer and, in some cases, with each other. However, the control system hardware alone cannot control the robot. Assessing the data coming from the hardware, making decisions based on this data, and sending the necessary commands to the actuator units are the tasks of the primary controllers developed for the robot. Controllers, which take into account many static and dynamic physical properties of the robot, are mathematical functions calculated on the robot with the help of program blocks. Program blocks that make up the software layer and perform the robot's essential functions, such as communication, trajectory planning, and kinematic analysis, are run on the main robot computer.

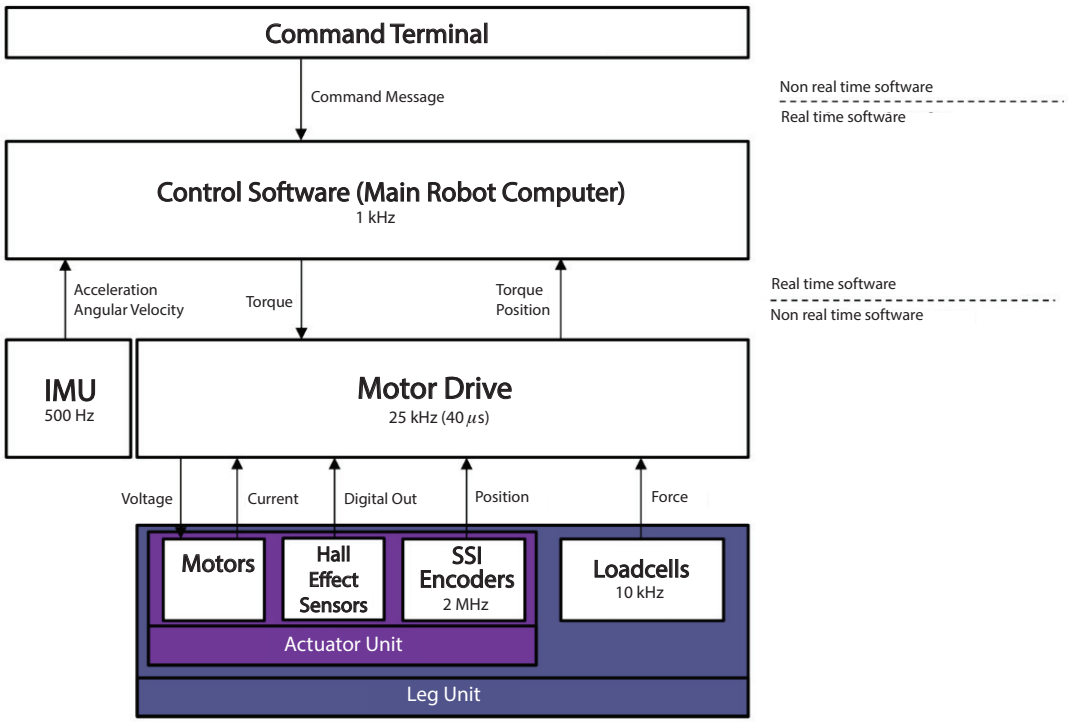

Figure 6: controller system layers block diagram.

The controller system maintains the essential functions of the robot and activates the relevant units of the robot with the command from the command terminal, and brings the robot to the state desired by the user. As shown in Fig. 6, the controller system can be grouped under three main categories.

The commands that the user assigns to the robot are collected via the command terminal. A Tablet computer was used for the command terminal so that user can communicate with the robot via a wireless data link. By using the interface on the tablet computer, the user is able to determine the speed of the robot's center of gravity. Furthermore, the command terminal can select the walking mode, speed, and height of the robot's center of gravity from the ground.

The control software running on the main robot computer is the next stage in the controller system flow from the command terminal. The control software includes many program blocks that perform functions such as balancing the robot and computing trajectories and program blocks that process user commands.

The torque values that will bring the robot to the user's desired state are provided by actuating the actuator units through the control system hardware. The motor drives in the control system hardware act as a bridge between the control software and the actuator units. Motor drives contain current controllers that account for information such as motor constant, rated current, and maximum current of the actuator unit they control. When the torque values are determined by the control software and transmitted over EtherCAT are converted into current values with the help of motor information, computed current values are applied by the motor driver. The torque from the current applied to the actuator units moves the actuator unit. This motion's velocity and position information are measured by the SSI absolute encoders integrated into the actuator unit. The encoders transmit the information needed by the control software primarily to the motor driver and from the motor driver to the host computer via EtherCAT.

Robot software is the sum of the program blocks that enable the robot to perform its functions and the operating system that executes the program blocks

on the host computer. Program blocks, whose controllers in the controller system are implemented, are developed in C/C++. The most important factors in using these languages are that they have a wide range of motion on the computer and that they produce programs that run quickly at application time. Python language is used in blocks such as test blocks, which do not have time constraints.

The robot host computer runs on a version of the Ubuntu 18.04 operating system, a Linux distribution customized by the robot host computer manufacturer NVIDIA for its Jetson product line. This special version includes software that supports computer-specific components such as the graphics processing unit and multi-purpose inputs/outputs on the computer. Although the operating system is presented as quite ready to use, it requires a real-time operation to control the quadruped robot. Real-time operating systems run programs on the computer, taking into account the priority order assigned to them. In this way, low-priority programs cannot intervene during the execution of high-priority programs, thus reducing system delays on high-priority programs. Since the programs running on real-time operating systems are not exposed to system delays, they can perform the operations they need to perform periodically without leaving the specified period, and they are characterized as deterministic. Since robot control software is also a periodic program that must run deterministically, a real-time operating system is required on the robot host computer, but most operating systems, including the Ubuntu version offered by NVIDIA, are not real-time.

Linux-based operating systems can be made in real time by several different methods. Two of the most used of these methods are PREEMPT_RT and Xenomai patches. These patches are applied to the kernel of the operating system and make the operating system deterministic. PREEMPT_RT can be used after following a set of installation steps in Ubuntu, which NVIDIA has customized for the Jetson product line. However, a more detailed installation process is required for Xenomai. Thus, PREEMPT_RT patch was used in the robot's host computer. As a result of the "Cyclictest", which is one of the basic tests of real-time operating

systems, the system latency graphs of the robot's host computer on non-real-time and real-time operating systems (with PREEMPT_RT patch) are shown in Fig. 7.

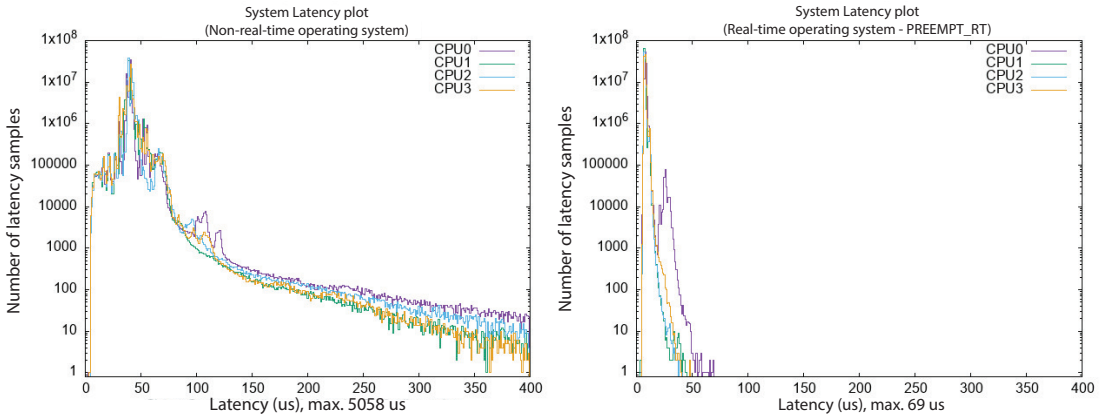

Figure 7: System latency graphs in non-real-time (left) and real-time (right) operating systems.

Although different robots are designed for various purposes with distinct components, they may share a common grounds. Remote communication, simulation, and data transfer between program blocks are some of the common elements. Robot Operating System (ROS) is a software middleware that incorporates these elements. Through this framework, robot developers can get their robots ready for use faster. ROS was also used in our quadruped robots. ROS will accelerate the robot's software development process and bring the robot's software to a standard that is accepted and used in the world. Another advantage of ROS over the robot is that the robot can be easily integrated into applications developed in the ROS ecosystem or to be developed in the future. For example, without the need to develop special software for a LIDAR sensor that will likely be used on the robot in future phases, the data produced by this sensor will enable the robot to move past the obstacles in its environment via software in the ROS ecosystem. The connection between the control terminal and the robot is also easily provided via ROS. The infrastructure of ROS is built on TCP/IP communication protocol. Two different devices on the same network can easily communicate using ROS-specific data packets. In this case, the robot host and the command terminal,

which are connected to the same network, use ROS data packets to communicate between them.

CHAPTER III

LOCOMOTION CONTROL FRAMEWORK

3.1 Simulation Environment

Simulations are conducted in RaiSim v1.1.5. RaiSim is a physics engine designed to simultaneously provide accuracy and speed for simulating robotic systems. It is a generic rigid-body dynamics simulator that solves contact dynamics problems based on an efficient bisection method [24].

3.2 Trajectory Planner

Quadrupedal locomotion has various gait patterns, such as galloping, walking, bounding, and trotting. In this work, the gait pattern was generated in accordance with the trot walking. Quadrupedal trotting was achieved by moving the diagonal feet of the robot simultaneously in the same motion. Thus, quadrupedal trotting consists of three phases.

- 2 legged support phase (Left front and right back)
- 4 legged support phase
- 2 legged support phase (Right front and left back)

Due to the lack of 3 legged support phase, quadrupedal trotting is similar to planar biped walking. This similarity was used to simplify the pattern generation by constructing an equivalent biped model. While constructing the equivalent biped model, the quadruped's diagonal legs are considered as a single biped leg. As shown in Fig. 8, the left front and right back legs are considered as the single left leg of the planar biped model, and the right front and left back legs are considered as the single right leg of planar biped model. This simplification allows us to modify the trotting phases. Two-legged support phases become the single

support phase, and the four-legged support phase becomes the double support phase.

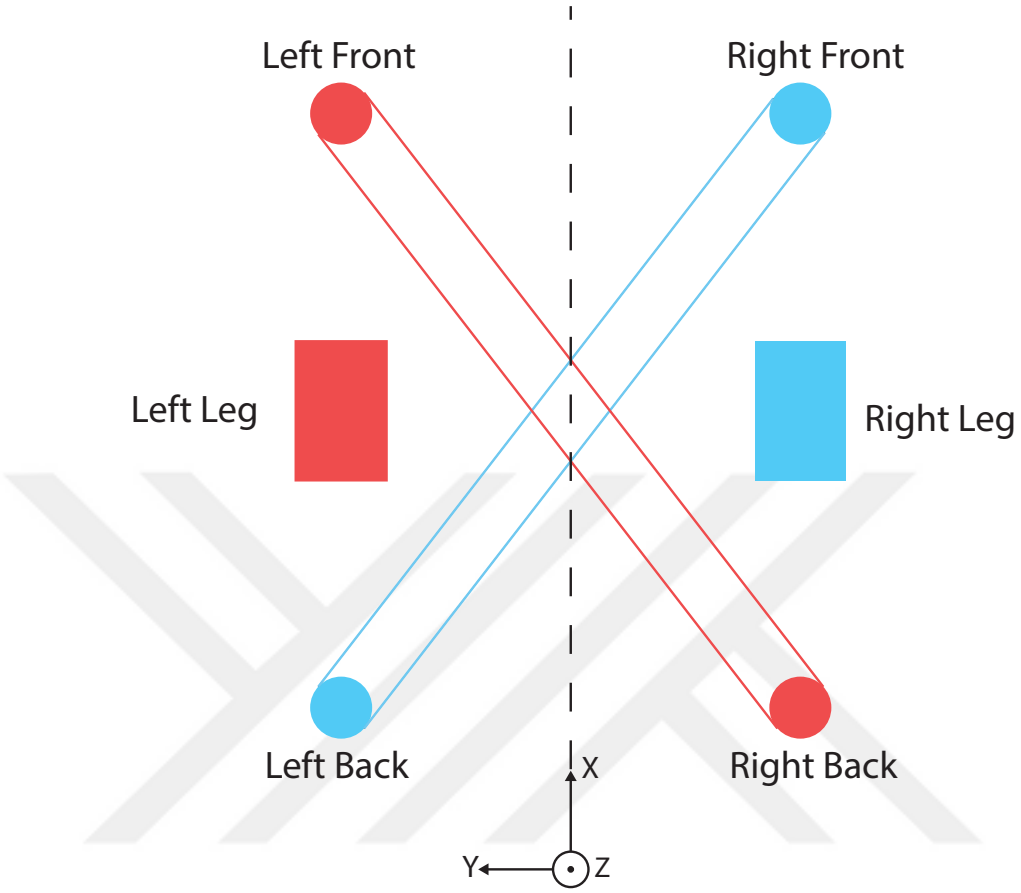

Figure 8: Equivalent planar biped model of quadruped feet during trot-walking.

The trajectory planner of the robot makes use of the ZMP concept for generating a dynamically balanced COM trajectories. The trajectory of the COM was analytically synthesized using the ZMP concept to achieve dynamically balanced locomotion. ZMP is the point on the contact surface where the sum of all the moments in the horizontal direction is zero [13]. ZMP concept works under no slipping conditions. This concept ensures the dynamic balance condition if ZMP is located inside the support polygon. ZMP equations can be driven from Three-Dimensional Linear Inverted Pendulum Mode (3D-LIPM); see Fig. 9.

Using to the 3D-LIPM model, the position of the ZMP in the x-axis direction can be formulated as follows:

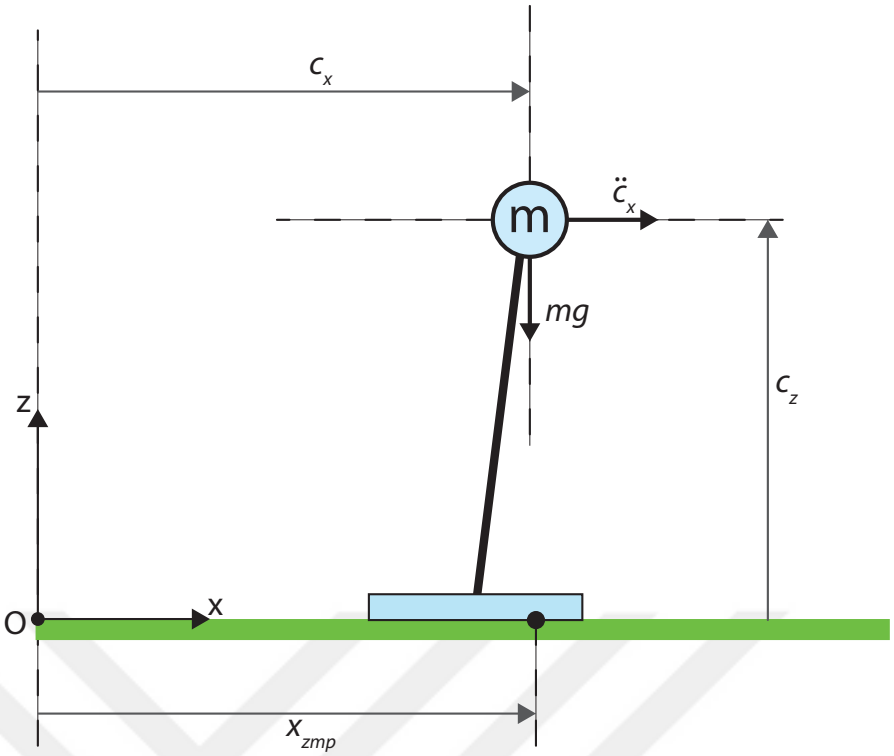

Figure 9: Equivalent planar biped model of quadruped feet during trot-walking.

$$X_{zmp} = c_x - c_z \frac{\ddot{c}_x}{g} \tag{1}$$

$$Y_{zmp} = c_y - c_z \frac{\ddot{c}_y}{g} \tag{2}$$

In this equation, X_{zmp} indicates the position of the ZMP in the x-axis with respect to the world frame, c_x and c_z represents the COM position in horizontal and vertical directions with respect to world frame, g is the gravitational acceleration. Depending on the X_{zmp} input, (1) can be analytically solved. For the COM trajectory, X_{zmp} input is chosen differently for single support and double support phases. For the single support phase, X_{zmp} input is chosen as constant value $X_{zmp} = p_x$. For double support phase, X_{zmp} input is chosen as a ramp input $X_{zmp} = p_x + K_x t$ where, p_x and K_x are positive constants.

For the single support phases, X_{zmp} is chosen as constant $X_{zmp} = p_x$ and (1) is analytically solved for c_x

$$c_x = (c_{x0} - p_x)\cosh(wt) + \frac{\dot{c}_{x0}}{w}\sinh(wt) + p_x \tag{3}$$

Where w is the natural frequency of the equivalent pendulum ($w = \sqrt{\frac{g}{c_z}}$), t is time variable, c_{x0} and $\dot{c}_{x0}$ represent initial COM position and COM velocity along x-axis respectively. The COM velocity and acceleration functions are derived by differentiation of the COM position function as shown in (4) and (5).

$$\dot{c}_x = (c_{x0} - p_x)w\sinh(wt) + \dot{c}_{x0}\cosh(wt) \tag{4}$$

$$\ddot{c}_x = (c_{x0} - p_x)w^2\cosh(wt) + \dot{c}_{x0}w\sinh(wt) \tag{5}$$

The initial COM velocity is obtained from the COM acceleration function. In the middle of the single support phase, which is $t = \frac{T_s}{2}$, COM acceleration is zero. Therefore, $\dot{c}_{x0}$ can be obtained via $\ddot{c}_x|_{t=\frac{T_s}{2}} = 0$.

$$\dot{c}_{x0} = w(p_x - c_{x0})\coth(\frac{wT_s}{2}) \tag{6}$$

The terminal position of the single support phase can be computed as shown in (7) by substituting a mean velocity (v_{mean}) for the single support phase.

$$c_{xd} = c_{x0} + v_{mean}T_s \tag{7}$$

Moreover, terminal position can be calculated via (3) when $t = T_s$ as follows:

$$c_x|_{t=T_s} = c_{xd} = p_x + (c_{x0} - p_x)\cosh(wT_s) + \frac{\dot{c}_{x0}}{w}\sinh(wT_s) \tag{8}$$

The combination of (6) and (8), c_{xd} is obtained:

$$c_{xd} = 2p_x - c_{x0} \tag{9}$$

Then plugging (7) into (9), c_{x0} is computed as:

$$c_{x0} = p_x - \frac{v_{mean}T_s}{2} \tag{10}$$

Therefore, initial conditions of the single support phase are calculated by using (6), (9), and (10) according to the selected single support phase duration T_s, target mean velocity v_{mean} and X_{zmp} input p_x. Then, these initial conditions are plugged in (3), and the COM trajectory for the single support phase is computed.

As mentioned before, quadrupedal trot walking consists of repeated single-support and double-support phases. Hence, the single support and double support phase are linked to each other. The double support phase terminal position velocity and acceleration should be equal to the initial conditions of the following single support phase. Moreover, the essential part of the double support phase is to move the X_{zmp} input from the current support feet to the next support feet. Therefore, (1) is solved for linearly increasing X_{zmp} input; $X_{zmp} = p_x + K_x t$. The COM position, velocity, and acceleration for the double support phase are computed as follows:

$$c_x = (c_{xd} - p_x)\cosh(wt) + \frac{\dot{c}_{x0} - K_x}{w}\sinh(wt) + p_x + K_x t \tag{11}$$

$$\dot{c}_x = (c_{xd} - p_x)w\sinh(wt) + (\dot{c}_{x0} - K_x)\cosh(wt) + K_x \tag{12}$$

$$\ddot{c}_x = (c_{xd} - p_x)w^2\cosh(wt) + (\dot{c}_{x0} - K_x)w\sinh(wt) \tag{13}$$

In (11), (12), and (13), the initial position of the double support phase is equal to the terminal position of the single support phase, and initial velocity of the double support phase is equal to the initial velocity of the single support phase. K_x represents the inclination value of the X_{zmp} input. For continuous connection between double support and single support phases, the terminal velocity of the double support phase should be equal to the initial velocity of the next single support phase.

$$\dot{c}_x|_{t=T_d} = \dot{c}_{x0} = (c_{xd} - p_x)w\sinh(wT_d) + (\dot{c}_{x0} - K_x)\cosh(wT_d) + K_x \tag{14}$$

K_x and stride length (Str) can be computed by using (14).

$$K_x = \dot{c}_{x0} + (c_{xd} - p_x)w\coth(\frac{wT_d}{2}) \tag{15}$$

$$Str = 2T_dK_x \tag{16}$$

The COM trajectory for the double support phase is generated by using (11). Finally, the swing leg trajectories are generated by using polynomial functions via stride length Str and maximum foot clearance F_c. Foot clearance is selected freely. For swing leg trajectories, initial and terminal velocity and acceleration of the foot are assigned to zero in order to reduce the impact force. For the swing leg x-axis trajectory, 5^{th} order polynomial function was used. For the swing leg z-axis trajectory, 6^{th} order polynomial function was used. [14], [25]

3.3 Robot Kinematics

The corresponding joint angles of generated trajectories are computed by the inverse kinematics of the robot. The kinematic model of the robot can be divided into two parts which are;

- Floating base kinematics.
- Leg kinematics.

The leg kinematic is constructed from the hip A/A joint to the foot. Thus, leg kinematics computes joint angles for given foot positions with respect to the hip A/A joint. In the floating base kinematics, the main aim is to compute each foot's position with respect to the hip A/A joint.

The floating base kinematics of the robot is driven by vectorial calculation. By examining Fig.10 and applying vectorial computation, the following equality is driven:

$$r_f = r_c + R_t(r_{ch} + r_{hf}) \tag{17}$$

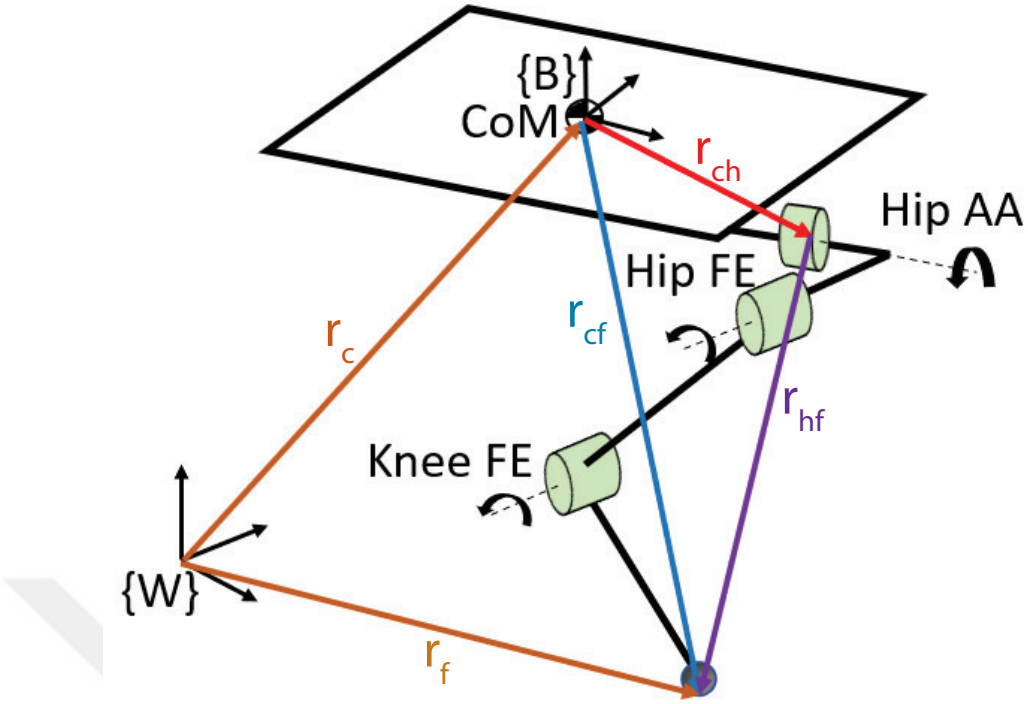

Figure 10: Kinematic scheme of robot.

In (17), r_c and r_f are the COM and foot position vector with respect to world frame (W), respectively. r_{hf} and r_{cf} represents position vectors of the foot with respect to hip A/A joint and COM, respectively. r_{ch} is the hip A/A joint position vector with respect to COM, and also, R_t is the torso orientation which is constructed by using Euler angles. By rewriting (17), foot position vector with respect to hip A/A joint is obtained as follows:

$$r_{hf} = R_t^T (r_f - r_c) - r_{ch} \tag{18}$$

After r_{hf} is driven, the joint angles are calculated by using analytically solved leg inverse kinematics. In 18, R_t, r_f and r_c are obtained from generated trajectories.

The kinematic scheme of a single leg can be seen in Fig. 11. In this figure, the box with dashed lines represents the torso of the robot. Green circles represent each joint, and black circles are the static connection between links. Analytical inverse kinematics of the leg can be driven from forward kinematics equations.

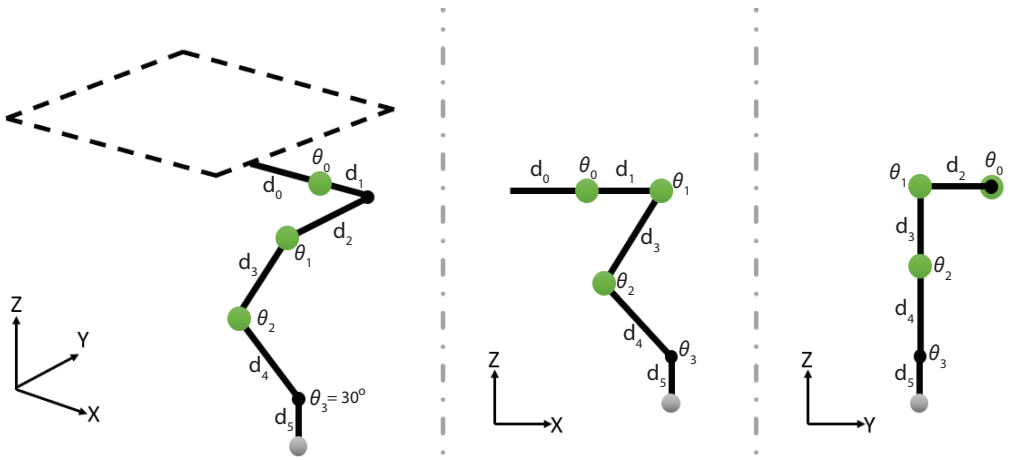

Figure 11: Kinematic scheme of single leg.

Foot positions driven from forward kinematics are obtained as below:

$$P_{fx} - d_0 = d_4 \sin(\theta_1 + \theta_2) + d_3 \sin(\theta_1) + d_5 \sin(\theta_1 + \theta_2 + \theta_3) \tag{19}$$

$$\begin{aligned} P_{fy} &= d_1 \cos(\theta_0) + d_2 \cos(\theta_0) - d_4 \cos(\theta_1 + \theta_2) \sin(\theta_0) \\ &- d_3 \cos(\theta_1) \sin(\theta_0) - d_5 \cos(\theta_1 + \theta_2 + \theta_3) \sin(\theta_0) \end{aligned} \tag{20}$$

$$\begin{aligned} P_{fz} &= d_1 \sin(\theta_0) + d_2 \sin(\theta_0) + d_4 \cos(\theta_1 + \theta_2) \cos(\theta_0) \\ &+ d_3 \cos(\theta_1) \cos(\theta_0) + d_5 \cos(\theta_1 + \theta_2 + \theta_3) \cos(\theta_0) \end{aligned} \tag{21}$$

In (19), (20), (21), and Fig. 11, d_i and θ_i represent the link lengths and joint angles, respectively and also, θ_3 is constant, and it is equal to 30 deg. By summing the square of each side of (19), (20), and (21), the following equation is obtained:

$$\begin{aligned} C_2 &= A_2 \cos(\theta_2) + B_2 \sin(\theta_2) \\ A_2 &= 2d_4 + 2d_5 \cos(\theta_3) \\ B_2 &= -2d_5 \cos(\theta_3) \end{aligned} \tag{22}$$

$$C_2 = \frac{1}{d_3}[(P_{fx} - d_0)^2 + P_{fy}^2 + P_{fz}^2 - (d_1 + d_2)^2 - d_3^2 - d_4^2 - d_5^2 - 2d_4 d_5 \cos(\theta_3)] \tag{23}$$

By using (22), θ_2 is computed. Furthermore, by rewriting (19), the following equality is driven:

$$\begin{aligned} P_{fx} - d_0 &= A_1 \cos(\theta_1) + B_1 \sin(\theta_1) \\ A_1 &= d_4 \sin(\theta_2) + d_5 \sin(\theta_2 + \theta_3) \\ B_2 &= d_3 + d_4 \cos(\theta_2) + d5 \cos(\theta_2 + \theta_3) \end{aligned} \tag{24}$$

Eq. (24) is used to compute θ_1. Moreover, (20) is rewritten as shown below:

$$\begin{aligned} P_{fy} &= A_0 \cos(\theta_0) + B_0 \sin(\theta_0) \\ A_0 &= d_1 + d_2 \\ B_0 &= d_4 \cos(\theta_1 + \theta_2) + d_3 \cos(\theta_1) + d_5 \cos(\theta_1 + \theta_2 + \theta_3) \end{aligned} \tag{25}$$

Eq. (22), (24), and (25) can be solved by using analytical solution of $A\cos(x) + b\sin(x) = c$ equality. θ_2, θ_1, and θ_0 are obtained by analytically solving (22), (24), and (25), respectively.

3.4 Newton-Euler Algorithm

In order to compute the required joint torques for the given trajectories, the recursive Newton-Euler algorithm was used. The Newton-Euler algorithm is an iterative method that considers each link of the robot as a rigid body and recursively applies Newton's and Euler's equations of motion to each link. The recursive Newton-Euler algorithm is composed of two main parts. These parts are the outward iterations and inward iterations.

Outward iterations compute linear and angular accelerations and angular velocity of each link's COM in an iterative way. Starting from link 0, each link's angular velocity and linear and angular acceleration are computed link by link to link n. The angular velocity and acceleration from link to link for joint $i + 1$ are computed as follows:

$$^{i+1}w_{i+1} = {}^{i+1}_{i}R\,{}^{i}w_i + \dot{\theta}_{i+1} \tag{26}$$

$$^{i+1}\dot{w}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{w}_i + {}^{i+1}_{i}R\,{}^{i}w_i \times \dot{\theta}_{i+1} + \ddot{\theta}_{i+1} \tag{27}$$

In (26) and (27), $^{i+1}w_{i+1}$ and $^{i+1}\dot{w}_{i+1}$ are the angular velocity and acceleration of the link, $^{i}w_i$ and $^{i}\dot{w}_i$ are the angular velocity and acceleration of the previous link, respectively. Furthermore, $^{i+1}_{i}R$ represents the rotation matrix from i^{th} frame to $(i+1)^{th}$ frame and $\dot{\theta}_{i+1}$ and $\ddot{\theta}_{i+1}$ are the joint velocity and acceleration of the joint $i+1$ which is the parent of the link $i+1$. The linear acceleration of the link frame (located at the parent joint of the link) $i+1$ is computed as below:

$$^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R[{}^{i}w_i \times {}^{i}P_{i+1} + {}^{i}w_i \times ({}^{i}w_i \times {}^{i}P_{i+1}) + {}^{i}\dot{v}_i] \tag{28}$$

In (28), $^{i+1}\dot{v}_{i+1}$ and $^{i}\dot{v}_i$ represent the linear acceleration of the link-frame we want to obtain and the previous link-frame respectively. $^{i}P_{i+1}$ is the position of the joint $i+1$ with respect to joint i. Moreover, the COM acceleration of each link is required. The COM acceleration of i^{th} link is obtained as shown below:

$$^{i}\dot{v}_{c_i} = {}^{i}\dot{w}_i \times {}^{i}P_{c_i} + {}^{i}w_i \times ({}^{i}w_i \times {}^{i}P_{c_i}) + {}^{i}\dot{v}_i \tag{29}$$

Where, $^{i}\dot{v}_{c_i}$ is the COM acceleration of the i^{th} link and $^{i}P_{c_i}$ represents the position of the COM of link i with respect to joint i which is the parent joint of link i. Having computed linear and angular acceleration and angular velocity of each link, Newton-Euler equations are applied for each link as shown below to compute forces and torques applied on each link.

$$F_i = m_i\,{}^{i}\dot{v}_{c_i} \tag{30}$$

$$N_i = {}^{c_i}I\,{}^{i}\dot{w}_i + {}^{i}w_i \times {}^{c_i}I\,{}^{i}w_i \tag{31}$$

In (30), F_i represents the forces acting on the COM of link i and m_i is the total mass of the i^{th} link. In (31), N_i is the torques acting on the i^{th} link and ${}^{c_i}I$ is the inertia tensor of the link i.

After computing the forces and torques acting on each link, joint torques are calculated through inward iterations. In order to calculate the joint torques which will result in net forces and torques acting on each link, force-balance and moment-balance equations based on a typical link can be derived as follows:

$$ {}^{i}F_i = {}^{i}f_i - {}^{i}_{i+1}R\,{}^{i+1}f_i \tag{32} $$

$$ {}^{i}N_i = {}^{i}n_i - {}^{i}_{i+1}R\,{}^{i+1}n_{i+1} - {}^{i}P_{c_i} \times {}^{i}F_i - {}^{i}P_{i+1} \times ({}^{i}_{i+1}R\,{}^{i+1}f_{i+1}) \tag{33} $$

In (32) and (33), ${}^{i}f_i$ and ${}^{i}n_i$ represent joint forces and torques respectively. In the algorithm for the quadruped robot, the Newton-Euler algorithm is constructed for one leg, and the algorithm is repeated for each leg.

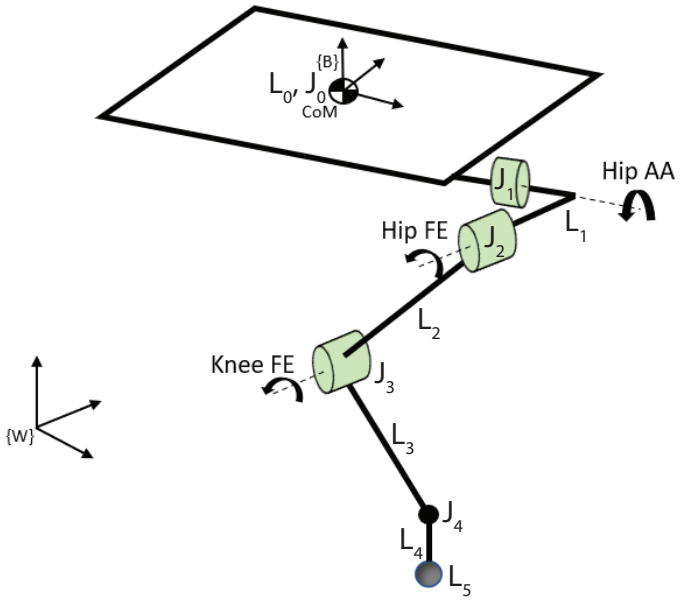

Figure 12: Joint configuration of quadruped robot.

The kinematic configuration of the robot's one leg is described in Fig. 12. In this figure, J_i and L_i represents the i^{th} joint and link respectively. Outward iterations begin with the floating base, which is link 0 iterates to one of the leg's foot

which is link 4. Floating base angular velocity and linear and angular accelerations are obtained through the IMU sensor. Then, angular velocity and linear and angular acceleration of link 1 through link 4 are computed by using (26), (27), and (29). Afterward, forces and torques acting on each link from link 1 to link 4 are calculated by using (32) and (33). Inward iterations begin from the tip of the foot, which is enumerated as 5 and represents the ground reaction forces and iterated through each joint until the hip A/A joint which is enumerated as 1. Joint forces and torques are computed through (32) and (33), respectively [26].

3.5 Capture Point

The capture point is the point on the ground for which the robot's kinetic energy can be led to zero with a feasible COM trajectory. Using to the capture point, the robot can maintain its ZMP to stay on the capture point either with its stance foot or by stepping toward the capture point [27].

$$\ddot{x} = x\frac{g}{z_0} \tag{34}$$

From the equation of motion of the linear inverted pendulum model shown in (34), the orbital energy of a linear inverted pendulum can be obtained as shown below:

$$E_{LIP} = \dot{x}^2\frac{1}{2} - x^2\frac{g}{2z_0} \tag{35}$$

By examining (35), one can say that if the COM is moving toward the foot and E_{LIP} is positive, it means that system has enough energy for the COM to move over the foot and keep going on its way. However, if E_{LIP} is negative, the COM will stop and reverse directions before getting over the foot. If the orbital energy is zero, the COM will come to rest over foot [27].

To achieve stability against external forces, a capture point can be computed. To compute the capture point, the required foot placement to obtain orbital energy of zero is the main interest. Therefore, by setting 35 to zero, the capture point

can be computed as follows:

$$x_{cp} = \dot{x}\sqrt{\frac{z_0}{g}} \tag{36}$$

In the robot, the capture point is calculated by using 36. Then, a threshold for the calculated capture point is determined. According to the selected threshold value robot takes one or more steps to the capture point.

3.6 Basic Controller Scheme

Fig. 13 shows the overview of the basic control framework. In this figure, q_j denotes the actuated joint positions of each leg, and q denotes the generalized coordinates of the robot that is defined as $q = \begin{bmatrix} q_f & q_j \end{bmatrix}^T$. Here, q_f symbolizes the floating base coordinates. For the dynamic locomotion of the robot, a ZMP-based trajectory planning algorithm was implemented for trot-walking; see section 3.2. The trajectory planning algorithm takes the commands from the operator. These commands are mean COM velocity, torso yaw orientation, single support (T_s), and double support (T_d) duration. The trajectory planner computed the desired foot trajectory for each leg in accordance with the operator commands and fed it to the inverse kinematic algorithm. The inverse kinematic algorithm computes the desired joint positions (q_j^{ref}) for each leg and feeds those desired joint positions to the computed torque controller. Implementing computed torque control, whose output is command torque T_{cmd} for each joint, which is fed to the system. The computed torque control method consists of two main blocks. The first one is the inverse dynamic block. In this block, system dynamics are modeled by using a recursive Newton-Euler algorithm. The contact forces are constructed by distributing the weight of the robot equally to the contact feet. The second block of computed torque control is the PD controller in order to compensate for the unmodeled dynamics in the inverse dynamic block.

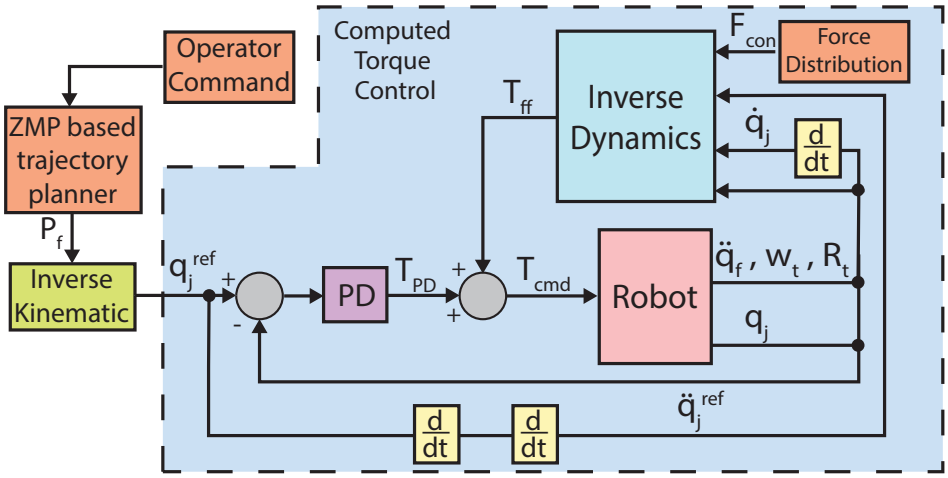

Figure 13: Overview of basic controller scheme.

3.7 Force Distribution via Centroidal Momentum Feedback

Contact forces at the feet of the quadruped robot have an essential role in dynamic locomotion. The dynamic model of a quadruped robot can be defined with

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = ST_j + J^T F_c \tag{37}$$

where $M(q)$ is the mass matrix, $C(q,\dot{q})$ is the Coriolis forces, $G(q)$ is the gravitational forces, q is the generalized coordinates, T_j is the joint torques, S is the selection matrix, J is the contact Jacobian matrix, and F_c is contact forces. For more stabilized and dynamic locomotion, contact force distribution can be used to regulate the robot's behavior. In a naive approach, contact forces can be distributed by taking into account the total weight of the robot, and then desired contact forces can be computed accordingly. Since this method only characterizes vertical forces, it has been observed that this method causes drift in the yaw angle of the robot [28]. Thus, instead of distributing the robot's total weight equally to each contact foot, contact forces are computed via Virtual Model Control (VMC) with centroidal momentum feedback.

The overview of the overall locomotion control framework is shown in Fig. 14. In this figure, q_j denotes the actuated joint positions of each leg, and q denotes the

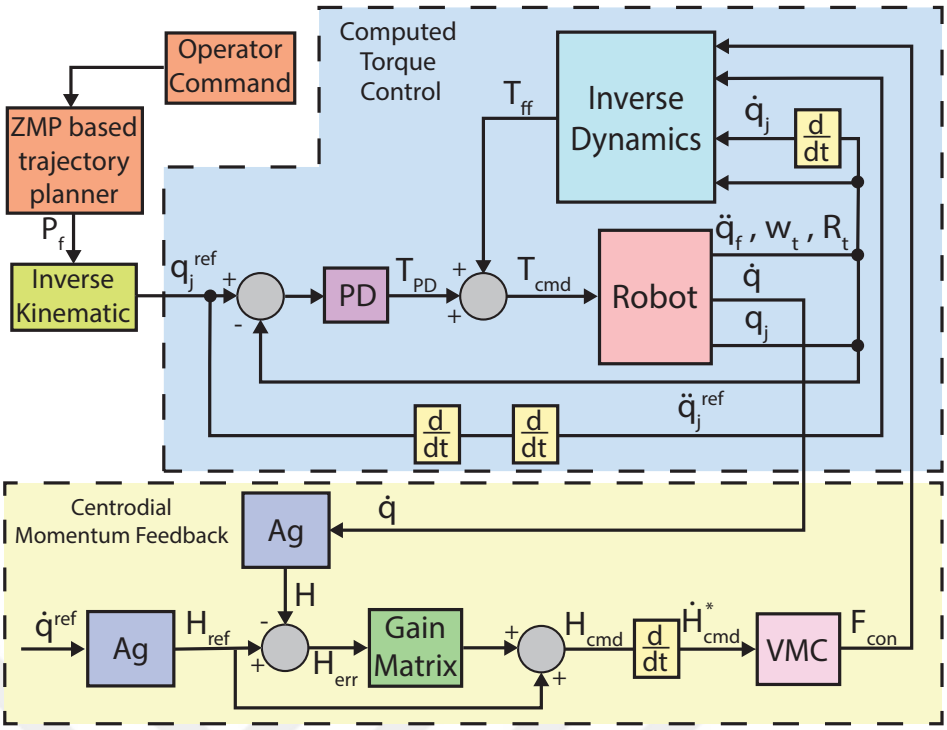

Figure 14: Overview of locomotion control framework.

generalized coordinates of the robot that is defined as $q = \begin{bmatrix} q_f & q_j \end{bmatrix}^T$. Here, q_f symbolizes the floating base coordinates.

VMC is a control framework that allows the computation of the desired joint torques by rendering virtual components. These virtual components can be defined as springs, dampers, masses, bearings, and so on [29]. It was reported that the virtual model controller was more advantageous than tuning the joint level controller gains in order to change the dynamic behavior of the system [30]. To that end, desired dynamic behavior of the system can be rendered through virtual model control. For the implementation of virtual model control, desired forces and moments acting on the torso (F_t and M_t) are defined as virtual forces. These virtual forces and moments should be balanced with the ground reaction forces f_c^i, where i indicates which foot has contact with the ground. In doing so, desired ground reaction forces can be computed as follows, in accordance with the desired virtual forces and moments.

$$\begin{bmatrix} f_c^1 \\ . \\ . \\ . \\ f_c^i \end{bmatrix} = \begin{bmatrix} 1_{3x3} & 1_{3x3} & ... & 1_{3x3} \\ S({}^1P_G) & S({}^2P_G) & ... & S({}^iP_G) \end{bmatrix}^{\dagger} \begin{bmatrix} F_t \\ M_t \end{bmatrix} \tag{38}$$

In (38), 1_{3x3} indicates a three by three identity matrix, ${}^iP_G$ indicates the foot contact position with respect to the center of mass and $S(*)$ indicates the skew-symmetric matrix of an input vector.

The centroidal momentum of a robot is the sum of the projected momentum of each link relative to the aggregated COM [31]. The spatial momentum of each link is a vector consisting of linear and angular momenta. Its calculation concerning the i^{th} link can be expressed below.

$$h_i = I_i v_i \tag{39}$$

In (39), I_i is the spatial inertia of the i^{th} link, and v_i is the spatial velocity of i^{th} link. The spatial inertia of the i^{th} body is calculated as follows:

$$I_i = \begin{bmatrix} \bar{I}_i & m_i S(c_i) \\ m_i S(c_i)^T & m_i \end{bmatrix} \tag{40}$$

In (40), $\bar{I}_i$ is the Cartesian inertia tensor, m_i represents the mass, and c_i is the relative center of mass position of i^{th} body. By summing up all the projected spatial momenta of each link, centroidal momentum is calculated. Thus, the spatial momentum of each link with respect to its own body frame can be transformed to the composite CoM by using a spatial momentum transformation matrix (${}^iX_G^T$):

$${}^iX_G^T = \begin{bmatrix} {}^GR_i & {}^GR_i S({}^ip_G) \\ 0 & {}^GR_i \end{bmatrix} \tag{41}$$

In Eq. (41), ${}^GR_i$ represents the rotation matrix and ${}^ip_G$ represents the position vector. Finally, centroidal momentum can be derived by adding up all the

transformed momenta [32].

$$H = \sum_{i=1}^{N} {}^{i}X_{G}^{T} h_i \tag{42}$$

The centroidal momentum is a function of generalized velocities. Therefore, it can be expressed as below,

$$H = A_g \dot{q}, \tag{43}$$

where H is the centroidal momentum, A_g is the centroidal momentum matrix, and $\dot{q}$ is the generalized velocity vector. A_g matrix can be computed as in the following.

$$A_g = {}^{1}X_{G}^{T} \Psi_{1}^{T} U_1 M(q) \tag{44}$$

In (44), ${}^{i}X_{G}^{T}$ represents spatial momentum transformation matrix, Φ_1 is the joint matrix which transforms quantity represented in the world frame to floating-base frame and Ψ_1 is defined as $\Psi_1 = \Phi_1^{-1}$. Furthermore, U_1 is defined as $U_1 = \begin{bmatrix} 1_{6x6} & 0_{6x12} \end{bmatrix}$ where 1_{6x6} is identity matrix and 0_{6x12} is a zero matrix [31].

In order to compute the contact force distributions via VMC with centroidal momentum feedback, the derivative of the desired centroidal momentum can be selected as virtual components. As a next step, desired contact forces, F_{con}, can be computed in accordance with these virtual components and plugged into the inverse dynamics block of the computed torque controller. In this sense, plugging the derivative of the desired centroidal momentum alone as a virtual component may not minimize the centroidal momentum error. To minimize for the centroidal momentum error, H_{err}, a controller with a feedback and feedforward controllers are implemented such that the command centroidal momentum, H_{cmd}, is computed.

$$H_{cmd} = H_{ref} + K_h H_{err}, \tag{45}$$

In (45), K_h is a diagonal matrix that stores centroidal feedback controller gains. Furthermore, the gravitational force acting on the whole system is directly added

to the derivative of the linear centroidal momentum command; $\dot{H}^{*}_{cmd} = \dot{H}_{cmd} + \begin{bmatrix} 0 & 0 & mg & 0 & 0 & 0 \end{bmatrix}^{T}$, where m and g represent the mass of the whole system and gravity respectively. Finally, the derivative of the centroidal momentum command ($\dot{H}^{*}_{cmd}$) is selected as a virtual component of VMC, and finally, F_{con} is yielded:

$$F_{con} = \begin{bmatrix} 1_{3x3} & 1_{3x3} & ... & 1_{3x3} \\ S({}^{1}P_{G}) & S({}^{2}P_{G}) & ... & S({}^{i}P_{G}) \end{bmatrix}^{\dagger} \dot{H}^{*}_{cmd} \tag{46}$$

Afterward, these computed contact forces F_{con} are fed to the inverse dynamics block, along with joint positions q_j, joint velocities, $\dot{q}_j$, reference joint accelerations $\ddot{q}_j$ and floating base orientation, angular velocity, angular and linear accelerations, and the required joint torques are calculated through Newton-Euler algorithm.

CHAPTER IV

SIMULATION AND EXPERIMENT RESULTS

The simulations/experiments are conducted to verify that the quadruped robot is able to achieve a set of goals with the proposed control framework. These goals are set as follows:

- The robot should be able to trot-walking with a maximum mean forward velocity of 0.5 $\frac{m}{s}$.
- The robot should be able to carry a 20 kg payload.
- The robot should be able to maintain its balance against external forces.
- The robot should be able to operate for at least one hour with a 5.5 kg payload and a single fully charged battery.

In order to validate that the robot succeeds in these achievements, simulations/experiments are conducted. These tests are walking simulation and experiment with 0.5 $\frac{m}{s}$ forward velocity, walking experiment with 20 kg payload, and endurance test (1 hour walking with 5.5 kg payload). Besides, the proposed contact force distribution via centroidal momentum feedback controller is validated in the simulation environment.

4.1 Walking Simulation: 0.5 m/s Forward Velocity

Before applying the locomotion control framework on the robot, a simulation studies were conducted. In the simulation, the robot executes trot-walking with a mean velocity of $0.5\frac{m}{s}$. The trajectory planner generates COM and feet trajectories online. The result of the online trajectory planner can be seen in Fig. 15. In this figure, the blue line indicates the COM trajectory, which increases as the desired mean velocity is kept constant. Moreover, the yellow and red line indicates the left

and right foot, respectively. The left foot and right foot are equivalent to the biped model of a quadruped robot [14]. Hence, the left foot indicates the trajectory of both left-front and right-back feet, and the right foot indicates both right-front and left-back feet. As seen in Fig. 15, the COM trajectory planner generates continuous and feasible COM and feet trajectories. Additionally, COM velocity and acceleration graphs are generated continuously, as shown in 16 and 17. The mean velocity of the generated trajectory is $0.534\frac{m}{s}$ and constant. Similarly, mean acceleration is zero for constant mean velocity as expected.

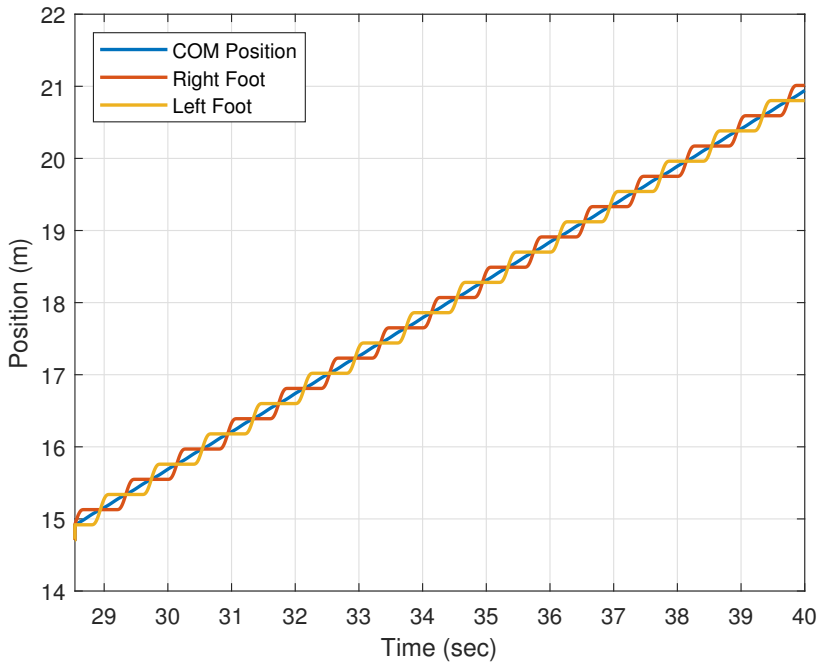

Figure 15: Generated COM and feet trajectories for gradually increasing mean velocity.

In order to determine if the locomotion is dynamically balanced, ZMP locations should be inside the support polygons. In Fig. 18 and 19, blue and red lines represents the upper and lower bounds of the support polygon, respectively. By examining these figures, both X_{zmp} and Y_{zmp} are located inside the upper and lower limits o the support polygon, which results in dynamically balanced locomotion for a quadruped robot.

Furthermore, the joint tracking of the hip F/E joint of the left-front leg can be seen in Fig. 20. Tracking performance graphs of the rest of the joints can be

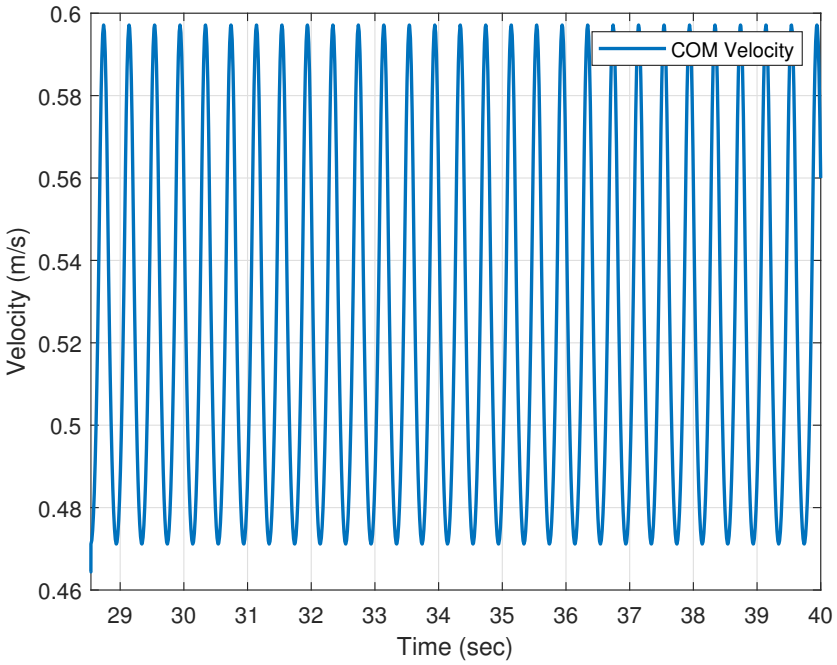

Figure 16: Generated COM velocity for gradually increasing mean velocity.

seen in Fig. 39-48 in APPENDIX A In these figures, the blue lines represent the joint reference trajectories, and the red lines represent actual joint variations. Hip A/A joints of each leg have negligible oscillation, and hip F/E and knee F/E joints have negligible overshoots and undershoots. RMS error of each joint can be seen in Table 1 and 2. RMS errors of each joint are almost zero, which yields good tracking for the joints. RMS errors of joints are increased from hip A/A joint to knee F/E joint for each leg. This increase in the RMS errors causes due to impact forces on the feet. These forces are absorbed by the PD controller of the joints, and the knee joint is the closest joint to the foot. Hence, relatively peak RMS error occurs at knee joints.

Table 1: Joint tracking of front legs RMS errors in simulation.

Left-Front Leg		Right-Front Leg	
Joint Name	RMSE (rad)	Joint Name	RMSE (rad)
Hip A/A	1.8272E-05	Hip A/A	1.7116E-05
Hip F/E	7.7311E-04	Hip F/E	8.2258E-04
Knee F/E	0.0019	Knee F/E	0.0019

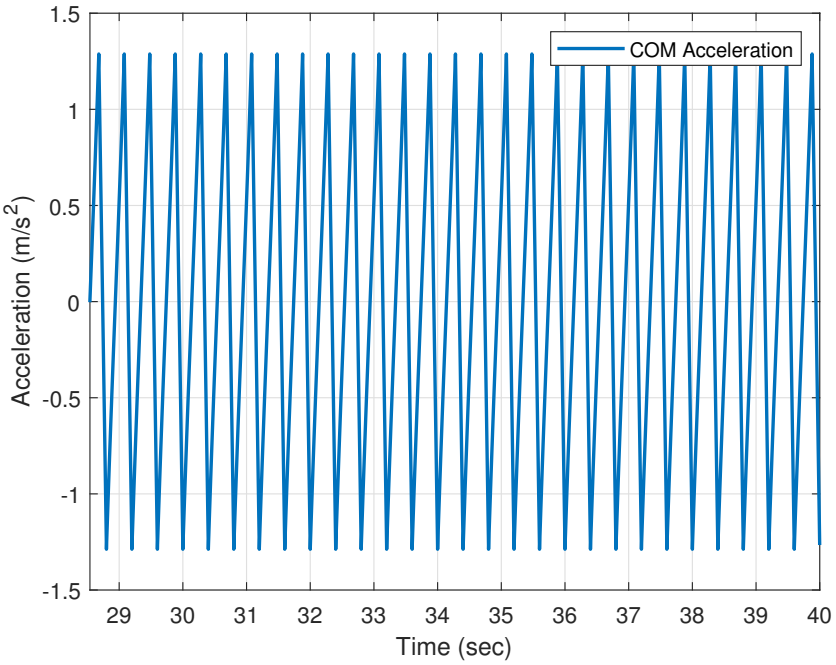

Figure 17: Generated COM acceleration for gradually increasing mean velocity.

Table 2: Joint tracking of back legs RMS errors in simulation.

Left-Back Leg		Right-Back Leg	
Joint Name	RMSE (rad)	Joint Name	RMSE (rad)
Hip A/A	6.7276E-06	Hip A/A	6.3575E-06
Hip F/E	9.3954E-04	Hip F/E	9.4556E-04
Knee F/E	0.0019	Knee F/E	0.0019

4.2 Walking Experiment: 0.5 m/s Forward Velocity

Walking experiments were conducted on a quadruped robot that is built in our laboratory. In these experiments, walking speed of the robot is gradually increased from zero to $0.5\frac{m}{s}$. Online trajectory planner generates desired trajectories for COM and feet as shown in Fig. 21. In this figure, the blue line indicates the COM trajectory, which increases exponentially as the desired mean velocity increased gradually. In Fig. 21, the yellow and red line indicates left and right foot, respectively. As mentioned in the previous section, the left foot and right foot are equivalent biped model of a quadruped robot. Generated COM velocity and acceleration for gradually increasing desired to mean velocity are shown in Fig.

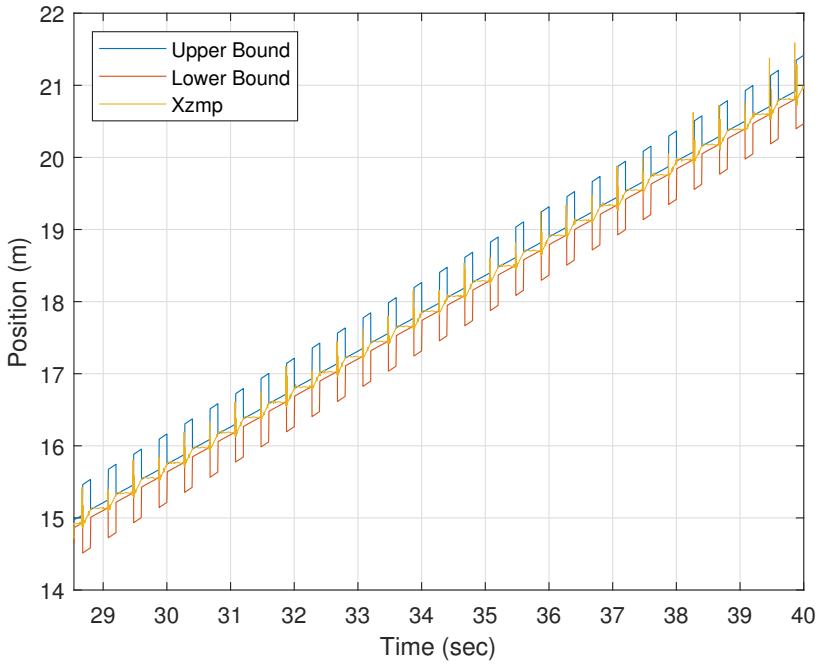

Figure 18: ZMP response along x-axis.

22 and 23. In Fig. 22, the mean of generated COM velocity gradually increases with desired mean velocity. In Fig. 23, the mean of generated COM acceleration is constant, as expected. Examining COM velocity and acceleration graphs shows the increase of maximum and minimum acceleration profiles caused due to the increasing velocity command.

X_{zmp} and Y_{zmp} positions are shown in Fig. 24 and 25. In these figures, the blue and red lines are upper and lower bound for the support polygon, and the yellow line indicates ZMP location along the x-axis and y-axis. ZMP locations are computed by using (1) and (2); see chapter 3. Observing the ZMP data, we see that X_{zmp} and Y_{zmp} are inside the support polygon, resulting in dynamically balanced locomotion achieved by the proposed locomotion control framework.

Moreover, joint tracking of the hip F/E joint of the left-front leg can be seen in Fig. 26. Tracking performance graphs of the rest of the joints of the robot can be seen in Fig. 49-59 in APPENDIX A. In these figures, the blue line represents the joint reference positions, and the red line represents actual joint positions in radians. Hip A/A joints of each leg have negligible oscillation, and hip F/E and

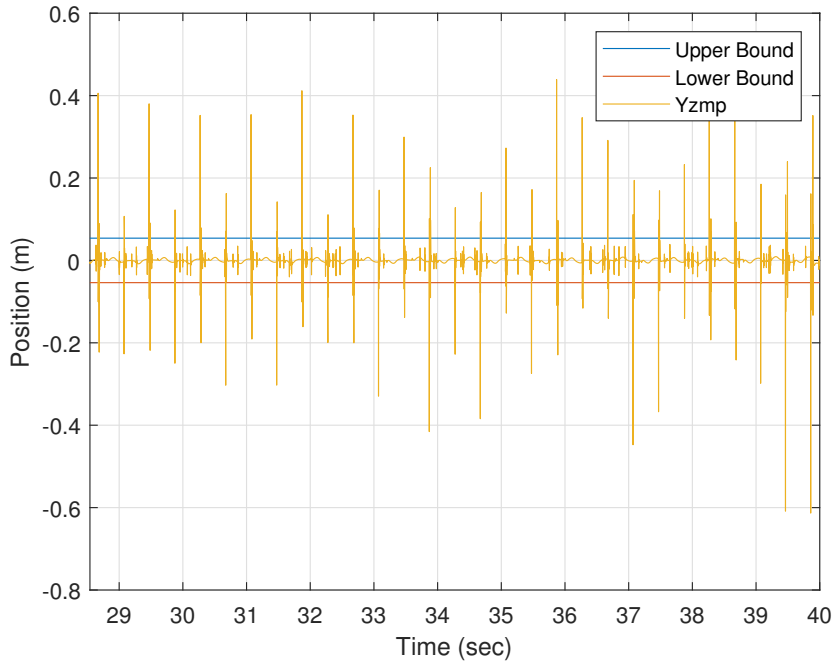

Figure 19: ZMP response along x-axis.

knee F/E joints have negligible overshoots and undershoots. RMS error of each joint can be seen in Table 3 and 4. RMS errors of each joint are almost zero, which yields good tracking for the joints. Similar to the simulation results, RMS errors of the robot's joints also increase from hip A/A joint to knee F/E joint for each leg. As mentioned 4.1, the increase in the RMS errors is caused due to impact forces on the feet. These forces are absorbed by the PD controller of the joints, and the knee joint is the closest joint to the foot. Hence, peak RMS error occurs at knee joints. Unlike simulation results, back leg hip F/E joints RMS errors are more than front leg hip F/E joints RMS errors. This difference between front and back leg hip F/E joints may be due to the uncertainty in the inverse dynamic model.

4.3 Walking Experiments with a Payload of 20 kg

In order to conduct this experiment, a 20 kg payload was added onto the torso of the robot. The maximum mean velocity of the robot was set to the 0.2 $\frac{m}{s}$. The weight of the payload was added to the weight of the robot in the inverse

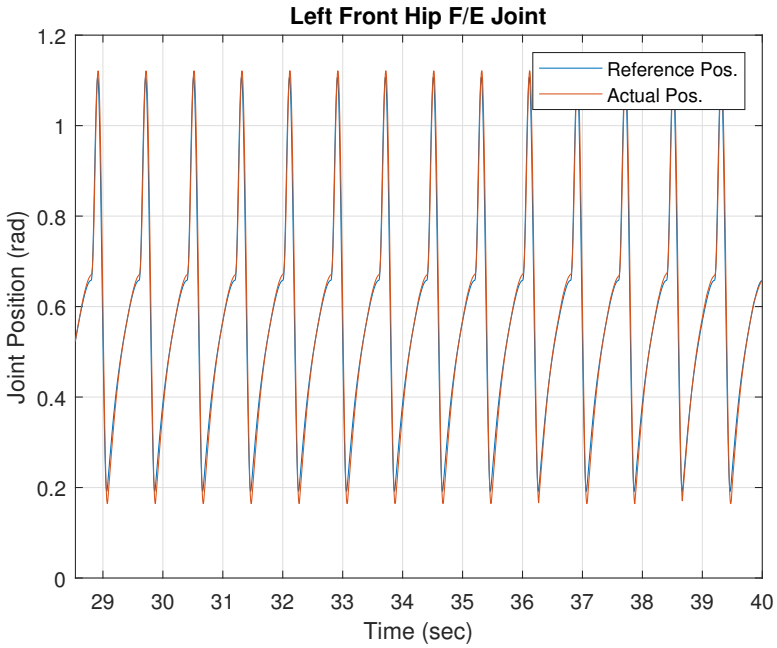

Figure 20: Joint tracking of hip F/E joint of left front leg.

Table 3: Joint tracking of front legs RMS errors.

Left-Front Leg		Right-Front Leg	
Joint Name	RMSE (rad)	Joint Name	RMSE (rad)
Hip A/A	5.1458e-04	Hip A/A	9.3164e-04
Hip F/E	9.1885e-04	Hip F/E	9.3162e-04
Knee F/E	0.0024	Knee F/E	0.0029

dynamics algorithm in order to calculate the joint torques. As seen in Fig. 27, the robot's mean velocity is increased gradually to maximum mean velocity to avoid any discontinuity in the COM trajectories. As a result, the robot performed a dynamically balanced trot-walking with a 20 kg payload. Fig. 28 and 29 indicates X_{zmp} and Y_{zmp} positions with upper and lower bounds of support polygon. Examining these figures, X_{zmp} and Y_{zmp} stay inside the support polygon during trot-walking with a mean velocity of 0.2 $\frac{m}{s}$, which results in dynamically balanced locomotion.

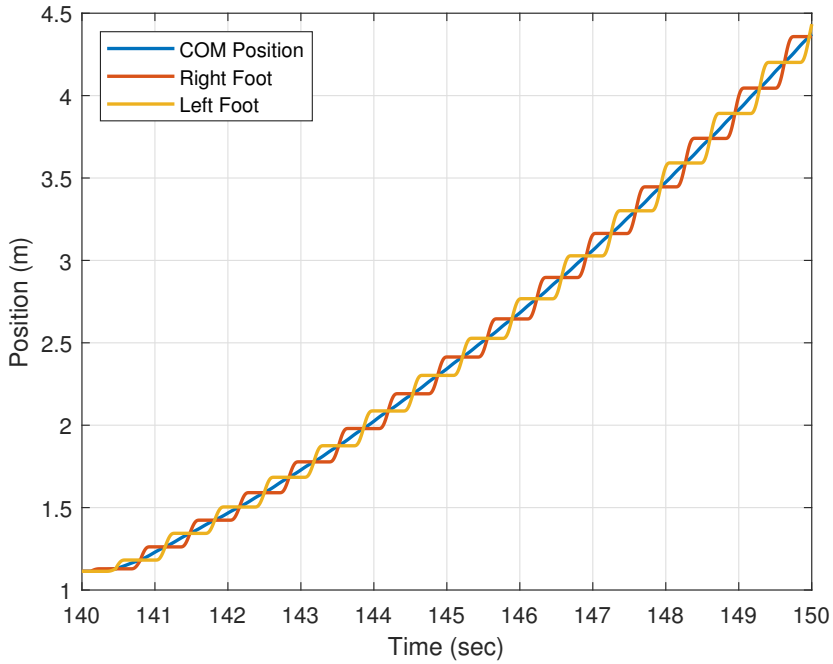

Figure 21: Generated COM and feet trajectories for gradually increasing mean velocity.

Table 4: Joint tracking of back legs RMS errors.

Left-Back Leg		Right-Back Leg	
Joint Name	RMSE (rad)	Joint Name	RMSE (rad)
Hip A/A	7.9436e-04	Hip A/A	5.6027e-04
Hip F/E	0.0018	Hip F/E	0.0021
Knee F/E	0.0025	Knee F/E	0.0026

4.4 Endurance Test

The endurance experiment was conducted on a treadmill with a fully charged battery. The robot was placed on a treadmill, and a 5.5 kg payload was loaded on the torso of the robot. In this experiment, the robot was initially placed on the treadmill with a crane. Afterward, both treadmill and robot velocity were increased to 0.2 $\frac{m}{s}$ synchronously, and the robot performed trot-walking for one hour. Instead of tracking the timer, the displacement of the treadmill, which can be calculated by the treadmill software, was tracked. Therefore, the robot kept walking until the treadmill's total distance travel reached $750m$, which means the robot kept trot-walking with 0.2 $\frac{m}{s}$ for one hour in average. The COM velocity

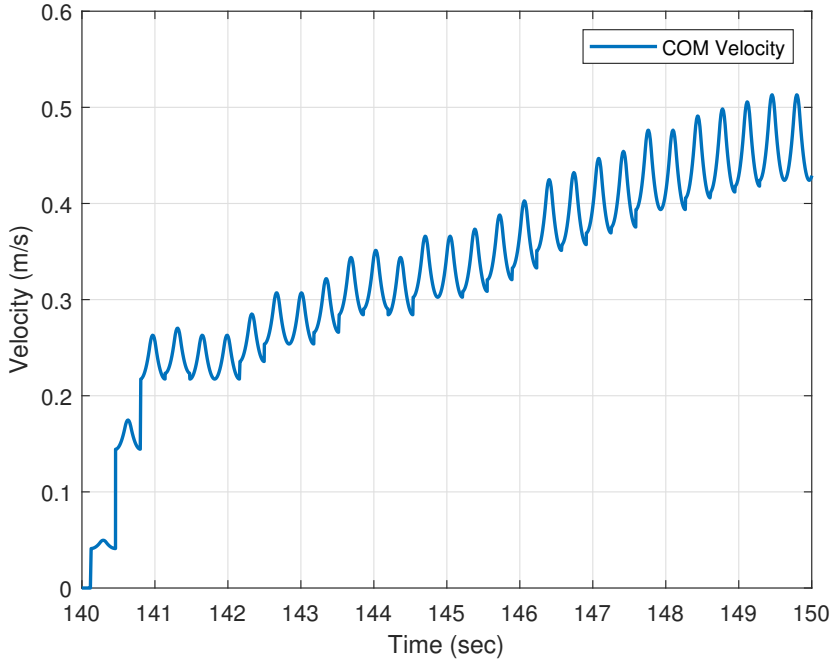

Figure 22: Generated COM velocity for gradually increasing mean velocity.

profile can be seen in Fig. 30. This figure shows that the mean velocity of the robot is 0.2 $\frac{m}{s}$.

4.5 Balancing Experiments

In this experiment, the capture point algorithm was verified. To conduct this experiment robot was placed on the ground. The robot was disturbed by pushing it from side of the torso. The disturbance force was applied along the y-axis. ZMP locations are shown in Fig. 31 and 32. In these figures, the blue and red lines are the upper and lower bound of the support polygon, and the yellow line is the ZMP location. By analyzing Fig. 31 and 32, lateral force is applied to the torso at around 87^{th} second, and the robot starts stepping in order to achieve balance by keeping both ZMP locations inside the support polygon.

4.6 Force Distribution via Centroidal Momentum

Simulation experiments were conducted in RaiSim v1.1.5 as explained in Section 3.1. Two different scenarios were considered:

i Trot-walking at 0.5 m/s

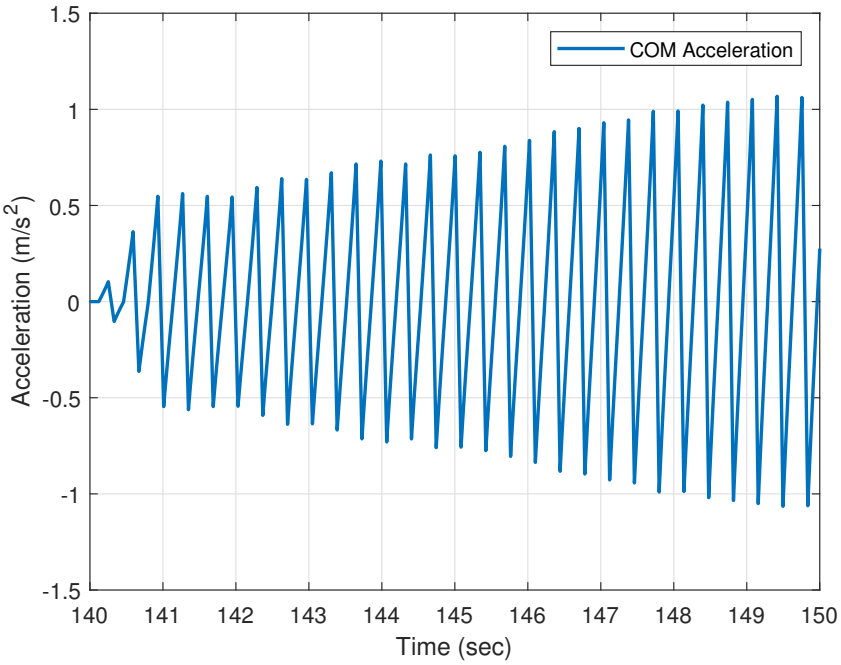

Figure 23: Generated COM acceleration for gradually increasing mean velocity.

ii Trot-walking at 0.5 m/s with a 4 kg payload

In each scenario, simulations were conducted two times: First, centroidal momentum feedback was removed, and contact forces were simply distributed by checking the contact feet and dividing the robot's total weight to contact feet equally. Second, the simulation scenario was repeated with the proposed contact force distribution method in effect.

4.6.1 Scenario 1: Trot-Walking at 0.5 m/s

In this scenario, the robot started trot-walking after 3 seconds, and its mean velocity was increased with a ramp input, from $0m/s$ to $0.5m/s$ (slope: 0.02). In Fig. 33, torso Euler angles (Roll, Pitch, Yaw) and angular velocities (ω_x, ω_y, ω_z) are compared. The blue lines indicate the behavior of the torso without the proposed method, while the orange lines show the torso behavior with the proposed centroidal momentum feedback. Examining Fig. 33 and Fig. 34, RMS error values concerning the torso orientation were $0.1089deg$ in roll, $0.2386deg$ in pitch, $2.5709deg$ in yaw when the proposed controller was not activated. On the contrary,

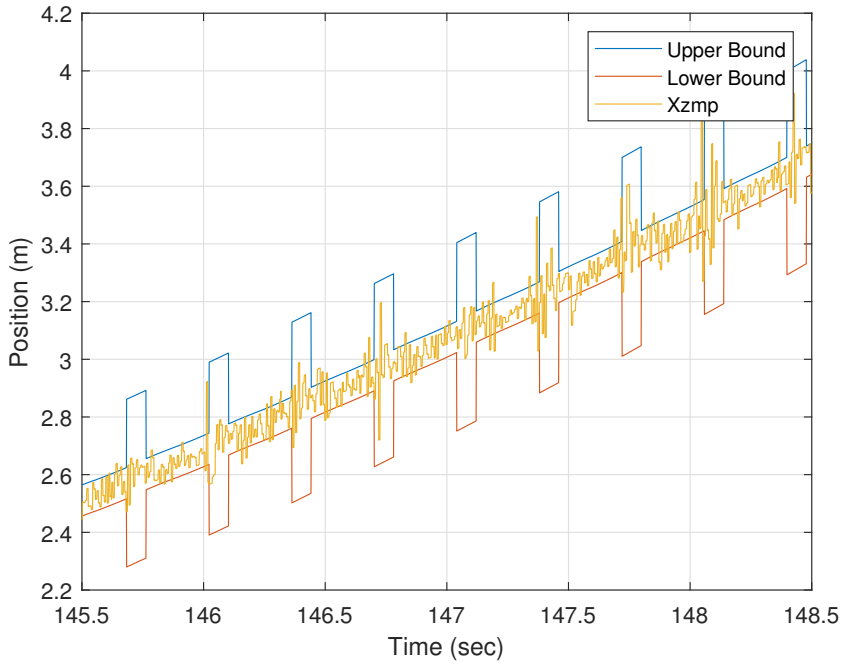

Figure 24: ZMP response along x-axis.

these values were dropped to $0.1053deg$, $0.0999deg$, $0.0986deg$, respectively. As a result, we observed a significant improvement in yaw axis, the robot heading was truly well-regulated. Furthermore, pitch axis orientation was also quite improved whereas the decrease in roll axis was relatively insignificant. A positive outcome was also observed in torso angle fluctuations; the RMS error values of angular velocity measurements along the roll, pitch, and yaw axes decreased by 29.7%, 65.8%, 61.7%, respectively.

The robot's balancing ability was also improved with the help of the proposed contact force distribution method. When it was activated, the RMS values of ZMP errors along x-axis and y-axis decreased for about 15.0% and 64.2%, respectively. See Fig. 35 for changes in ZMP errors concerning both x-axis and y-axis.

4.6.2 Scenario 2: Trot-Walking with a 4 kg Payload

In this scenario, a 4 kg payload was added to the top of the torso and trot-walking with the same parameters concerning the first scenario was applied. As seen in Fig. 36, the proposed method caused RMS error values to decrease in the torso orientation; namely, 29.1% in roll angle, 78.7% in pitch angle, 92.5% in yaw

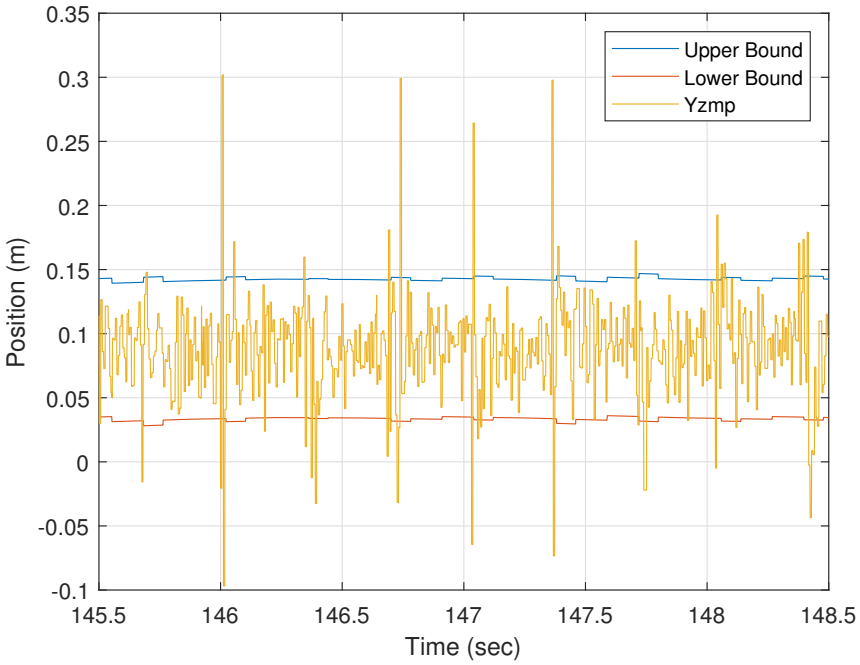

Figure 25: ZMP response along y-axis.

angle. Furthermore, RMS error values of the torso angular velocities decreased by approximately 56.0% along the roll axis, 73.8% along the pitch axis, and 72.0% along the yaw axis; see Fig. 37.

Observing Fig. 38, RMS value of X_{zmp} error dropped from 0.0153 m to 0.0109 m and RMS value Y_{zmp} error decreased from 0.003 m to 7.8x10^{-4} m. Hence, centroidal momentum feedback with the proposed control scheme minimizes the ZMP errors and increases the balancing capability of the robot despite the 4 kg payload.

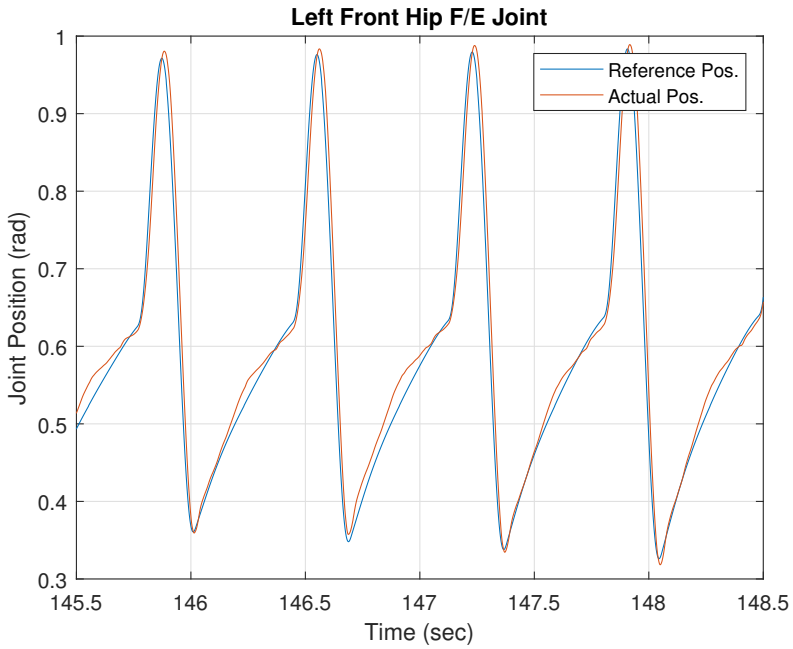

Figure 26: Joint tracking of hip F/E joint of left front leg.

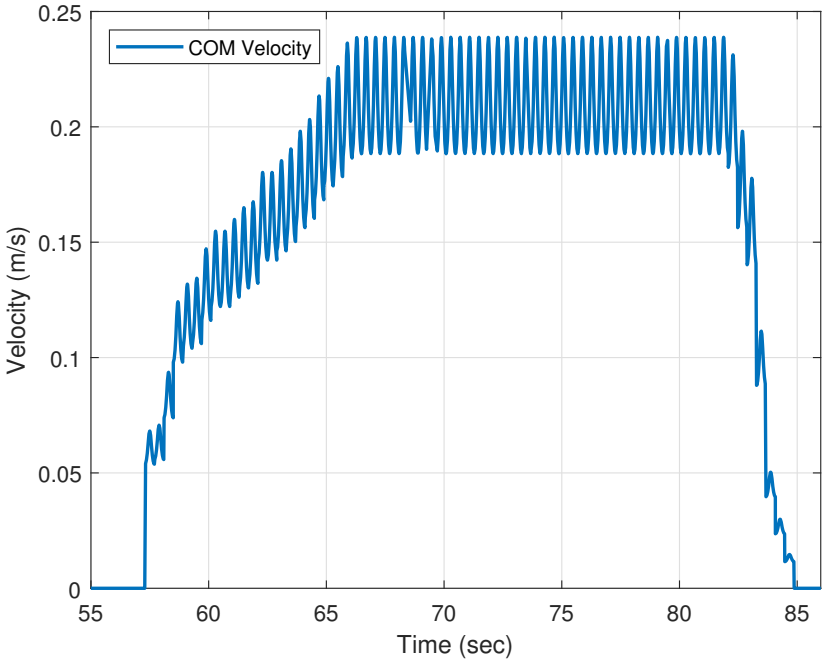

Figure 27: COM velocity with 20 kg payload.

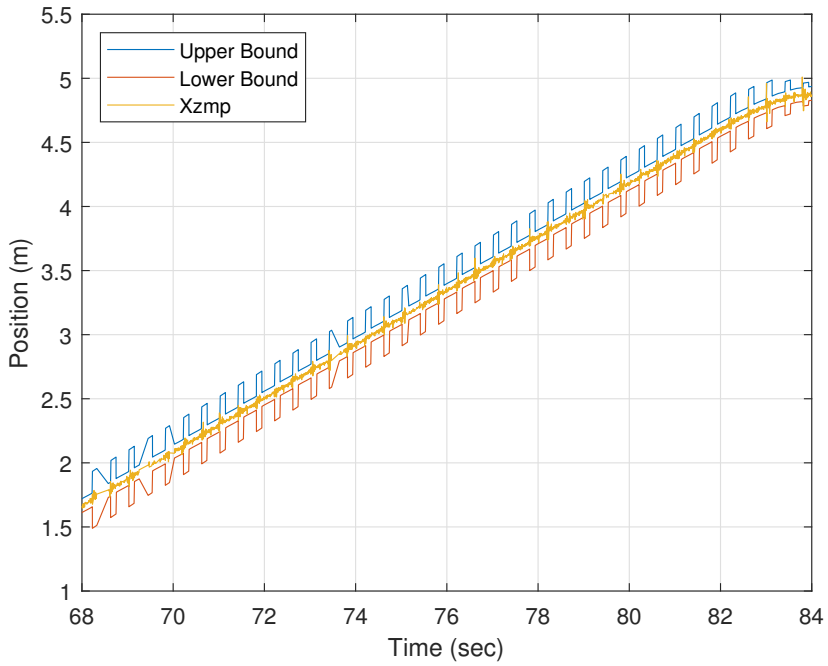

Figure 28: ZMP response with 20 kg along x-axis.

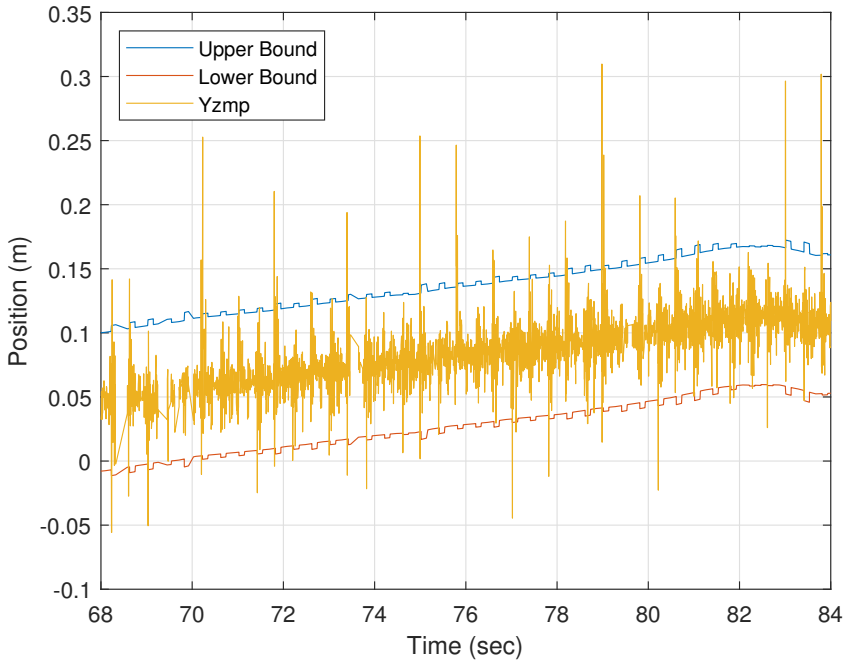

Figure 29: ZMP response with 20 kg along y-axis.

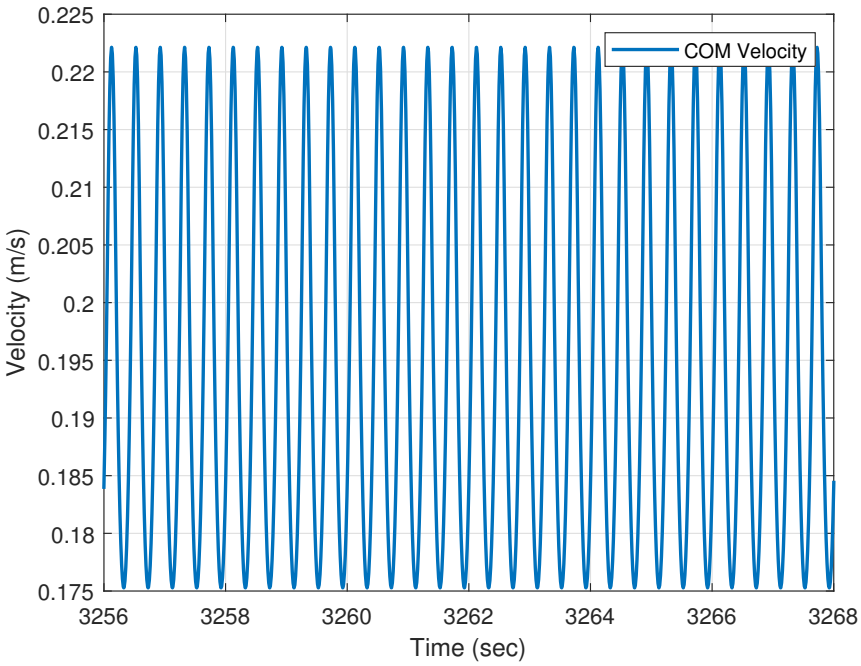

Figure 30: COM velocity.

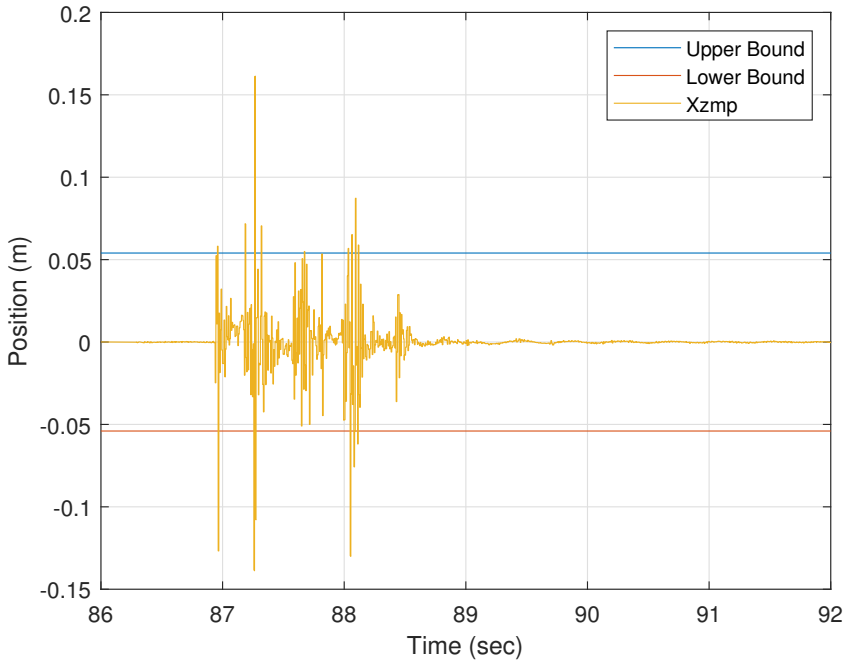

Figure 31: ZMP location along x-axis.

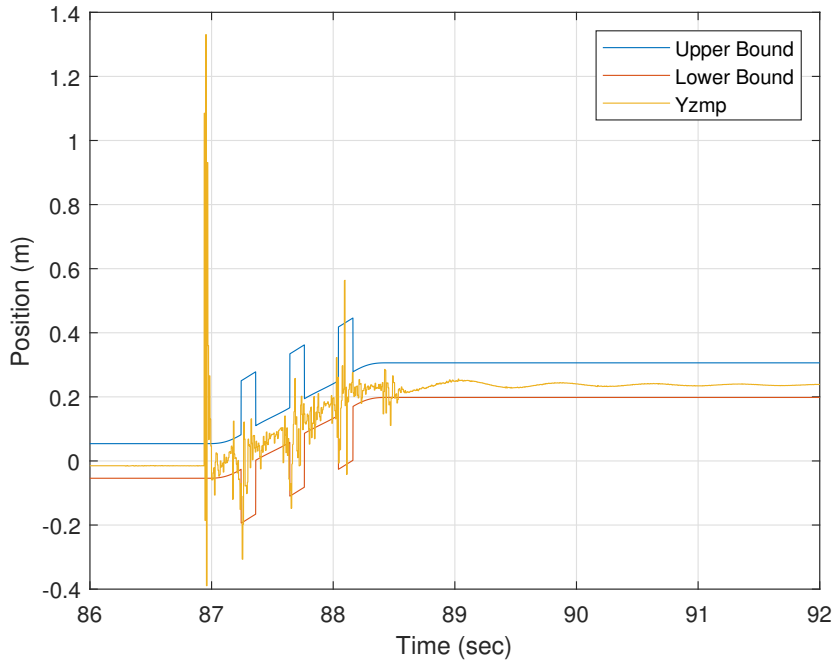

Figure 32: ZMP location along x-axis.

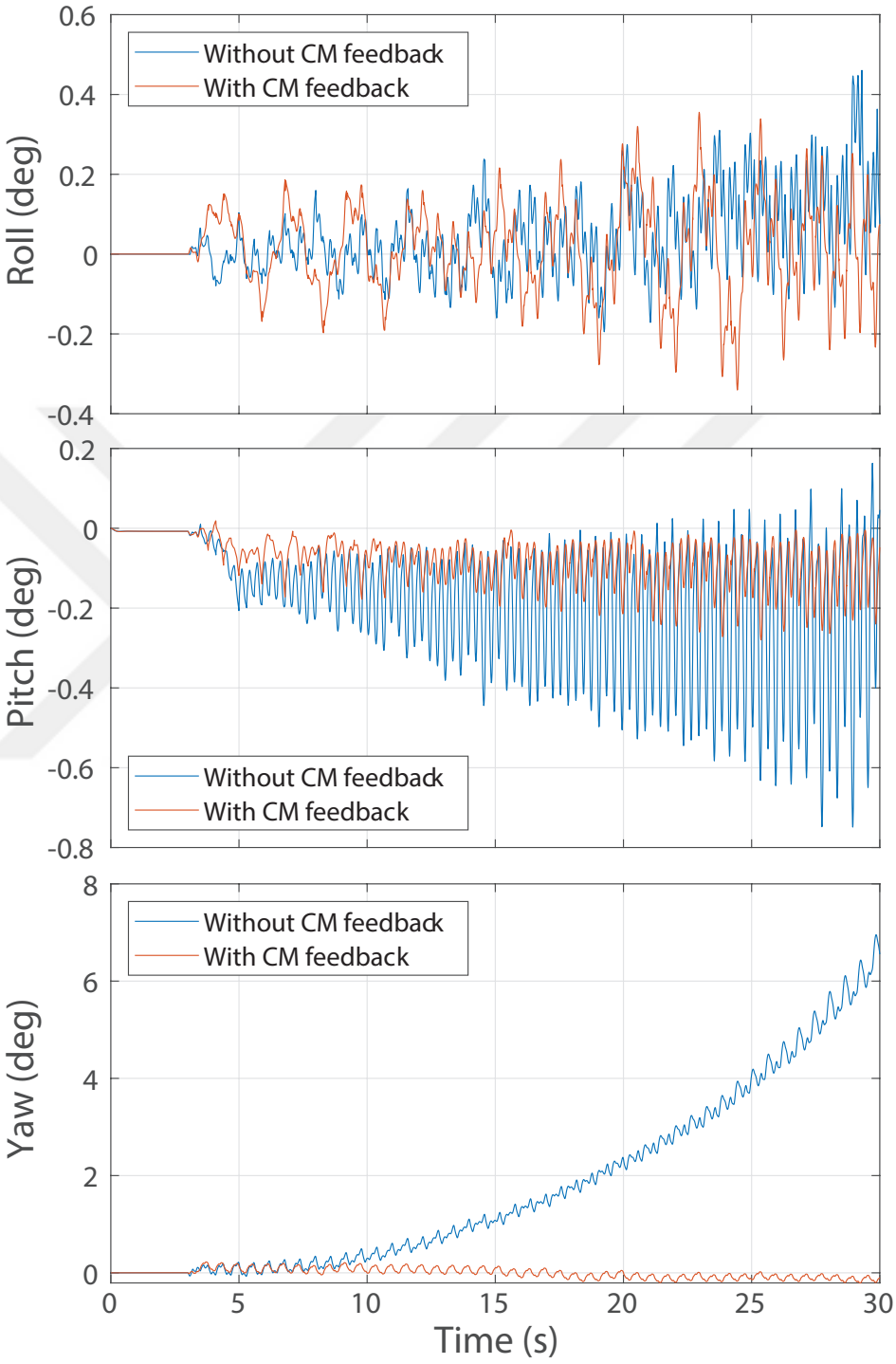

Figure 33: Torso orientation without payload.

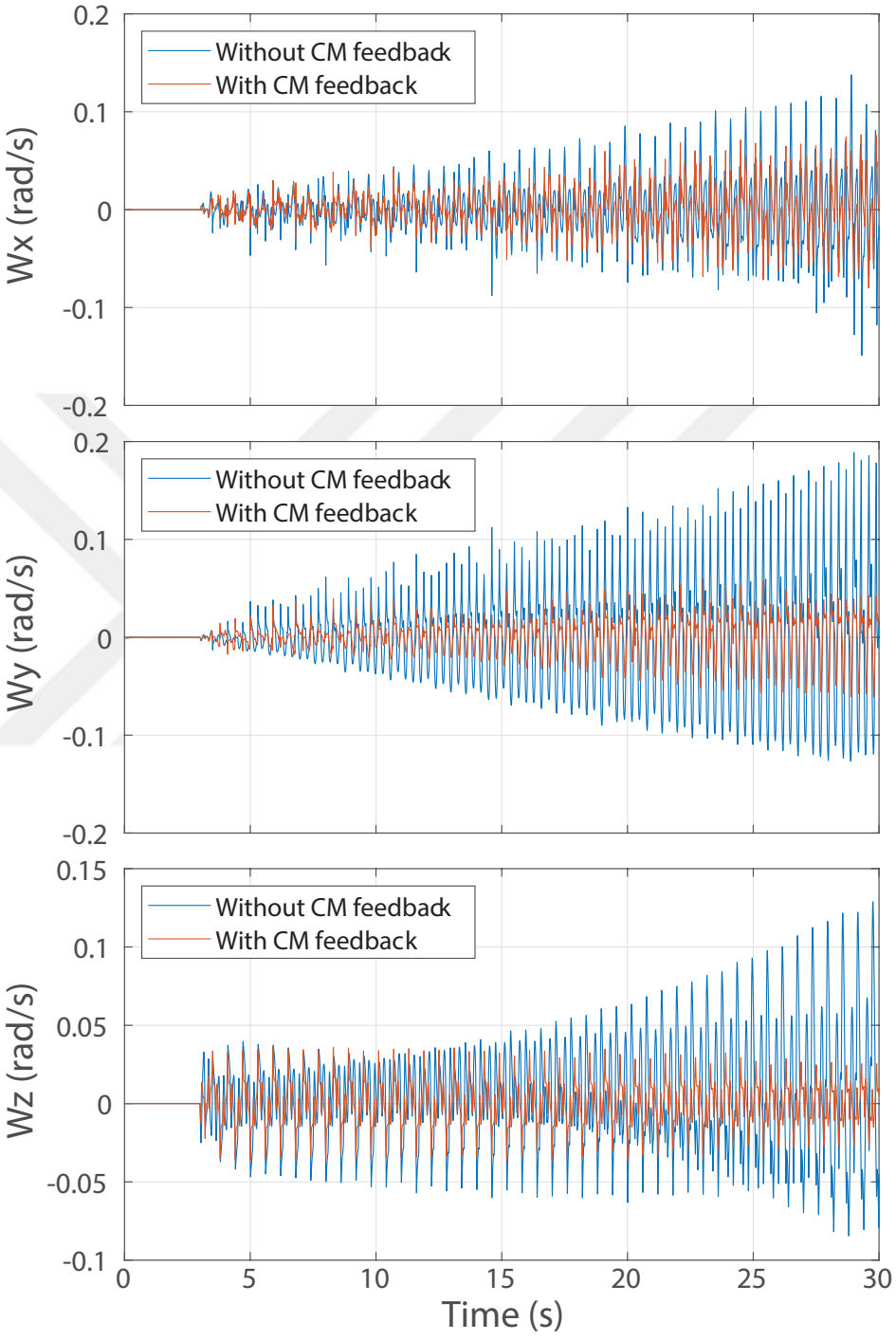

Figure 34: Torso angular velocity without payload.

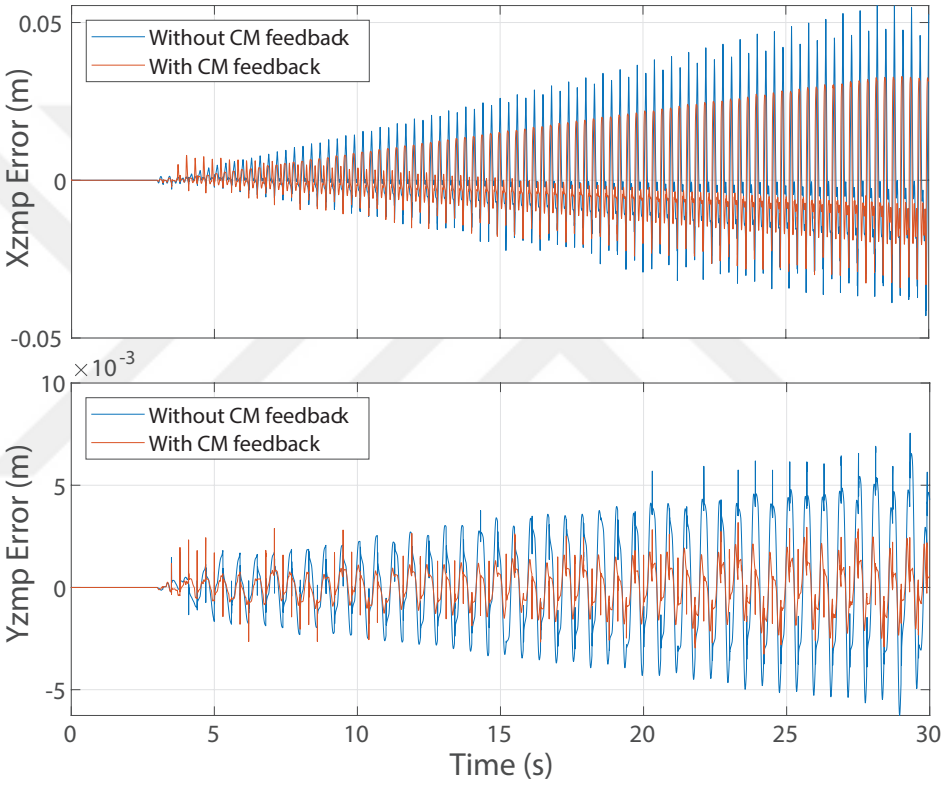

Figure 35: ZMP error without payload.

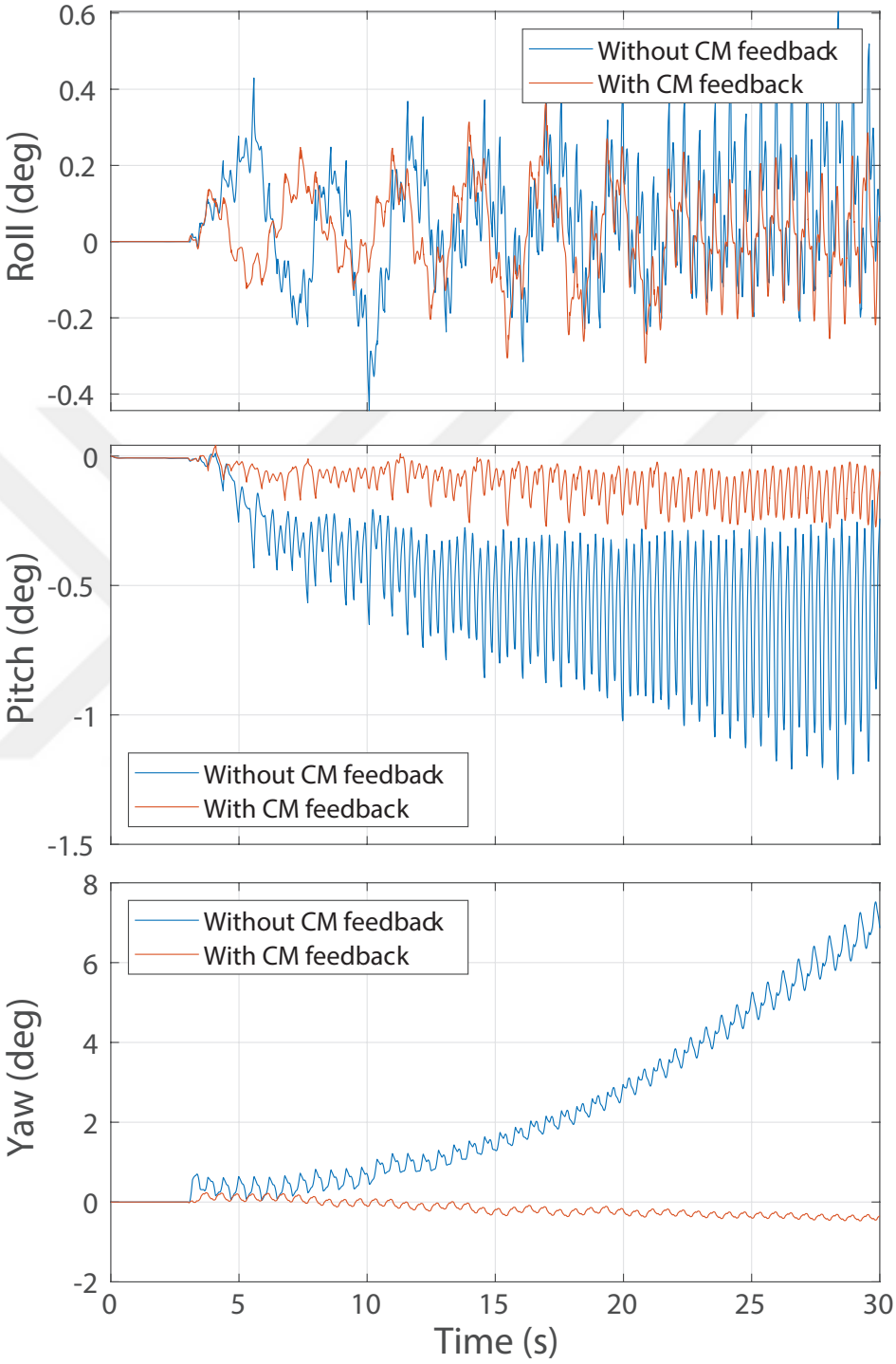

Figure 36: Torso orientation with 4 kg payload.

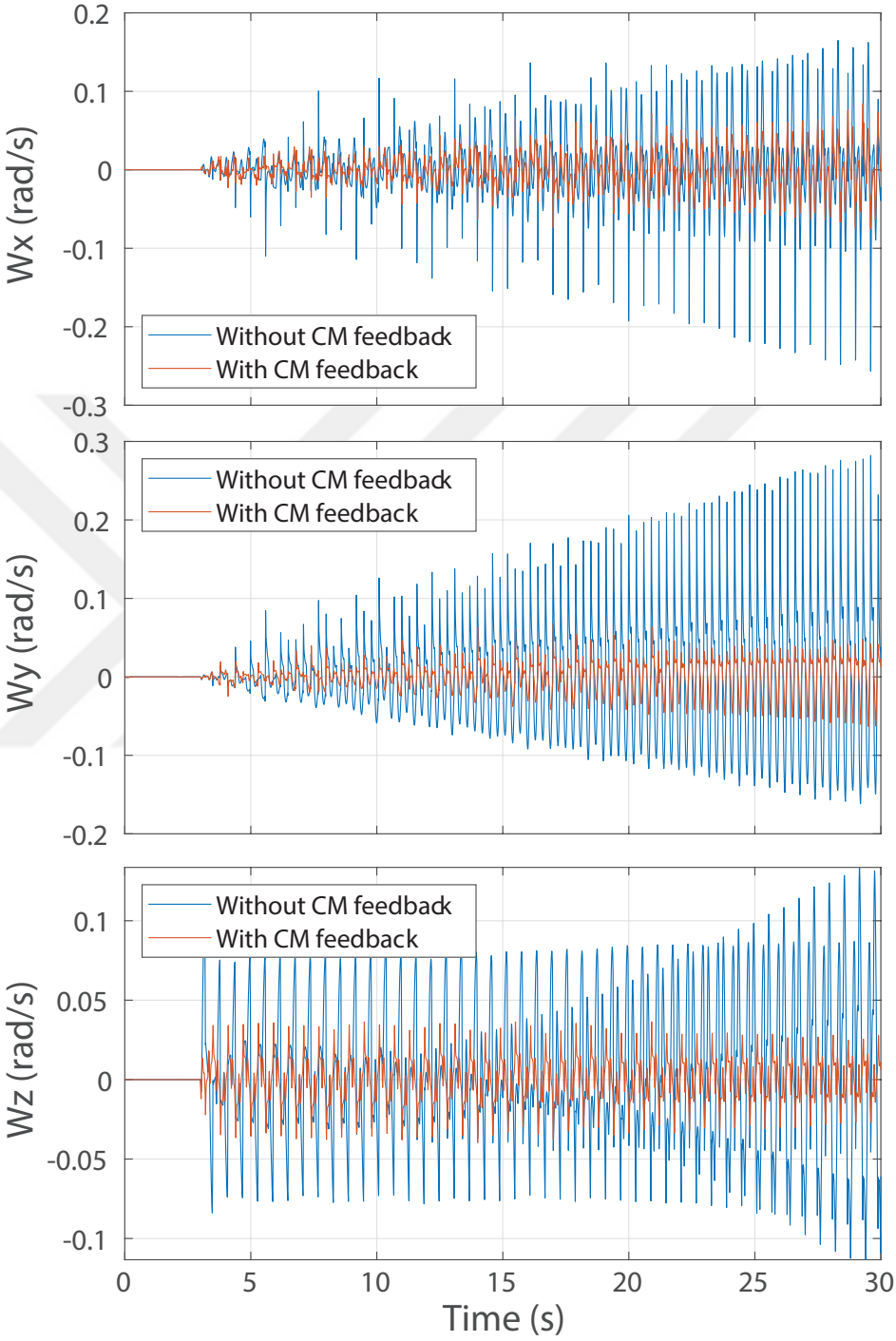

Figure 37: Torso angular velocity with 4 kg payload.

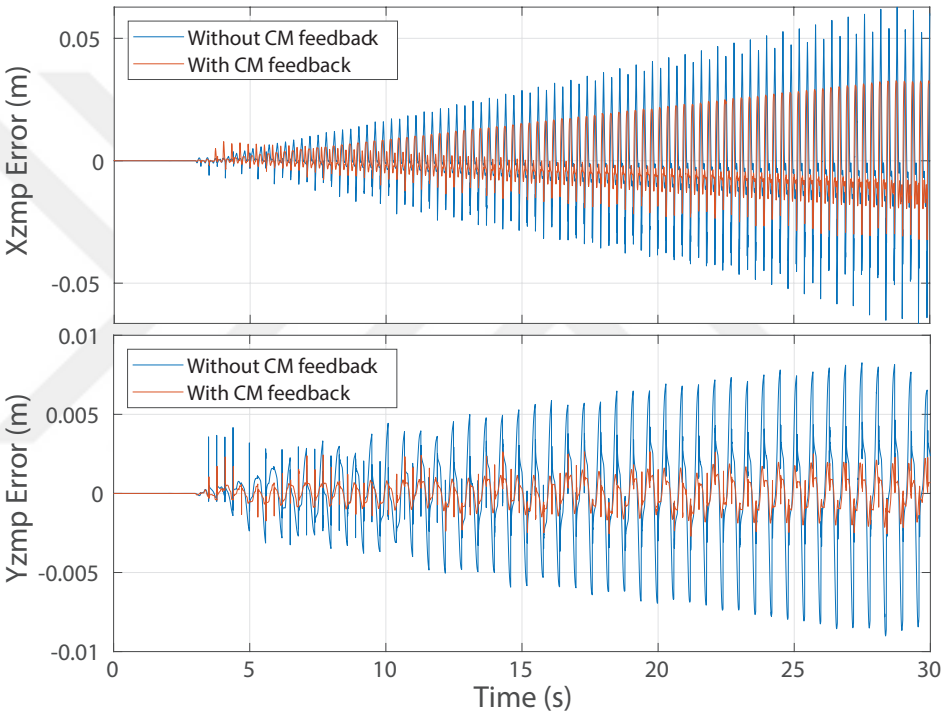

Figure 38: ZMP error with 4 kg payload.

CHAPTER V

CONCLUSION

This thesis presents a robust locomotion control framework for a quadruped robot. In this study, the implementation of trajectory planning for smooth and continuous COM and feet trajectories, the dynamic model of a quadruped robot, the primary control framework, and contact force distribution were investigated.

The component of the locomotion control framework were constructed and tested in the simulation environment, then implemented to the actual quadruped robot. The trajectory planning algorithm of the robot was constructed based on the ZMP concept in order to generate continuous and dynamically balanced COM trajectories. The foot trajectories of the robot were generated in accordance with to the gait pattern, which is constructed for trot-walking. These trajectories were used as reference for the robot to track. The foot trajectory tracking is achieved by computed torque control method, which consists of a PD controller and a feedforward loop with inverse dynamics. The inverse dynamic model of the robot was constructed with a recursive Newton-Euler algorithm which computes required joint torques for given trajectory. Basic controller framework was constructed by combining trajectory planning and computed torque control. Apart from that, the capture point method was used for the robot to preserve its dynamic balance state against external perturbation. Moreover, contact force distribution via centroidal momentum feedback was tested in the simulation environment, which uses VMC to compute desired contact forces.

The locomotion control framework was first tested in simulation, and then, experimental tests were conducted on the actual robot. Both in the simulation and experimental tests, the robot executed trot-walking with a forward velocity of 0.5 $\frac{m}{s}$. The result of both the simulation and experimental test were similar

to each other. The trajectory planner generates a feasible COM trajectory with continuous COM velocity and acceleration profiles. Furthermore, the locomotion control framework was able to execute dynamically balanced locomotion via generated COM trajectory by keeping the ZMP position inside the support polygon. Besides that, RMS errors of joint trajectory tracking of experimental data were similar to simulation data, as expected. Both in simulation and experimental tests, the highest RMS error occurs at the knee joint due to the impact force on the feet, and RMS errors decrease in the hip joints. Joint tracking RMS errors in experimental data are more than joint tracking RMS errors in simulation. This may be caused due to the uncertainties in the inverse dynamic model.

The second experiment was trot-walking with a 20 kg payload. The robot was able to preserve its dynamically balanced state while trot-walking with 0.2 $\frac{m}{s}$ mean forward velocity and an additional 20 kg payload. Endurance of the robot was also tested via an endurance test, and the robot was able to preserve its dynamically balanced state with a fully charged battery, 0.2 $\frac{m}{s}$ mean velocity, and 5.5 kg payload for one hour. The balance of the robot was preserved with the capture point method, and experimental results of the balance test verified the fact that robot preserve its balance by utilizing a step when there are external perturbation.

Finally, simulation results of the centroidal momentum feedback controller showed that the proposed contact force distribution method is able to decrease torso fluctuation and regulates the heading of the robot, and also reduce the ZMP error. In order to implement this controller on robot hardware, floating base linear and angular velocities are needed. The angular velocity of the floating base can be obtained from IMU; however, IMU only measures linear acceleration. To overcome this issue, a state estimation algorithm could be used to obtain floating base linear velocities. In our future work, the proposed controller will be implemented in our quadruped robot.

To conclude, the locomotion control framework is constructed for a quadruped

robot. The control algorithm is tested on both the simulation environment and the quadruped robot, which is built and designed in our laboratory.

APPENDIX A

WALKING SIMULATION JOINT TRACKING RESULTS

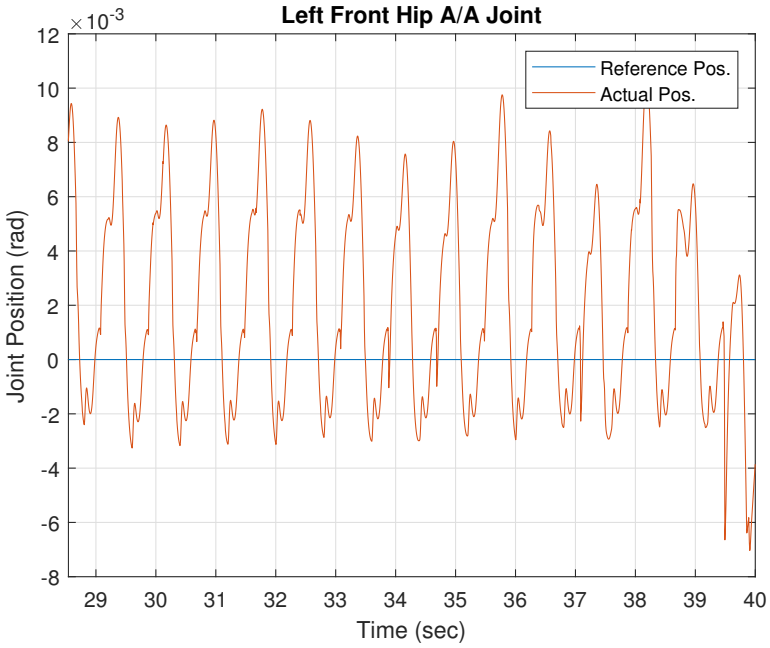

Figure 39: Joint tracking of hip A/A joint of left front leg.

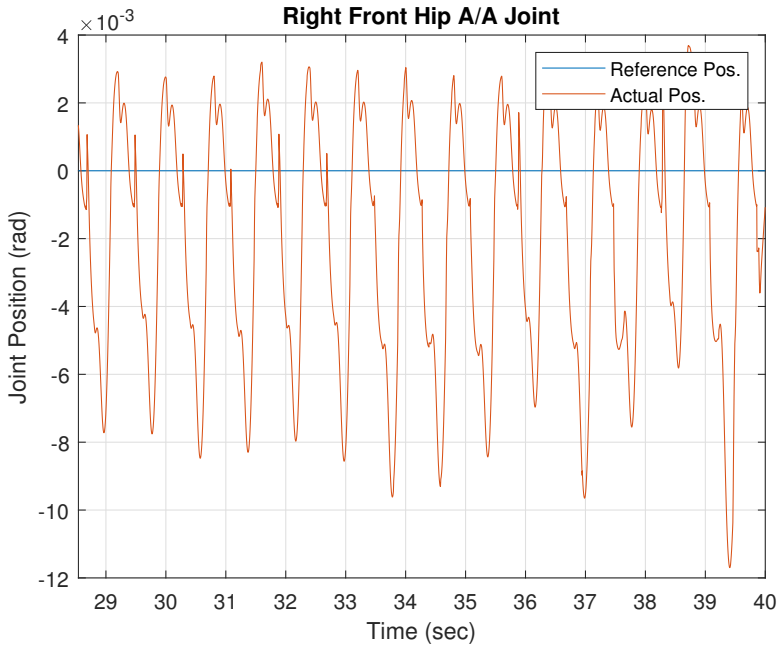

Figure 40: Joint tracking of hip A/A joint of right front leg.

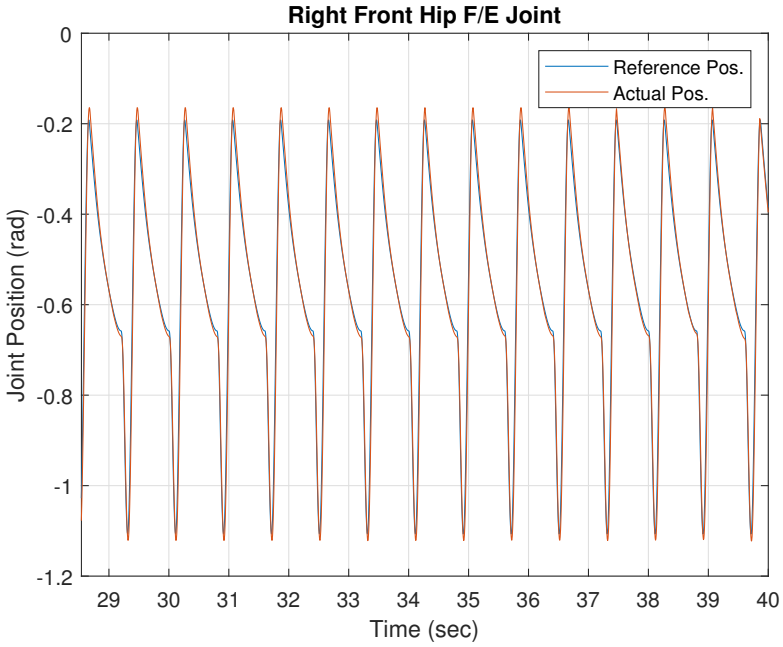

Figure 41: Joint tracking of hip F/E joint of right front leg.

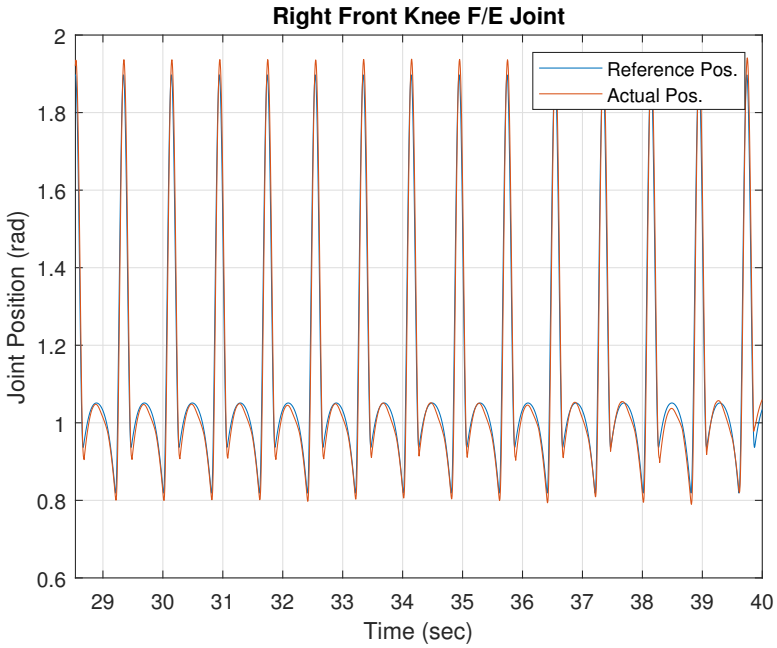

Figure 42: Joint tracking of knee F/E joint of right front leg.

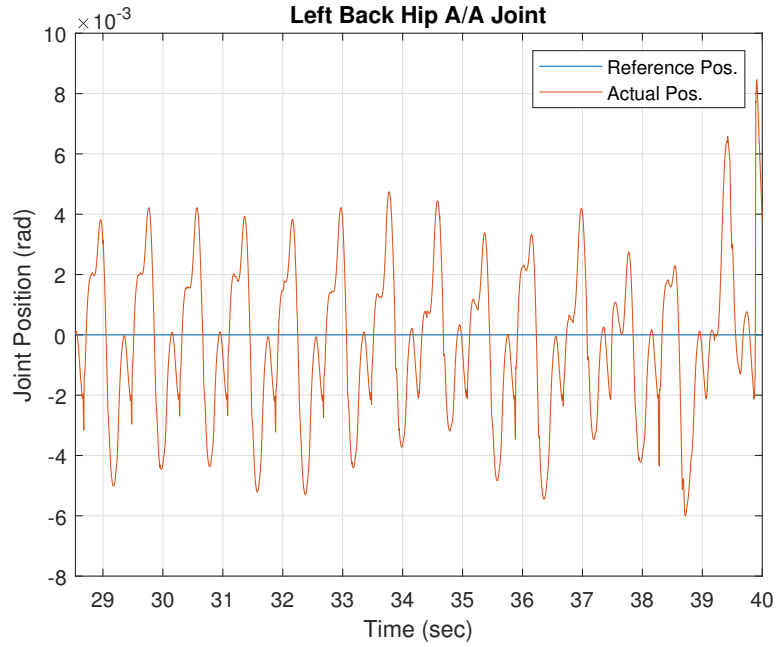

Figure 43: Joint tracking of hip A/A joint of left back leg.

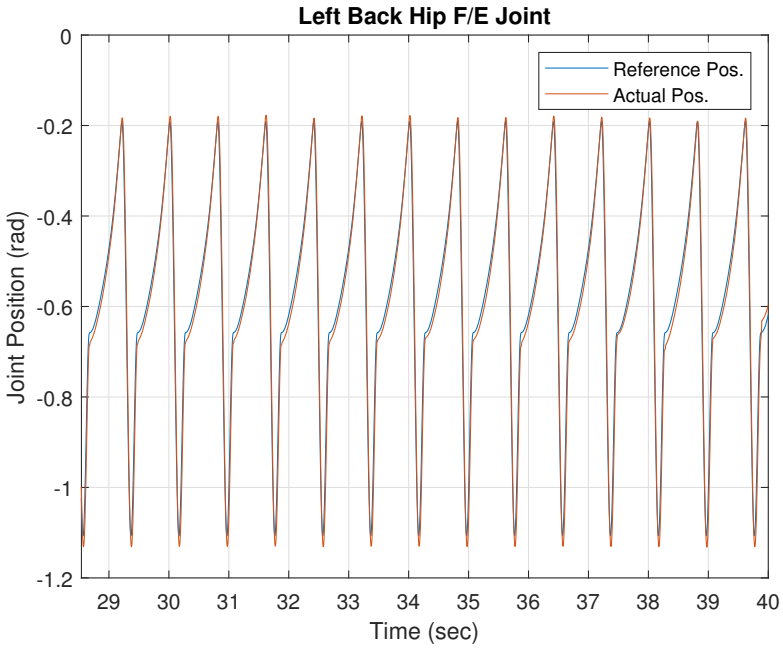

Figure 44: Joint tracking of hip F/E joint of left back leg.

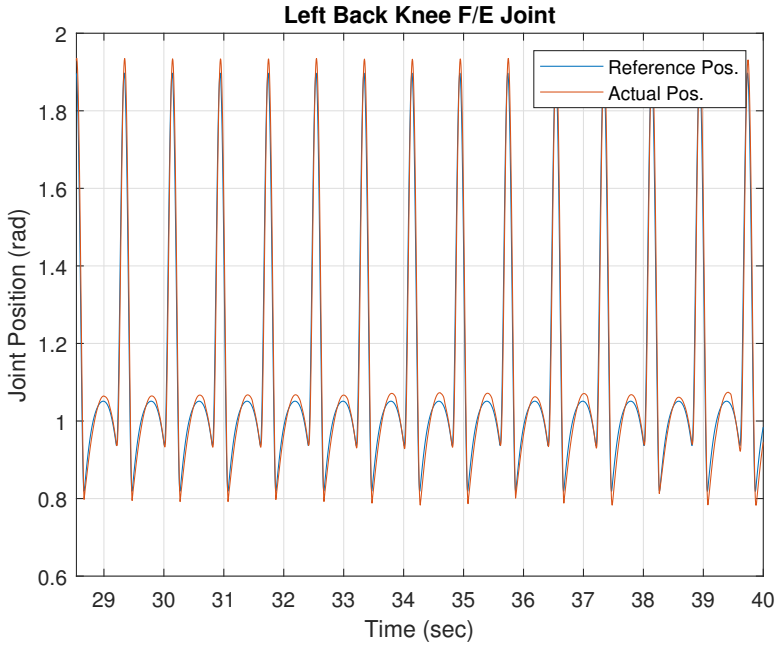

Figure 45: Joint tracking of knee F/E joint of left back leg.

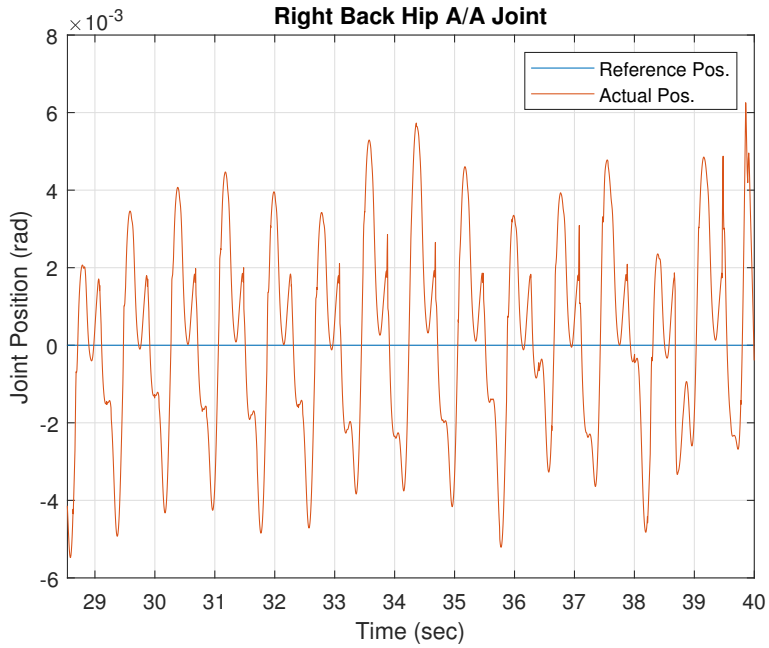

Figure 46: Joint tracking of hip A/A joint of right back leg.

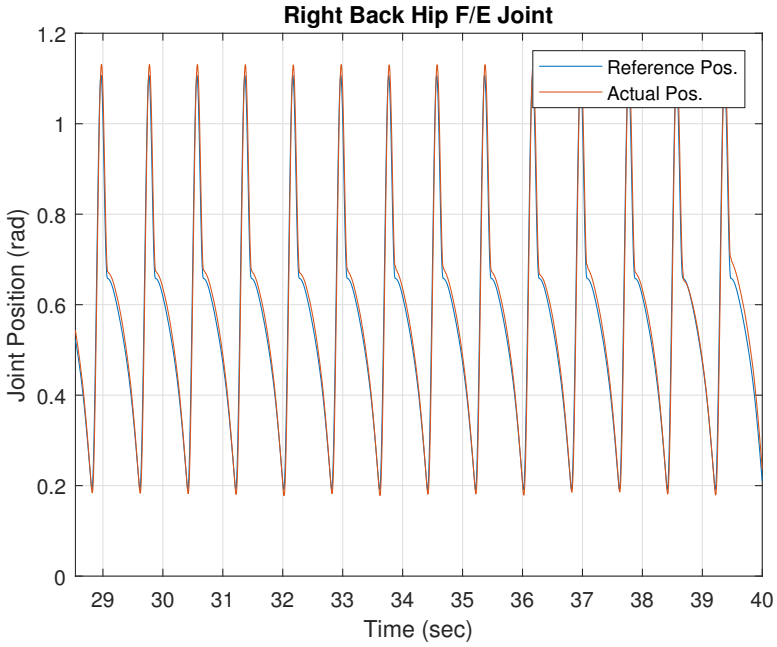

Figure 47: Joint tracking of hip F/E joint of right back leg.

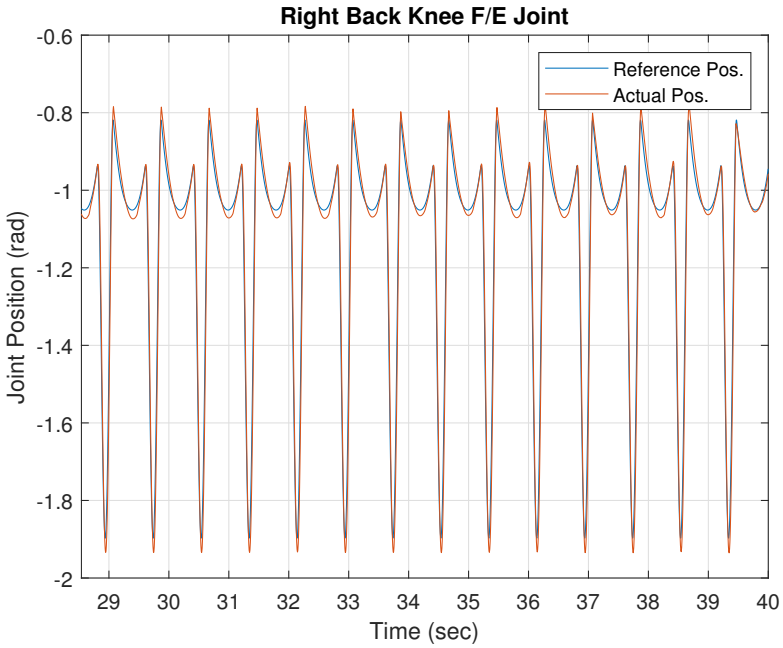

Figure 48: Joint tracking of knee F/E joint of right back leg.

APPENDIX B

WALKING EXPERIMENT JOINT TRACKING RESULTS

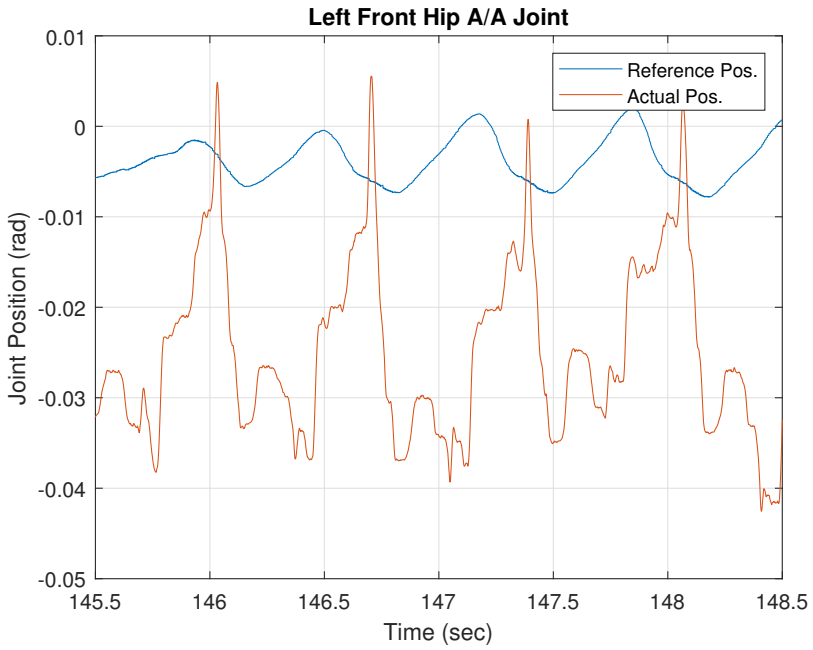

Figure 49: Joint tracking of hip A/A joint of left front leg.

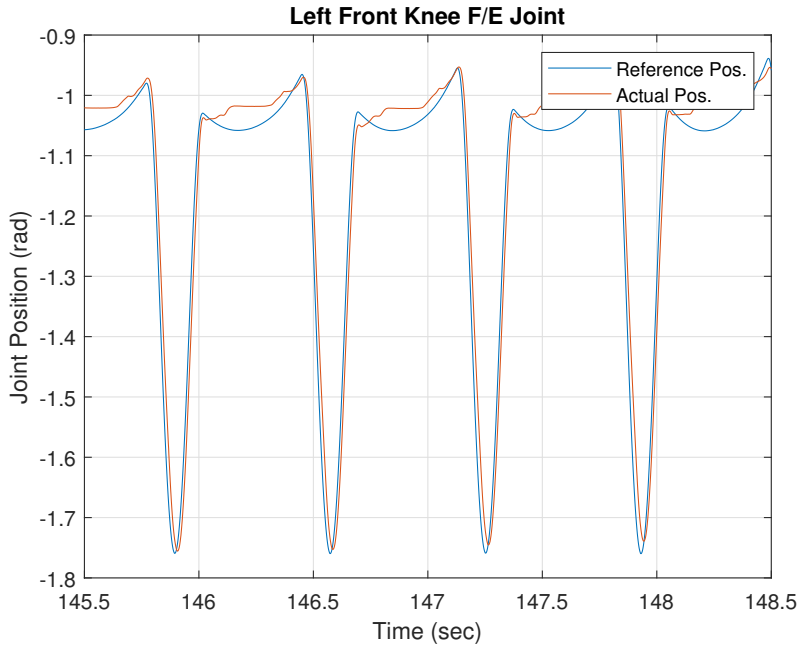

Figure 50: Joint tracking of knee F/E joint of left front leg.

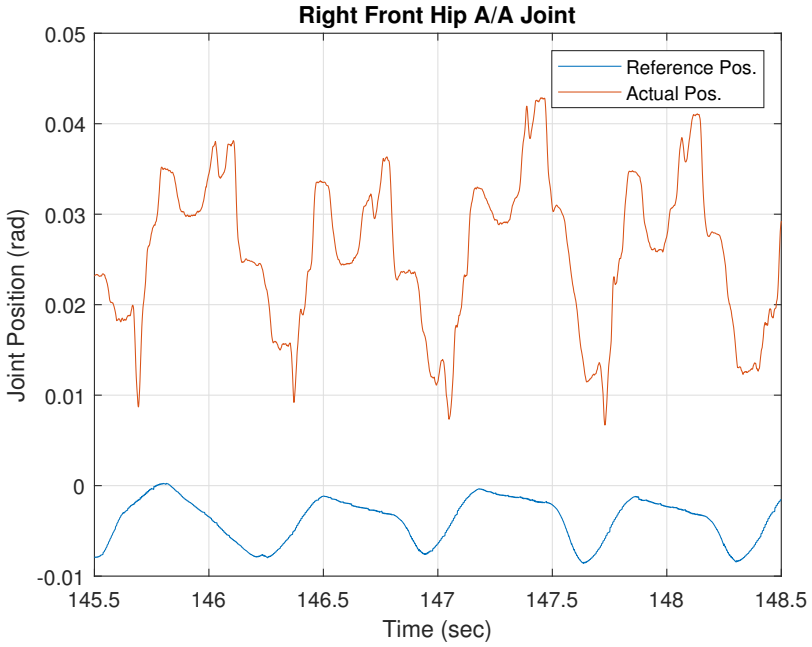

Figure 51: Joint tracking of hip A/A joint of right front leg.

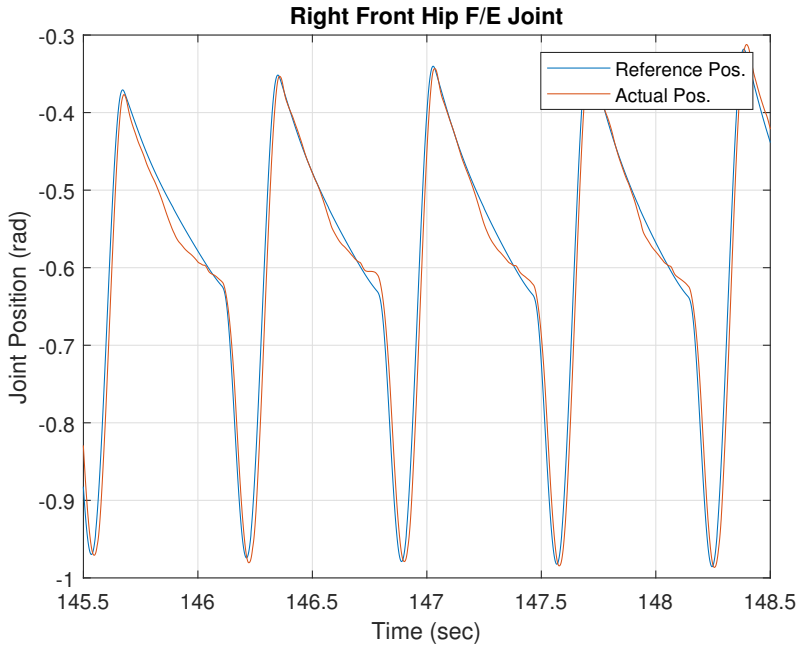

Figure 52: Joint tracking of hip F/E joint of right front leg.

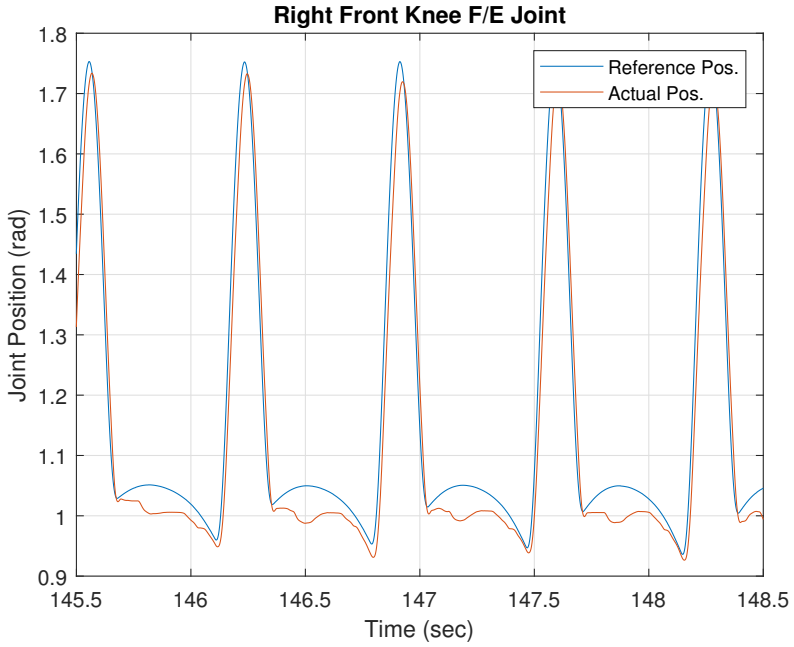

Figure 53: Joint tracking of knee F/E joint of right front leg.

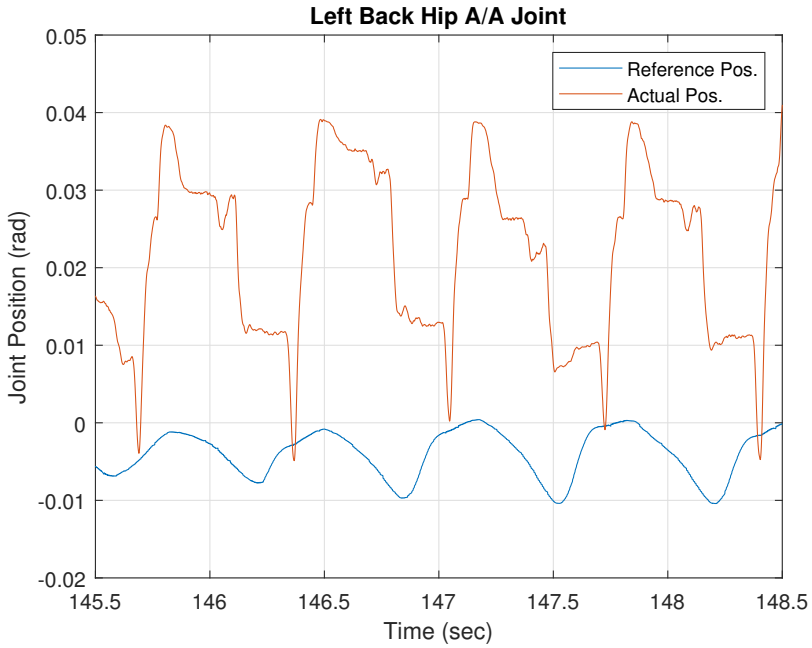

Figure 54: Joint tracking of hip A/A joint of left back leg.

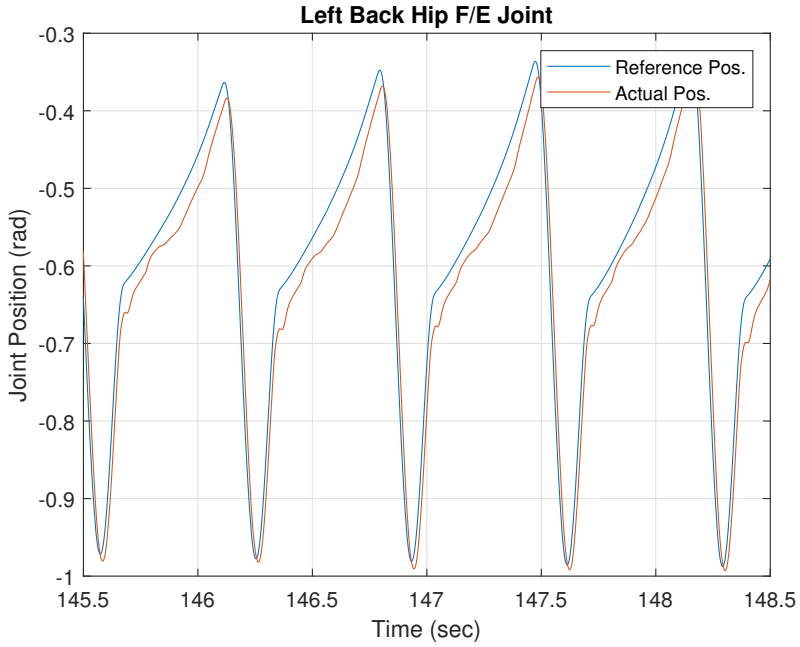

Figure 55: Joint tracking of hip F/E joint of left back leg.

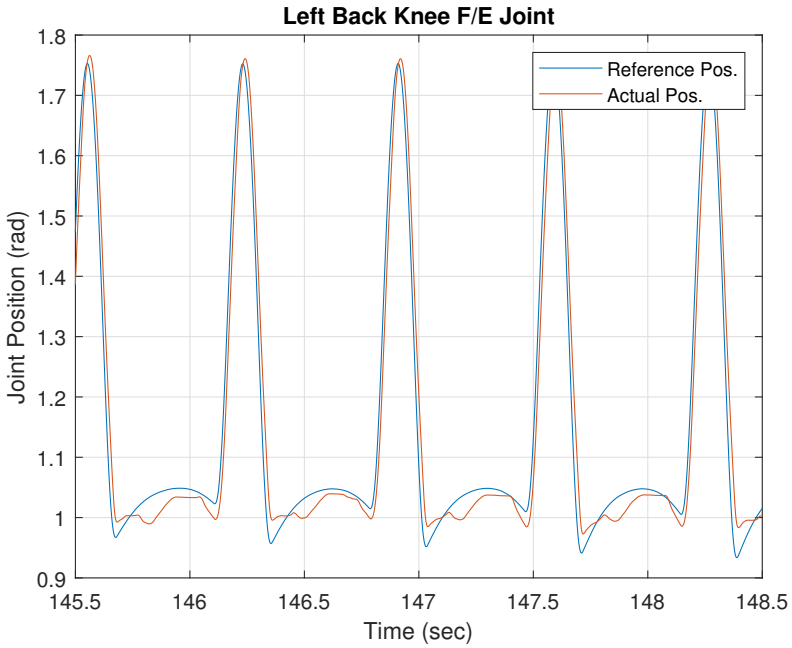

Figure 56: Joint tracking of knee F/E joint of left back leg.

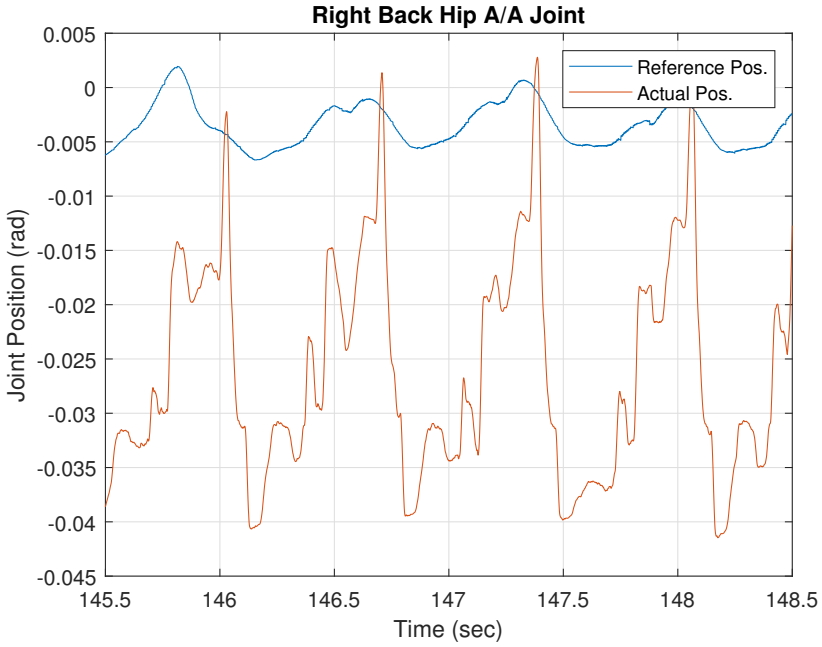

Figure 57: Joint tracking of hip A/A joint of right back leg.

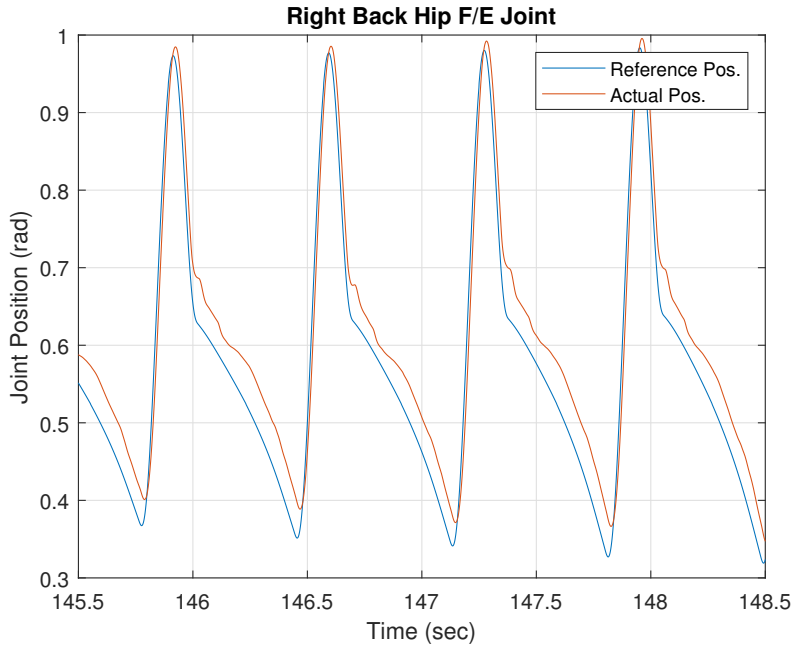

Figure 58: Joint tracking of hip F/E joint of right back leg.

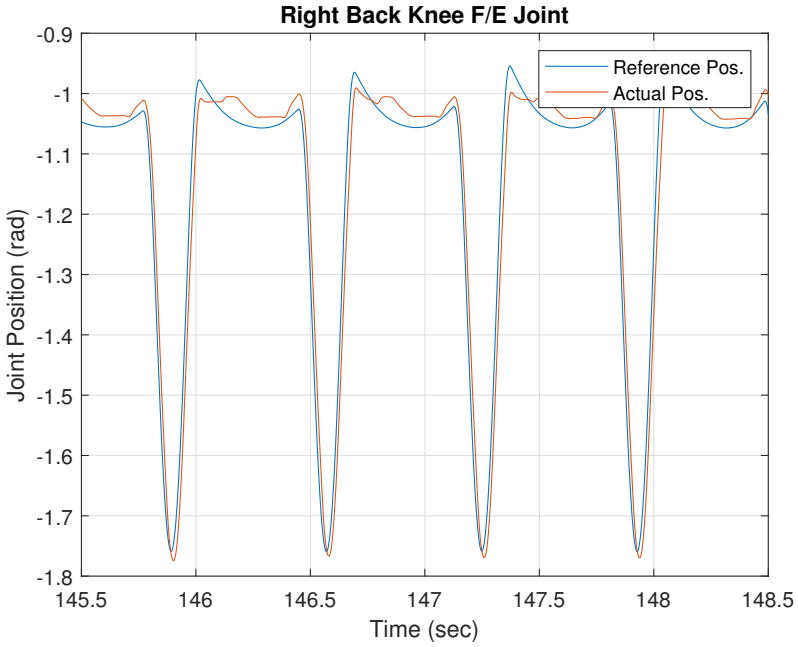

Figure 59: Joint tracking of knee F/E joint of right back leg.

REFERENCES

[1] J. A. T. Machado and M. F. Silva, "An overview of legged robots," 2006.

[2] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krusi, R. Siegwart, and M. Hutter, "Navigation planning for legged robots in challenging terrain," in *RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2016.

[3] L. Wellhausen and M. Hutter, "Rough terrain navigation for legged robots using reachability planning and template learning," in *RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Sept. 2021.

[4] Q. Nguyen, M. J. Powell, B. Katz, J. D. Carlo, and S. Kim, "Optimized jumping on the MIT cheetah 3 robot," in *International Conference on Robotics and Automation (ICRA)*, IEEE, May 2019.

[5] H.-W. Park, P. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *Robotics: Science and Systems XI*, Robotics: Science and Systems Foundation, July 2015.

[6] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10822–10825, 2008. 17th IFAC World Congress.

[7] E. Guizzo, "By leaps and bounds: An exclusive look at how boston dynamics is redefining robot agility," *IEEE Spectr.*, vol. 56, pp. 34–39, Dec. 2019.

[8] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "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.

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

[10] M. Hutter, C. Gehring, M. Bloesch, M. A. Hoepflinger, C. D. Remy, and R. Siegwart, "StarlETH: A Compliant Quadrupedal Robot for Fast, Efficient, and Versatile Locomotion," in *Adaptive Mobile Robotics*, pp. 483–490, World Scientific, July 2012.

[11] S. Rutishauser, A. Sprowitz, L. Righetti, and A. J. Ijspeert, "Passive compliant quadruped robot using central pattern generators for locomotion control," Oct. 2008.

[12] E. CanOzcinar, O. Bebek, and B. Ugurlu, "Contact force distribution using centroidal momentum feedback for quadruped locomotion," *Biological Cybernetics*, vol. 107, pp. 309–320, Mar. 2013.

[13] M. Vukobratović and B. Borovac, "Zero-Moment Point — Thirty Five Years of Its Life," *International Journal of Humanoid Robotics*, vol. 01, pp. 157–173, Mar. 2004.

[14] B. Ugurlu, I. Havoutis, C. Semini, K. Kayamori, D. G. Caldwell, and T. Narikiyo, "Pattern generation and compliant feedback control for quadrupedal dynamic trot-walking locomotion: experiments on RoboCat-1 and HyQ," *Autonomous Robots*, vol. 38, pp. 415–437, Feb. 2015.

[15] B. Ugurlu, J. A. Saglia, N. G. Tsagarakis, S. Morfey, and D. G. Caldwell, "Bipedal hopping pattern generation for passively compliant humanoids: Exploiting the resonance," *IEEE Transactions on Industrial Electronics*, vol. 61, pp. 5431–5443, Oct. 2014.

[16] R. Kurazume, S. Hirose, and K. Yoneda, "Feedforward and feedback dynamic trot gait control for a quadruped walking vehicle," in *Proceedings ICRA. International Conference on Robotics and Automation (Cat. No.01CH37164)*, IEEE, 2001.

[17] K. Yoneda, H. Iiyama, and S. Hirose, "Intermittent trot gait of a quadruped walking machine dynamic stability control of an omnidirectional walk," in *Proceedings of International Conference on Robotics and Automation*, IEEE, 1996.

[18] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. D. Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," in *International Conference on Robotics and Automation*, IEEE, May 2013.

[19] 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*, vol. 36, pp. 1635–1648, Dec. 2020.

[20] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," 2019.

[21] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 236–258, 2011.

[22] S. Emre, "Design, development, and real-life implementation of a high power quadruped robot," Master's thesis, Ozyegin University, 8 2022.

[23] D. Jansen and H. Buttner, "Real-time ethernet: the EtherCAT solution," *Computing and Control Engineering*, vol. 15, pp. 16–21, Feb. 2004.

[24] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.

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

[26] Craig, *Introduction to robotics: Pearson new international edition*. London, England: Pearson Education, 3 ed., Nov. 2013.

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

[28] F. L. Moro, A. Spröwitz, A. Tuleu, M. Vespignani, N. G. Tsagarakis, A. J. Ijspeert, and D. G. Caldwell, "Horse-like walking, trotting, and galloping derived from kinematic motion primitives (kMPs) and their application to walk/trot transitions in a compliant quadruped robot," *IEEE International Conference on Mechatronics*, pp. 309–320, Mar. 2023.

[29] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *The International Journal of Robotics Research*, vol. 20, pp. 129–143, Feb. 2001.

[30] C. D. Remy, M. Hutter, M. Hoepflinger, M. Bloesch, C. Gehring, and R. Siegwart, "Quadrupedal robots with stiff and compliant actuation," *Automatisierungstechnik*, vol. 60, pp. 682–691, Nov. 2012.

[31] P. M. Wensing and D. E. Orin, "Improved computation of the humanoid centroidal dynamics and application for whole-body control," *International Journal of Humanoid Robotics*, vol. 13, p. 1550039, Mar. 2016.

[32] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, pp. 161–176, June 2013.

VITA

Erim Can Özçınar had taken high school education in Marmara Collage. He received his Bachelor's Degree in Mechanical Engineering at Ozyegin University in 2019. He had worked as an undergraduate research student at OzU Biomechatronics Laboratory for one year with Asst. Prof. Barkan Uğurlu. He is a M.Sc. Student in Özyeğin University under the supervision of Assist. Prof. Barkan Uğurlu. He is studying control systems, robust locomotion control, dynamic modeling and robotics.