# A HYBRID PLANNING AND CONTROL MODEL FOR BIPED FEET ROTATION

BY

KAIYI FU

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science in Electrical Engineering
in the Undergraduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Seth Hutchinson

# ABSTRACT

This thesis proposes methods for biped walking locomotion with feet rotation. The chief objective of this work is to first generate a guide trajectory based on designing a zero moment point (ZMP) trajectory within the support polygon and obtain linear controlling methods to stabilize the walking procedure with feet rotation. With feet rotation, the walking procedure will be more humanlike, more flexible and possible saving energy. However, when the feet are rotating around their edge, either toe or heel, the entire robot is under-actuated which are more difficult to control. By using preview control, a dynamic model of the system can be derived to control the robot.

This thesis is based upon a simplified model of the Reemc Robot by PAL Robotics. The simplified model has fixed arms, since only leg motions are considered, and two legs. Each leg has three degrees of freedom. The robot is presented as a three mass model. A guided gait trajectory is first generated as the boundary condition for the ZMP. Interpolation methods are used to generate a ZMP trajectory from a set of discrete points that stay inside the boundary condition. By designing the transition model from single support phase and double support phase, a general schema can be achieved. Following the assumptions of a linear inverted pendulum, trajectories of all three masses can be solved. Inverse kinematics can now give the reference joint trajectories, which, together with the reference ZMP trajectory, is used in control methods to minimize the error between the reference trajectory and actual trajectory in simulation.

Control methods are used to stabilize the motion of the walking procedure. Preview control is used for the single support phase where the behavior of all three masses is linear. A proper input can be obtained through optimization. During the double support phase, the feet rotations are nonlinear and under-actuated since the feet are rotating around their edge where no torque can be produced from the ground. By using preview control, an input can be

applied to the robot so that the robot can maintain dynamic stability.
***Keywords***— biped locomotion, feet rotation, preview control, zmp

*To my parents, for their love and support.*

# ACKNOWLEDGMENTS

I would first like to thank my advisor, Professor Seth Hutchinson, for guiding my research in recent semesters. Working with him has given me an opportunity to learn all aspects about robotics. I would also like to thank Hilario Tome, Adria Rogue and many other colleges at PAL Robotics for giving me much help in doing research on Reemc based projects.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ZMP | Zero moment point |
| COM | Center of mass |
| COF | Center of foot |
| IMU | Inertial measurement unit |
| FTS | Force torque sensor |
| ROS | Robot operating system |
| DOF | Degree of freedom |
| IPOPT | Interior point optimization |
| DH | Denavit—Hartenberg |
| POG | Periods of Gaits |
| DSR | Ratio of double support to $t_{\mathrm{s}}$ |
| $M_{\mathrm{foot}}$ | Sum of each foot mass, which concentrates at the center of the foot |
| $M_{\mathrm{pend}}$ | Mass of the trunk, which can also be referred as the mass of the inverted pendulum |
| $M_{\mathrm{total}}$ | Total mass of the robot ($M_{\mathrm{total}} = M_{\mathrm{pend}} + 2M_{\mathrm{foot}}$) |
| $x_{\mathrm{pend}}, y_{\mathrm{pend}}, z_{\mathrm{pend}}$ | The position of the inverted pendulum in world frame, i.e., the position of the trunk |
| $x^{\mathrm{L}}_{\mathrm{toe}}, y^{\mathrm{L}}_{\mathrm{toe}}, z^{\mathrm{L}}_{\mathrm{toe}}$ | Position of the toe of the left foot in world frame |
| $x^{\mathrm{R}}_{\mathrm{toe}}, y^{\mathrm{R}}_{\mathrm{toe}}, z^{\mathrm{R}}_{\mathrm{toe}}$ | Position of the toe of the right foot in world frame |
| $x^{\mathrm{L}}_{\mathrm{heel}}, y^{\mathrm{L}}_{\mathrm{heel}}, z^{\mathrm{L}}_{\mathrm{heel}}$ | Position of the heel of the left foot in world frame |
| $x^{\mathrm{R}}_{\mathrm{heel}}, y^{\mathrm{R}}_{\mathrm{heel}}, z^{\mathrm{R}}_{\mathrm{heel}}$ | Position of the heel of the right foot in world frame |

| | |
|---|---|
| $x^{\mathrm{L}}_{\mathrm{cof}}, y^{\mathrm{L}}_{\mathrm{cof}}, z^{\mathrm{L}}_{\mathrm{cof}}$ | Position of the center of foot of the left foot in world frame |
| $x^{\mathrm{R}}_{\mathrm{cof}}, y^{\mathrm{R}}_{\mathrm{cof}}, z^{\mathrm{R}}_{\mathrm{cof}}$ | Position of the center of foot of the right foot in world frame |
| $l_{\mathrm{step}}$ | Step length of the robot |
| $l_{\mathrm{t}}$ | Length of the thigh |
| $l_{\mathrm{s}}$ | Length of the shank |
| $w$ | Width of the robot, i.e., the distance between the two foot centers in sagittal direction |
| $l_{\mathrm{f}}$ | Length of the foot and the width of the foot |
| $t_{\mathrm{s}}$ | Time required to finish a step |
| $t_{\mathrm{ss}}$ | Time when the robot is in single support state for each step |
| $t_{\mathrm{ds}}$ | Time when the robot is in double support for each step |
| $h$ | Height of the inverted pendulum, i.e., the height of the trunk. |
| $g$ | Gravitational constant |
| $x_{\mathrm{zmp}}, y_{\mathrm{zmp}}$ | Position of the ZMP in world frame |
| $x_{\mathrm{COM}}, y_{\mathrm{COM}}, z_{\mathrm{COM}}$ | Position of the Center Of Mass for the entire robot |
| $\theta^{\mathrm{L}}_{\mathrm{h}}, \theta^{\mathrm{L}}_{\mathrm{k}}, \theta^{\mathrm{L}}_{\mathrm{a}}$ | Angle of the hip, knee and ankle of the left-foot links |
| $\theta^{\mathrm{R}}_{\mathrm{h}}, \theta^{\mathrm{R}}_{\mathrm{k}}, \theta^{\mathrm{R}}_{\mathrm{a}}$ | Angle of the hip, knee and ankle of the right-foot links |
| $\eta$ | Heel—to—ankle ratio which is the ratio of the length from heel to ankle to that of the foot |
| $\vec{F}_{\mathrm{left}}$ | Force at the ankle of the left foot |
| $\vec{F}_{\mathrm{right}}$ | Force at the ankle of the right foot |
| $\vec{\tau}_{\mathrm{left}}$ | Torque at the ankle of the left foot |
| $\vec{\tau}_{\mathrm{right}}$ | Torque at the ankle of the right foot |

# CHAPTER 1

# INTRODUCTION

Humanoid robots are a popular topic nowadays since they are similar to humans in many aspects. Due to high level of similarity, humanoid robots can replace humans to do some dirty work that requires less intelligence and more on labor. These tasks are repetitive so that a designed algorithm can be formulated. Also, thanks to the development of computation technology, the computation speed has improved so much that real-time planning is now possible, which was impossible for many researchers decades ago.

The research on biped robots began decades ago, yet the walking of biped robots is still a tough problem. Such bipedal robots are dynamically unstable when they walk since they can easily tip over during stepping. The support polygon, which is defined by the convex hull of the footprints, is so small. Humans can leverage the stability easily while the robots cannot. On example could be the Assimo robot produced by Honda in Japan. With a complicated controlling mechanical system which has been developed for many years, it can easily walk; however, it cannot jump or do more complicated behaviors that humans do.

At the very beginning, bipedal walking can be achieved without using any control methods, yet this has a lot to do with the precise walking condition [1]. Even if one condition is not satisfied, the walking procedure cannot proceed. However, by adding control methods, researchers can maneuver the robot in more complicated environments, and even with disturbance [2,3]. Numerous methods have emerged to control the robot. Some are linear controlling methods such as preview control [4] and capture point control [5]. Preview control is also used in many other areas [6–9]. Others are some non-linear controlling methods such as hybrid Zero dynamics [10]. Others using different control methods such as divergent component of motion [11]. The most important breakthrough can be traced back to the birth of zero moment point (ZMP) [12, 13]. The concept of ZMP is so popular that many biped

robot locomotions are designed upon using the concept of ZMP, including Assimo [14, 15] and HRP-2 [16]. Works have been proposed to control ZMP [17]. However, ZMP has many restrictions [18]. Generally speaking, ZMP can only be applied to environments that have the following conditions: non-slipping, flat ground and hands are not touching the environment. In other words, ZMP should only be applied to situations where the motion of centers of mass are 2D, i.e., stay inside parallel planes. Other works have also tried to extend ZMP into more applications such as jumping or running [19].

There are other planning methods that can achieve far more than ZMP. Some methods focus on multi-contact planning where the robot can touch the environment with both hands and feet [20, 21]. There are also some other methods that involve feet rotation [22]. These methods aim to achieve control methods that are more suitable in the human environment. Since our environment is complicated, unknown to the robot, we have to design different strategies for such an environment. Several papers propose to solve planning methods in cluttered environments with multicontact planning [23–25]. Simultaneously, researchers are also trying to use the same method that human use when facing a new environment–learning. Like humans, robots may use past experience to solve the control problems in a new environment [26, 27].

Discussed so far, most locomotion methods, though successful, focus on touching the ground with the entire foot. The walking procedure involves lifting and lowering the foot entirely and vertically. However, when a human walks, feet rotation is always involved because it makes the walking more flexible. Most locomotion methods ignore this aspect. Feet rotation is a normal characteristic of human walking, which should also be added to robot walking locomotion. Yet, this idea is very hard to achieve since in most scenarios for feet rotation, the robot is under-actuated since there is no torque exerted from the ground when the feet is rotating around heel or toe. This locomotion is very hard to keep balanced. Proposed solutions to this problem focused on dynamical balance and control systems [28]. A very closely related solution introduces the hybrid zero dynamic model for feet multicontact [1, 22].

This thesis proposes a new simplified model for feet rotation locomotion. This simplified model leverages a hybrid planning and control systems. Decomposing the walking procedure into two phases, single support phase and

double support phase, we did analysis for both states and the transitions between different states. A three-mass model, as proposed in [29], will be used to represent the simplified model for the robot, where two masses are concentrated at the center of the feet and the third mass concentrates at the trunk of the robot. Such modeling will generate modelling error that is less than using the center of mass (COM) only; however, the modelling error can still be significant. Thus, controlling methods are applied to compensate for such error.

The overall idea is developed upon ZMP control. ZMP refers to zero moment point where the total of horizontal inertia and gravity forces equal to zero [13]. The only goal for ZMP in this thesis is to keep the robot in stable states, which can be tracked upon whether the ZMP stays inside the support polygon [30], which is the convex hull of the robot's footprints. For example, when the biped robot is in double support phase, the support polygon is the horizontal region bounded by the outside edges of both feet, where the ZMP must stay to keep balance. An F-ZMP describes the ZMP when the robot is unstable [31] and will always stay at the edge of the support polygon. F-ZMP only exists when the robot is unstable and ZMP only exists when the robot is stable. The discussion of F-ZMP is out of the scope of this thesis and will not be discussed in the following passages.

As the planning part uses ZMP as the criterion for dynamical balance, control methods can be proposed to minimize the error between the desired ZMP trajectory and actual ZMP trajectory. A hybrid control model will be used in the control system. During linear phases, the preview control will be used. Preview control has been broadly used. Preview control is to use future information, such as desired ZMP trajectory, and current state, such as actual ZMP trajectory, to generate current input to the dynamical system to minimize the error between the actual ZMP trajectory and desired ZMP trajectory. The input will be generated through optimization.

The thesis is organized as follows. **Chapter 2** gives a basic formulation of the ZMP concept, nomenclatures and the modelling of the robot. It will also introduce different states or phases the robot will experience through walking. **Chapter 3** focuses on ZMP planning and interpolation. An analysis is given of the mathematical models for ZMP interpolation. **Chapter 4** focuses on the transition model between single support phase and double support phase. This transition provides the boundary condition for generat-

ing center of mass trajectories. **Chapter 4** also solves for the center of mass trajectories for all three masses. **Chapter 5** introduces and analyzes our hybrid control systems. **Chapter 6** covers simulation techniques and some other technical methods we used during the simulation. **Chapter 7** presents the simulation result for a biped walking robot achieved using the previous methods. **Chapter 8** details the conclusion and future work.

# CHAPTER 2

# ROBOT MODELLING

## 2.1 ZMP Equations

A ZMP equation based on multimass (N masses) robot can be presented as follows [31]:

$$x_{\mathrm{zmp}} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{N}} m_{\mathrm{i}}((\ddot{z}_{\mathrm{i}} + g)x_{\mathrm{i}} - z_{\mathrm{i}}\ddot{x}_{\mathrm{i}})}{\sum_{\mathrm{i}=1}^{\mathrm{N}} m_{\mathrm{i}}(\ddot{z}_{\mathrm{i}} + g)} \tag{2.1}$$

$$y_{\mathrm{zmp}} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{N}} m_{\mathrm{i}}((\ddot{z}_{\mathrm{i}} + g)y_{\mathrm{i}} - z_{\mathrm{i}}\ddot{y}_{\mathrm{i}})}{\sum_{\mathrm{i}=1}^{\mathrm{N}} m_{\mathrm{i}}(\ddot{z}_{\mathrm{i}} + g)} \tag{2.2}$$

Here we do not consider $z$ component since the robot is not moving upwards or downs wards. Therefore we can assume that $p_{\mathrm{zmp}} = [x_{\mathrm{zmp}} \quad y_{\mathrm{zmp}}]^{\mathrm{T}} \in \mathbb{R}^2$ stays inside a plane. From the equations, we can calculate the ZMP position by knowing the linear acceleration and position of all the bodies of the robot. However, it is extremely difficult to do so for a real robot since each body must have an acceleration sensor, which is impossible in most cases. There is another way to measure the ZMP points which is more practical. This method only requires torque and force sensors at the ankles [32]. The details will be discussed in **Chapter 6**.

## 2.2 Three-Mass Model

To control the heel and toe of a robot, the biped robot must have a degree of freedom at the ankle of each leg. Along with the knee and hip, the simplified model will have six degrees of freedom in configuration space with three joints at each leg. That is, the end effector of each leg, which refers to the center of the foot, has three degrees of freedom. It is allowed to translate in two directions and rotated in one direction. Together with the body of the robot,

which also shares six degrees of freedom in $SE(3)$, the simplified model has twelve degrees of freedom in the workspace.

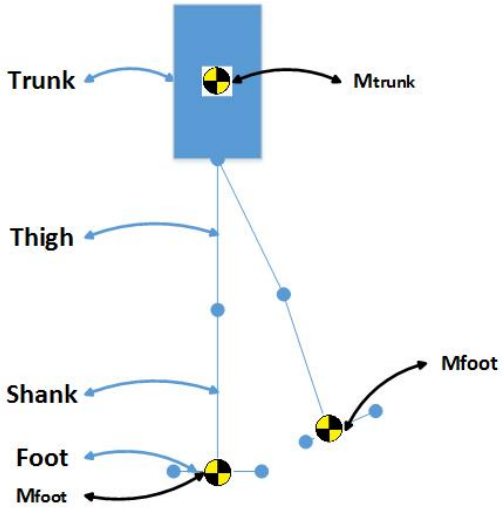



Figure 2.1: Three Mass Model

A three-mass model is applied to model the simplified robot. The three-mass model, first proposed in [29] assumes that a robot can be mathematically modeled as three masses instead of one mass, which had been conventionally assumed in previous research. It follows the assumption of the three-dimensional linearly inverted pendulum mode (3D-LIPM), which has been applied to many biped robot models [33]. In this thesis, the three-mass model is assumed where two masses concentrate at the center of the feet and one mass concentrates at the center of the trunk. The model is shown in figure 2.1. The mass of both feet can be regarded as the sum of the masses of all thighs, shanks and feet. The mass of the trunk is the sum of the masses of the rest of the bodies of the robot. A three-mass model can thus be created. From equations (2.1) and (2.2), the ZMP formula for this model can be derived as the robot has three bodies.

$$x_{\mathrm{zmp}} = \frac{\sum_{\mathrm{i}=1}^{3} m_{\mathrm{i}}((\ddot{z}_{\mathrm{i}} + g)x_{\mathrm{i}} - z_{\mathrm{i}}\ddot{x}_{\mathrm{i}})}{\sum_{\mathrm{i}=\mathrm{i}}^{\mathrm{N}} m_{\mathrm{i}}(\ddot{z}_{\mathrm{i}} + g)} \tag{2.3}$$

$$y_{\mathrm{zmp}} = \frac{\sum_{\mathrm{i}=1}^{3} m_{\mathrm{i}}((\ddot{z}_{\mathrm{i}} + g)y_{\mathrm{i}} - z_{\mathrm{i}}\ddot{y}_{\mathrm{i}})}{\sum_{\mathrm{i}=\mathrm{i}}^{\mathrm{N}} m_{\mathrm{i}}(\ddot{z}_{\mathrm{i}} + g)} \tag{2.4}$$

Now the forward kinematics can be derived according to the model described above. We based our model on the Reemc Robot by Pal Robotics.

We followed the design of the original robot by making the ratio between the length from the toe to ankle and the entire foot length (heel-to-ankle ratio), $\eta$, to be 0.5. So we can achieve the following forward kinematics for hip, toe and center of foot. All positions are expressed in the robot trunk frame, denoted as $P$ frame.

$$\begin{bmatrix} x^P_{\text{toe}} \\ y^P_{\text{toe}} \\ z^P_{\text{toe}} \end{bmatrix} = \begin{bmatrix} l_{\text{t}} \sin(\theta_{\text{h}}) + l_{\text{s}} \sin(\theta_{\text{h}} + \theta_{\text{k}}) \\ +0.5 l_{\text{f}} \cos(\theta_{\text{h}} + \theta_{\text{k}} + \theta_{\text{a}}) \\ \pm \frac{w}{2} \\ -l_{\text{t}} \cos(\theta_{\text{h}}) - l_{\text{s}} \cos(\theta_{\text{h}} + \theta_{\text{k}}) \\ +0.5 l_{\text{f}} \sin(\theta_{\text{h}} + \theta_{\text{k}} + \theta_{\text{a}}) \end{bmatrix} \tag{2.5}$$

$$\begin{bmatrix} x^P_{\text{heel}} \\ y^P_{\text{heel}} \\ z^P_{\text{heel}} \end{bmatrix} = \begin{bmatrix} l_{\text{t}} \sin(\theta_{\text{h}}) + l_{\text{s}} \sin(\theta_{\text{h}} + \theta_{\text{k}}) \\ -0.5 l_{\text{f}} \cos(\theta_{\text{h}} + \theta_{\text{k}} + \theta_{\text{a}}) \\ \pm \frac{w}{2} \\ -l_{\text{t}} \cos(\theta_{\text{h}}) - l_{\text{s}} \cos(\theta_{\text{h}} + \theta_{\text{k}}) \\ -0.5 l_{\text{f}} \sin(\theta_{\text{h}} + \theta_{\text{k}} + \theta_{\text{a}}) \end{bmatrix} \tag{2.6}$$

$$\begin{bmatrix} x^P_{\text{cof}} \\ y^P_{\text{cof}} \\ z^P_{\text{cof}} \end{bmatrix} = \begin{bmatrix} l_{\text{t}} \sin(\theta_{\text{h}}) + l_{\text{s}} \sin(\theta_{\text{h}} + \theta_{\text{k}}) \\ \pm \frac{w}{2} \\ -l_{\text{t}} \cos(\theta_{\text{h}}) - l_{\text{s}} \cos(\theta_{\text{h}} + \theta_{\text{k}})) \end{bmatrix} \tag{2.7}$$

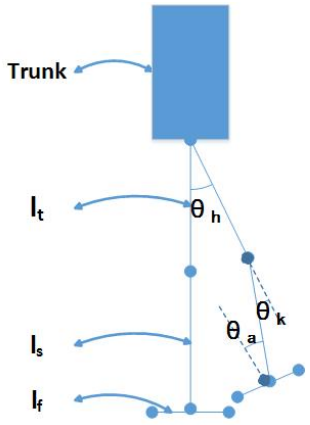



Figure 2.2: The Stand Pose for the robot, i.e., the default pose.

The equations above are based on the assumptions:

(1) The default configuration for the robot is the standing configuration, which is shown in figure 2.2. All joint angles are zero at this

configuration. The thigh and the shank are vertical while the foot is horizontal.

(2) The angles are positive when the links are rotating around the positive Y direction, which is to the right of the robot.

## 2.3 State of Walking

By first making the following assumptions, we can decouple the walking process into different phases and solve them separately [34].

(1) Biped robot walking can be broken down into different states: Single Support Phase, Double Support Phase, Start Phase, End Phase and Stand Phase.

(2) Single Support Phase and Double Support Phase will repeat in a periodic pattern.

(3) There is no impact or recoil during state transitions.

A robot starts from Standing, Start Phase, Double Support Phase, Single Support Phase, Double Support Phase, Single Support Phase, etc. This process will not stop until the robot is in last Double Support Phase and followed by End Phase to Stand at the end of the walking process.

### 2.3.1 Stand Phase

Standing is the basic pose for the robot. It requires all joint angles to be default value, i.e., zero. Since the robot is in static balance, the center of pressure should be at the vertical projection from the ankle to the ground. The state is shown in figure 2.3 a.

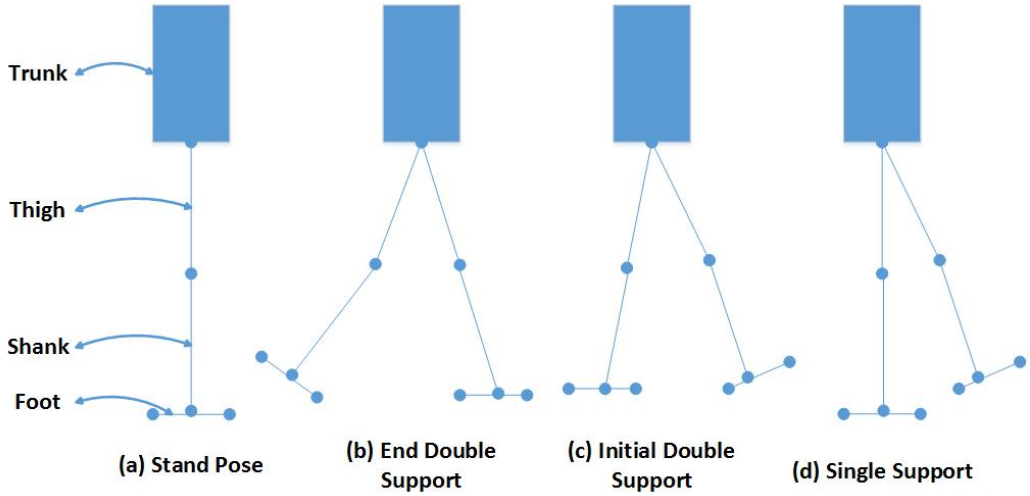



Figure 2.3: Different States for robot walking.

### 2.3.2 Single Support Phase

Single Support Phase describes when the robot is walking with only one foot touching the ground while the swing foot is dangling in the air. This state should be in dynamical balance. ZMP is bounded within the support foot. The state is shown in figure 2.3 d.

### 2.3.3 Double Support Phase

Double Support Phase describes when the robot is walking and both feet are touching the ground. This state should also be in dynamical balance so that the robot will not fall. ZMP is bounded in the support polygon. In the phase, the ZMP will shift from the foot in the back to the foot in the front.

### 2.3.4 Start Phase and End Phase

Start Phase and End Phase describe the robot from standing to walking and walking to standing. In other words, the robot shifts from static balance to dynamic balance. The general schema is similar to the Single Support Phase in that all involve moving with one foot supporting the body. The overall speed of the robot will increase and decrease during these two states.

### 2.3.5 Transition Model

A transition Model is sometimes called an impact model [1], which specifies the shifting from Single Support to Double Support. However the impact model focuses more on leg shifts and the shift matrices [1]. In this thesis, the transition model can be separated into **Initial Double Support** where the robot is shifting from Single Support to Double Support and **End Double Support** where the robot is shifting from Double Support to Single Support.

In **Initial Double Support**, the swing foot of the walking robot touches the ground. In detail, the heel of the swing foot will first touch the ground. The same foot will begin to rotate clockwise around the heel and the foot in the back; i.e., the standing foot in Single Support Phase will rotate clockwise around the toe. The ZMP is bounded within the back foot. This pose is shown in figure 2.3 c.

In **End Double Support**, the toe of the foot in the front will touch the ground. This is marked as the end of the feet rotation in Double Support. ZMP is bounded within the foot in the front. This pose is shown in figure 2.3 b.

Other work has focused on the transition from Double Support Phase and Single Support Phase. [10] illustrates that the robot is experiencing under-actuated, fully-actuated and over-actuated state periodically. This impact model is very important in the controlling part.

# CHAPTER 3

# TRAJECTORY GENERATION

In order to control the robots for walking, we have to determine some parameters of the robot's motion, including walking direction, total distance of walking, step length and speed of walking. The parameters can always be adjusted as a human can adjust his or her walking speed and direction easily. The details for the numerical values of walking parameters will be discussed in **Chapter 6** and **Chapter 7**. Here we will derive a mathematical model for Trajectory Generation.

The trajectory generation can be decoupled into several sections: gait generation, ZMP boundary generation, ZMP interpolation and COM generation. All parts will be discussed in the following text. At the end of trajectory generation, we will be able to generate reference trajectories for three masses in the three-mass model. Simultaneously, we will also be able to generate the reference joint trajectories. In other words, a kinematic solution to the heel-toe planner can be achieved.

## 3.1 Gait Generation

The first step for trajectory planning is gait generation. Since we are only interested in walking in a straight line, we may plan the step accordingly. Before heels of both feet will be planned, the gaits should be planned. The gait trajectory will thus be a collection of discrete patches. We may first assume the following in gait generation:

(1) The robot starts at the origin, i.e., $x_{\text{left}}(t=0) = x_{\text{right}}(t=0) = 0, y_{\text{left}}(t=0) = -\frac{w}{2}, y_{\text{right}}(t=0) = \frac{w}{2}$, where $w$ is defined in section 2.2.

(2) The robot starts walking with the right foot. After walking N steps,

the robot will stop with the left foot.

(3) The robot is moving in X direction.

The walking can be decoupled into sagittal and frontal plane [34]. As the two legs are switching alternately, heel positions will update in a period of $2t_{\mathrm{s}}$. The equations of the $x$ component for each gait can thus be derived as follows[1]:

$$x_{\mathrm{left}} = \sum_{\mathrm{i}=1}^{\frac{\mathrm{N}}{2}-1} 2l_{\mathrm{step}}u(t-2it_{\mathrm{s}}) + l_{\mathrm{step}}u(t-Nt_{\mathrm{s}}) \tag{3.1}$$

$$x_{\mathrm{right}} = \sum_{\mathrm{i}=1}^{\frac{\mathrm{N}}{2}-1} 2l_{\mathrm{step}}u(t-(2i+1)t_{\mathrm{s}}) + l_{\mathrm{step}}u(t-t_{\mathrm{s}}) \tag{3.2}$$

where $u(t)$ is the Heaviside step function and $l_{step}$ is the step length. The $y$ component is kept constant since the robot is walking in X direction only.

$$y_{\mathrm{left}} = -\frac{w}{2}, \quad y_{\mathrm{right}} = \frac{w}{2} \tag{3.3}$$

## 3.2 ZMP Boundary Condition

From the gait trajectories generated above, ZMP boundary conditions can be determined. In order to achieve dynamical balance, the ZMP has to stay inside the support polygon. As defined in section 2.3, we can define the upper bound and lower bound of the ZMPs. Here we use the terminology Periods of Gaits (POG) to describe periodic phases when the robot walks. The POG is described as the following sequence: **Single Support of Left Foot**, **First Double Support**, **Single Support of Right Foot** and **Second Double Support**. Since walking is a periodic sequence of foot step, we can decompose the walking into $\frac{N}{2}$ periods of gaits. Each POG is composed of single support of left foot, first double support, single support of right foot and second double support.

By applying the variables in Section 3.1, $t_{\mathrm{s}} = t_{\mathrm{ds}} + t_{\mathrm{ss}}$ where $t_{\mathrm{ds}}$ is the double

[1]This section only plans the steps of the robot instead of actual feet trajectory. It is not continuous, but it will give the boundary condition of ZMP trajectory.

support duration and $t_{\text{ss}}$ is the single support duration. For the $i^{\text{th}}$ POG, the robot has already walked $2(i-1)$ steps. Since the walking is periodic, we can calculate stepping for each period. Suppose the time starts at the beginning of the $i^{\text{th}}$ POG, the time offset will be $t_{\text{offset}} = 2(i-2)t_{\text{step}} + t_{is}$ where $t_{is}$ is the time spent in reaching POG from standing pose. The boundary conditions can be generated as the following sequence.

### 3.2.1 Singe Support of Left Foot

For the $i^{\text{th}}$ POG, the left foot will be the only support foot when $t \in [t_{\text{offset}}, t_{\text{offset}} + t_{\text{ss}})$. The following conditions can be derived. The $w$ represents the width of the robot, i.e. the distance between the two foot centers in sagittal direction and $u$ represents the width of the foot.

$$\begin{aligned} x_{\text{upper}}(t) &= x_{\text{left}}((2i-2)t_{\text{s}}) + l_{\text{f}} \\ x_{\text{lower}}(t) &= x_{\text{left}}((2i-2)t_{\text{s}}) \end{aligned} \tag{3.4}$$

$$\begin{aligned} y_{\text{upper}}(t) &= -\frac{w}{2} + \frac{u}{2} \\ y_{\text{lower}}(t) &= -\frac{w}{2} - \frac{u}{2} \end{aligned} \tag{3.5}$$

### 3.2.2 First Double Support

For the $i^{\text{th}}$ POG, the First Double Support happens when $t \in [t_{\text{offset}} + t_{\text{ss}}, t_{\text{offset}} + t_{\text{s}})$.

$$\begin{aligned} x_{\text{upper}}(t) &= x_{\text{right}}((2i-1)t_{\text{s}}) \\ x_{\text{lower}}(t) &= x_{\text{left}}((2i-2)t_{\text{s}}) + l_{\text{f}} \end{aligned} \tag{3.6}$$

$$\begin{aligned} y_{\text{upper}}(t) &= \frac{w}{2} + \frac{u}{2} \\ y_{\text{lower}}(t) &= -\frac{w}{2} - \frac{u}{2} \end{aligned} \tag{3.7}$$

This is the boundary condition in time domain. However, it cannot describe the legal points since the robot is not moving at a constant speed. Therefore, $x_{\text{zmp}}(t)$ and $y_{\text{zmp}}(t)$ cannot be determined. The boundary condition above is a huge rectangular in x-y plane, which contains lots of points that are illegal. This idea is illustrated in figure 3.1. A smaller range is nec-

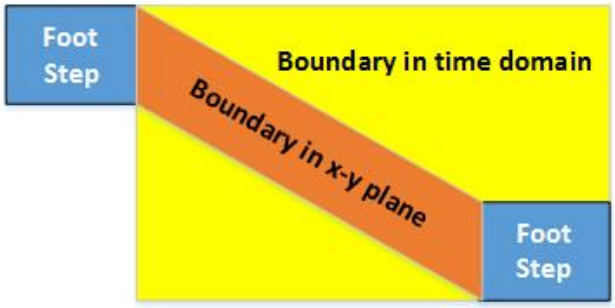



Figure 3.1: The Stand Pose for the robot, i.e., the default pose.

essary. The boundary in the x-y plane is a parallelogram. The $x$ component will be the same as in time domain since they will not change throughout the Double Support Phase. The $y$ component now can be expressed as a function of x. The upper boundary is a straight line passing through $(x_{\text{upper}}(t), y_{\text{upper}}(t))$ and $(x_{\text{lower}}(t), y_{\text{upper}}(t))$ where all variables are defined in (3.6) and (3.7). The lower boundary is another straight line passing through $(x_{\text{upper}}(t), y_{\text{lower}}(t))$ and $(x_{\text{lower}}(t), y_{\text{lower}}(t))$ where all variables are also defined in (3.6) and (3.7). The $y$ component with respect to time can be expressed as follows:

$$\begin{aligned} y^{(2)}_{\text{upper}}(x) &= \frac{[wx - \frac{w}{2}(x_{\text{upper}}(t) + x_{\text{lower}}(t)) + \frac{u}{2}(x_{\text{upper}}(t) - x_{\text{lower}}(t))]}{x_{\text{upper}}(t) - x_{\text{lower}}(t)} \\ y^{(2)}_{\text{lower}}(x) &= \frac{[wx - \frac{w}{2}(x_{\text{upper}}(t) + x_{\text{lower}}(t)) - \frac{u}{2}(x_{\text{upper}}(t) - x_{\text{lower}}(t))]}{x_{\text{upper}}(t) - x_{\text{lower}}(t)} \end{aligned} \tag{3.8}$$

### 3.2.3 Single Support of Right Foot

For the $i^{\text{th}}$ POG, the left foot will be the only support foot when $t \in [t_{\text{offset}} + t_{\text{s}}, t_{\text{offset}} + t_{\text{s}} + t_{\text{ss}})$. As in **Single Support of Left Foot**, boundary conditions can be derived as follows:

$$\begin{aligned} x_{\text{upper}}(t) &= x_{\text{right}}((2i-1)t_{\text{s}}) + l_{\text{f}} \\ x_{\text{lower}}(t) &= x_{\text{right}}((2i-2)t_{\text{s}}) \end{aligned} \tag{3.9}$$

$$
\begin{aligned}
y_{\text{upper}}(t) &= \frac{w}{2} + \frac{u}{2} \\
y_{\text{lower}}(t) &= \frac{w}{2} - \frac{u}{2}
\end{aligned}
\tag{3.10}
$$

### 3.2.4 Second Double Support

For the $i^{\text{th}}$ POG, the Second Double Support happens when $t_{\text{offset}} + t \in (t_{\text{s}}+t_{\text{ss}}, t_{\text{offset}}+2t_{\text{s}})$. Again, the boundary conditions expressed in time domain can be shown:

$$
\begin{aligned}
x_{\text{upper}}(t) &= x_{\text{right}}(2it_{\text{s}}) \\
x_{\text{lower}}(t) &= x_{\text{left}}((2i-1)t_{\text{s}}) + l_{\text{f}}
\end{aligned}
\tag{3.11}
$$

$$
\begin{aligned}
y_{\text{upper}}(t) &= \frac{w}{2} + \frac{u}{2} \\
y_{\text{lower}}(t) &= -\frac{w}{2} - \frac{u}{2}
\end{aligned}
\tag{3.12}
$$

In the x-y plane, we can also achieve the parallelogram by computing the straight lines. The upper boundary is the line passing through $(x_{\text{upper}}(t), y_{\text{upper}}(t))$ and $(x_{\text{lower}}(t), y_{\text{upper}}(t))$ where all variables are defined in (3.6) and (3.7). The lower boundary is another straight line passing through $(x_{\text{upper}}(t), y_{\text{lower}}(t))$ and $(x_{\text{lower}}(t), y_{\text{lower}}(t))$ where all variables are also defined in (3.6) and (3.7). The analytic expressions for two lines are the following.

$$
\begin{aligned}
y^{(4)}_{\text{upper}}(x) &= \frac{[wx - \frac{w}{2}(x_{\text{upper}}(t) + x_{\text{lower}}(t)) - \frac{u}{2}(x_{\text{upper}}(t) - x_{\text{lower}}(t))]}{x_{\text{upper}}(t) - x_{\text{lower}}(t)} \\
y^{(4)}_{\text{lower}}(x) &= \frac{[wx - \frac{w}{2}(x_{\text{upper}}(t) + x_{\text{lower}}(t)) + \frac{u}{2}(x_{\text{upper}}(t) - x_{\text{lower}}(t))]}{x_{\text{upper}}(t) - x_{\text{lower}}(t)}
\end{aligned}
\tag{3.13}
$$

## 3.3 ZMP Trajectory Generation

As we have already generated the boundary conditions for ZMP trajectory, we can now generate discrete ZMP trajectory points. In general, we can separate the entire trajectory to periodic pieces, each starting with a Double Support Phase and ending with a Single Support Phase. As the motion can be decoupled into sagittal plane and frontal plane, desired ZMP trajectory

in both x direction and y direction can be generated separately.

### 3.3.1 ZMP Trajectory Generation

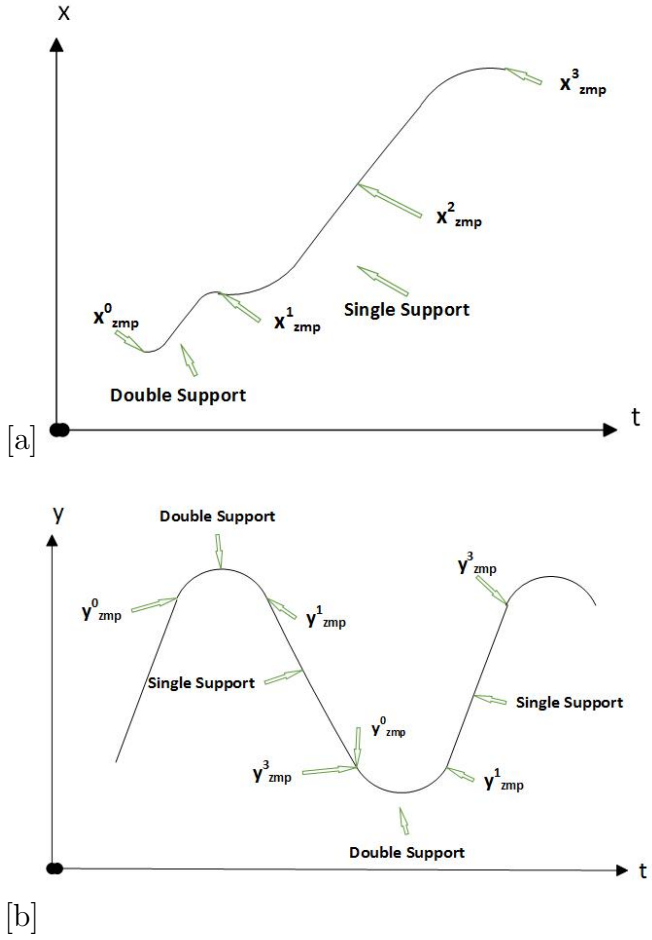



Figure 3.2: (a) represents the ZMP boundary notation for $x$ component; (b) represents the ZMP boundary notation for $y$ component

First, we have to give several points to both $x$ trajectory and $y$ trajectory to conduct interpolation. At $i^{\text{th}}$ POG, the following points are assumed for each stage. The points are different with odd steps and even steps. The labelling of number can be referred in the figure 3.2.

$$
\begin{aligned}
x^0_{\text{zmp}} &= x_{\text{right}}((2i-1)t_{\text{s}}) & x^0_{\text{zmp}} &= x_{\text{left}}(2it_{\text{s}}) \\
x^1_{\text{zmp}} &= x^0_{\text{zmp}} + l_{\text{f}} & x^1_{\text{zmp}} &= x_{\text{left}}(2it_{\text{s}}) + l_{\text{f}} \\
x^3_{\text{zmp}} &= x_{\text{left}}(2it_{\text{s}}) & x^3_{\text{zmp}} &= x_{\text{left}}(2it_{\text{s}}) + \frac{1}{2}l_{\text{step}} \\
x^2_{\text{zmp}} &= \frac{1}{2}(x^1_{\text{zmp}} + x^3_{\text{zmp}}) & x^2_{\text{zmp}} &= \frac{1}{2}(x^1_{\text{zmp}} + x^3_{\text{zmp}}) \\
y^0_{\text{zmp}} &= \frac{w}{2} - \frac{u}{2} & y^0_{\text{zmp}} &= -\frac{w}{2} + \frac{u}{2} \\
y^1_{\text{zmp}} &= \frac{w}{2} - \frac{u}{2} & y^1_{\text{zmp}} &= -\frac{w}{2} + \frac{u}{2} \\
y^3_{\text{zmp}} &= -\frac{w}{2} + \frac{u}{2} & y^3_{\text{zmp}} &= \frac{w}{2} - \frac{u}{2}
\end{aligned}
\tag{3.14}
$$

In the above points, the left corresponds to the period where the right foot is standing during the Single Support Phase while the right corresponds to the period where the left foot is standing during the Single Support Phase. With the correct time, the trajectory can be interpolated correctly. In order to fit the ZMP trajectory into the ZMP boundary condition, an interpolation method is proposed. This method use Cosine-Cosine Interpolation for $x$ trajectory and Quadratic-Cosine Interpolation for $y$ trajectory. Some other methods are used to ensure that both trajectories fit inside the ZMP boundary conditions. We used Cosine based interpolation method is due to the fact that we can control ZMP trajectory inside the support polygon while third polynomial depends on boundary conditions. They will be discussed in the following sections.

The reason why we chose cosine based interpolation is because during the experiment, we found that the some parts of ZMP may exceeds the boundary using third order polynomial. If we use cosine based interpolation, we can assure that the entire ZMP will be in the support polygon by controlling the angular frequency $\omega$ as introduced in later sections.

### 3.3.2 Cosine-Cosine Interpolation

The cosine-cosine interpolation is designed for the $x$ component, which means that two cosine functions are used to describe the $x$ component in one period.

$$
\begin{aligned}
x_1(t) &= A_1 \cos(\omega_1 t + \varphi_1) + O_1 \\
x_2(t) &= A_2 \cos(\omega_2 t + \varphi_2) + O_2
\end{aligned}
\tag{3.15}
$$

The index refers to the phase where Double Support Phase when the index is 1 and Single Support Phase when index is 2. $O$ is the offset. $A_\mathrm{i}$, $\omega_1$, $\varphi_\mathrm{i}$, and $O_\mathrm{i}$ are constants throughout each phase. To simplify the solving process, we set the offset $O_2$ to be $x^2_{\mathrm{zmp}}$ as in (3.14). This means that the first order derivative should be zero at the transition, i.e., the transition from double support to single support. The advantage for such setting is that the system tries to minimize the error within one step instead of minimizing the error when it is already been too large. Hence, the following equation can be achieved by scrutinizing the continuity and first order continuity:

$$\frac{x^0_{\mathrm{zmp}} - x^1_{\mathrm{zmp}}}{2}\cos(\omega_1 t_{\mathrm{ds}}) + \frac{\dot{x}^0_{\mathrm{zmp}}}{\omega_1}\sin(\omega_1 t_{\mathrm{ds}}) = \frac{x^1_{\mathrm{zmp}} - x^0_{\mathrm{zmp}}}{2} \tag{3.16}$$

Here the $\dot{x}^0_{\mathrm{zmp}}$ should be the first order derivative of the $x$ component of the ZMP trajectory when the robot enters Double Support. Normally, according to the setting, this term should be zero. This equation should yield $\omega_1 < \frac{2\pi}{t_{\mathrm{ss}}}$. The $\varphi_1$ and $\varphi_2$ are cancelled out from property of continuity. The detail of the solution will be discussed later in **Chapter 6**. The solution will give $\varphi_2 = \mathrm{atan2}(\frac{-\dot{x}^0}{\omega_1 x^0_{\mathrm{zmp}}})$. Using similar methods, we can derive the equation for $x_2$:

$$\frac{x^1_{\mathrm{zmp}} - x^3_{\mathrm{zmp}}}{2}\cos(\omega_2 t_{\mathrm{ss}}) + \frac{\dot{x}^1_{\mathrm{zmp}}}{\omega_2}\sin(\omega_2 t_{\mathrm{ss}}) = \frac{x^3_{\mathrm{zmp}} - x^1_{\mathrm{zmp}}}{2} \tag{3.17}$$

After solving for $\varphi_\mathrm{i}$ and $\omega_\mathrm{i}$, we can plug in the boundary conditions as in (3.14), we can solve for the rest of the constants.

### 3.3.3 Quadratic-Cosine Interpolation

This interpolation method is designed for the $y$ component. In other words, we use a cosine function to express the $y$ component for Double Support Phase and a quadratic function to express the $y$ component for Single Support Phase.

$$\begin{aligned} y_1(t) &= B\cos(\omega_3 t + \varphi_3) + O_3 \\ y_2(t) &= b_0 + b_1(t - t_0) + b_2(t - t_0)^2 \end{aligned} \tag{3.18}$$

The index refers to the phase, i.e., Double Support when the index equals and Single Support when the index is two. $B$, $b_\mathrm{i}$, $\omega_\mathrm{i}$, $\varphi_\mathrm{i}$ and $O_\mathrm{i}$ are constants. By plugging the boundary points defined in (3.14), $b_0, b_1, b_2$ can be easily solved. Since the transition from $y_1$ to $y_2$ should be continuous and first

order continuous, we can derive the following equation:

$$y^1_{\text{zmp}} \cos(\omega_3 t_{\text{ds}}) + \frac{\dot{y}^1_{\text{zmp}}}{\omega_3} \sin(\omega_3 t_{\text{ds}}) = y^3_{\text{zmp}} \tag{3.19}$$

In this equation, $\dot{y}^1_{\text{zmp}}$ is the first order derivative at the end of $y_1$, or the beginning of $y_2$. Since there are two cases where the left foot is standing or the right foot is standing, $\varphi_3$ can be calculated separately.

$$\varphi_3 = \begin{cases} \text{atan2}\left(\frac{-\dot{y}^1_{\text{zmp}}}{\omega_3 y^1_{\text{zmp}}}\right) & \dot{y}^1_{\text{zmp}} < 0 \\ \text{atan2}\left(\frac{y^1_{\text{zmp}}}{-\omega_3 y^1_{\text{zmp}}}\right) & \dot{y}^1_{\text{zmp}} > 0 \end{cases} \tag{3.20}$$

After solving for $\varphi_{\text{i}}$ and $\omega_{\text{i}}$, we can plug in the boundary conditions as in (3.14), we can solve for the rest of the constants.

Although the quadratic equation is easy to solve, it may exceed the boundary at the maximum or minimum of the parabola. We implement a correction method, which is a quartic function, which is added to the original quadratic function if the quadratic function will exceed the boundary condition. This equation can be defined as follow.

$$\delta(t) = At^4 + Bt^3 + Ct^2 + Dt + E \tag{3.21}$$

By adding this correction function to the original quadratic function, i.e., $y_{2\text{corrected}}(t) = y_2(t) + \delta(t)$, the boundary conditions are fully met. The detail of when the original quadratic function will exceed the boundary condition and analysis of the quintic function will be given in **Appendix A**.

### 3.3.4 End Phase

End Phase is quite unique. One can hardly find a function described above to fit this phase since the first order derivative of the $y$ component of ZMP trajectory should reach zero since the ZMP stops moving as the robot stops. At this moment we do not need to care about $x$ component since the $x$ component will reach its final position with first order derivative equals to zero. For $y$ order derivative equals to zero. We used a simple function

$$y_{\text{zmp}}(t) = A + Be^{-\alpha \text{t}} \tag{3.22}$$

By selecting proper A, B and $\alpha$, we can assure that the first order of $y$ component at the end of the walk will reach zero, i.e., $\lim_{t\to\infty} \dot{y}_{zmp} = 0$. And also, A, B and $\alpha$ can be solved by following the continuity and first order continuity. The continuity can be achieved by plugging the boundary condition for at the End Phase. An analysis of the stability of the $y$ component at the End Phase will be given in **Appendix A**.

We have derived all analytic functions of ZMP trajectories with respect to time in the world frame. The next step will be to generate the COM trajectories for the three-mass model. Since the three-mass model will have three masses, we will generate trajectories for all three masses. From there we can visualize the result and see whether the ZMP stays inside the support polygon.

# CHAPTER 4

# COM TRAJECTORIES GENERATION

In order to generate all the reference joint angles, trajectories of all three point masses should be first determined. After that, we can do inverse kinematic to solve for the joint angles.

To begin with, suppose this system has nine free variables for translation: the position of two feet and the position of the trunk. However, this system is far less constrained than it should be since the only constraints are the ZMP trajectory and some assumptions made in **Chapter 2** for the thee-mass model. Therefore, before proceeding several assumptions should be made based on which phase the robot is experiencing. As introduced earlier, the ZMP equations can be shown in (2.3) and (2.4). Now we make assumptions similar to the 3D linear inverted pendulum [35, 36].

(1) The robot is walking on a flat ground, i.e., $z_{\text{zmp}} = 0$.

(2) When in Single Support Phase, all three point masses' heights have negligible change.

(3) The support leg keeps straight throughout the entire Single Support Phase.

Due to symmetry, only half of a walking period needs to be studied before going into the generation of the COM trajectories. Here we use new notations to simplify the model. In the ZMP equations for three masses, i.e. (2.3) and (2.4), we make the following rules:

(1) $m_1$ represents the mass of the standing foot and $(x_1, y_1, z_1)$ represents center of the standing foot.

(2) $m_2$ represents the mass of the trunk and $(x_2, y_2, z_2)$ represents center of the trunk.

(3) $m_3$ represents the mass of the swinging foot and $(x_3, y_3, z_3)$ represents

center of the swinging foot.

## 4.1 Transition Model

### 4.1.1 ZMP Trajectory Generation

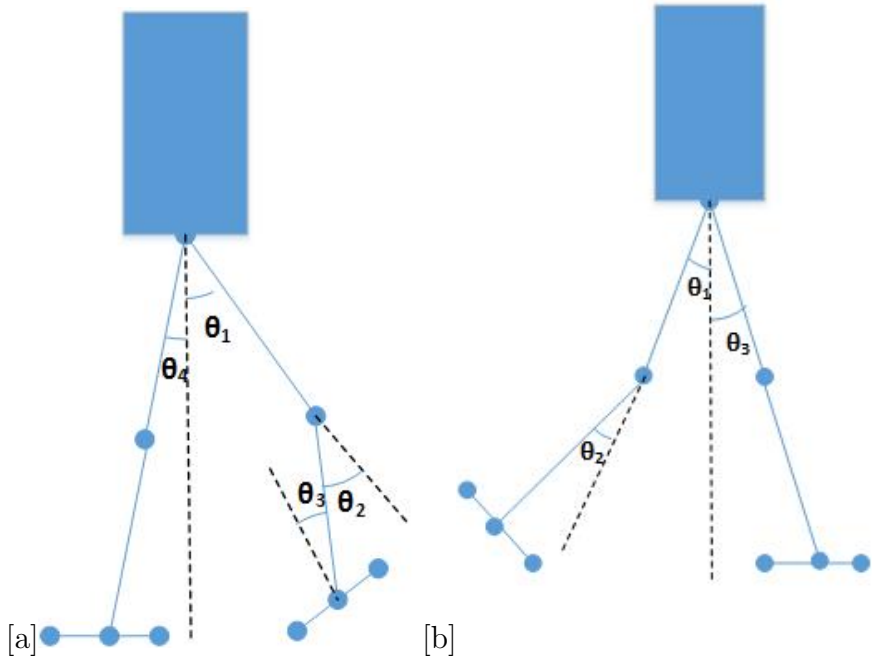



Figure 4.1: (a) shows the notation of angles for Initial Double Support; (b) shows the notation of angles for End Double Support

So far, transition has been mentioned many times. Here we will define the behavior of the robot during the transition from Single Support Phase to Double Support Phase and from Double Support Phase to Single Support Phase. A general idea inspired by human walking is shown in figure 4.1. Once this guideline is set, we can move on to solve the differential equations for Center Of Masses.

### 4.1.2 From Double Support to Single Support

We set the robot at this transition model to have the following properties following figure 4.1 b.

1) The leg in the front remains straight, i.e., $\theta_1 = 0$.

2) The ankle of the foot in the back should maintain right angle, i.e., $\theta_3 = \frac{\pi}{2}$.

Using the indexing in the figure 4.1, we need to solve three angles $\theta_{\mathrm{i}}$, which is a vector of angles with dimension of three. According to these properties, we can derive the following nonlinear equations. All the parameters are determined by the model of the robot. The details of solving these equations will be discussed in **Chapter 6**.

$$\begin{aligned} l_{\mathrm{t}} \sin(-\theta_1) - l_{\mathrm{s}} \sin(\theta_1 + \theta_2) - (1-\eta) l_{\mathrm{f}} \cos(\theta_1 + \theta_2) + \\ (l_1 + l_2) \sin(\theta_3) = (l_{\mathrm{step}} - l_{\mathrm{f}} + l_{\mathrm{f}}\eta) \\ l_{\mathrm{t}} \cos(\theta_1) + l_{\mathrm{s}} \cos(\theta_1 + \theta_2) - (1-\eta) l_{\mathrm{f}} \sin(\theta_1 + \theta_2) = h \\ (l_1 + l_2) \cos(theta_3) = h \end{aligned} \tag{4.1}$$

According to figure 4.1, $\theta_1, \theta_2 < 0$ and $\theta_3 > 0$. Therefore, when the robot is in this configuration, the foot in the front should touch the ground with the entire sole while the foot in the back should have an angle with the ground which is specified by $\theta_{\mathrm{i}} = \theta_2 + \theta_3$. The height of the swinging foot during Single Support Phase can be calculated using $\theta_{\mathrm{i}}$ as $z_{\mathrm{i}} = (1-\eta) * \sin(\theta_{\mathrm{i}})$. The positions of all three masses with respect to the world can be determined. Since we are trying to solve nonlinear equations, there must be some error. The error analysis will be presented in **Chapter 6**.

### 4.1.3 From Single Support to Double Support

Also, this transition model is given with the following properties in figure 4.1 a.

1) The leg in the back remains straight, i.e., $\theta_3 = 0$.

The configuration is also illustrated in figure 4.1. Again, using the indexing in the figure and elementary geometry, we are going to solve for $\theta_{\mathrm{e}}$, which is a

vector with dimension of four. We can achieve the following set of nonlinear equations, where $z_{\mathrm{i}}$ has been calculated in the previous part.

$$\begin{aligned} l_{\mathrm{t}}\sin(\theta_1) + l_{\mathrm{s}}\sin(\theta_1+\theta_2) - \eta l_{\mathrm{f}}\cos(\theta_1+\theta_2-\theta_3) + \\ (l_{\mathrm{s}}+l_{\mathrm{t}})\sin(-\theta_4) = (l_{\mathrm{step}} - l_{\mathrm{f}} + l_{\mathrm{f}}\eta) \\ l_{\mathrm{t}}\cos(\theta_1) + l_{\mathrm{s}}\cos(\theta_1+\theta_2) + \eta l_{\mathrm{f}}\sin(\theta_1+\theta_2-\theta_3) \\ = (l_{\mathrm{t}}+l_{\mathrm{s}})\cos(\theta_4) \\ (l_{\mathrm{t}}+l_{\mathrm{s}})\cos(\theta_4) = h \\ (1-\eta)l_{\mathrm{f}}\sin(\theta_1+\theta_2-\theta_3) = z_{\mathrm{i}} \end{aligned} \tag{4.2}$$

According to the figure 4.1, $\theta_1 > 0$ and $\theta_2, \theta_4 < 0$. $z_{\mathrm{i}}$ is calculated in previous section. With this configuration, the feet in the back should touch the ground with the entire sole while the foot in the front should have an angle between itself and the ground as $\theta_e = \theta_1 + \theta_2 - \theta_3$. The height of the center of the front foot should be $z_e = (1-\eta)l_{\mathrm{f}}\sin(\theta_e)$ . As before, the detail for solving the equations numerically will be discussed in **Chapter 6**.

## 4.2 Trajectory Generation

As we have derived so far, all ZMP trajectories and COM boundary conditions have been generated. Next we can move to trajectory generation and solve positions and velocity for all three masses. By transforming equations (2.3), (2.4) and the assumption $\ddot{z}_{\mathrm{i}} = 0$, we have the following equations:

$$\begin{aligned} \sum_{\mathrm{i}=1}^{3} m_{\mathrm{i}}[g x_{\mathrm{i}} - z_{\mathrm{i}}\ddot{x}_{\mathrm{i}}] = M_{\mathrm{total}} g x_{\mathrm{zmp}}(t) \\ \sum_{\mathrm{i}=1}^{3} m_{\mathrm{i}}[g y_{\mathrm{i}} - z_{\mathrm{i}}\ddot{y}_{\mathrm{i}}] = M_{\mathrm{total}} g y_{\mathrm{zmp}}(t) \end{aligned} \tag{4.3}$$

The motions can also be decoupled into two directions: sagittal plane and frontal plane [34]. We will decouple the equations for each phase, i.e., Single Support Phase and Double Support Phase.

### 4.2.1 Single Support Phase

According to previous assumptions, if the heights of all three masses remain constant, the equation (4.3) can be written as following equations:

$$\begin{aligned} &m_1 g x_1 + m_2 g x_2(t) + m_3 g x_3(t) - m_2 h \ddot{x}_2(t) - m_3 z_e \ddot{x}_3(t) = M_{\text{total}} g x_{\text{zmp}}(t) \\ &m_1 g y_1 + m_2 g y_2(t) + m_3 g y_3 - m_2 h \ddot{y}_2(t) = M_{\text{total}} g y_{\text{zmp}}(t) \end{aligned} \tag{4.4}$$

Decoupling equation (4.4), we analyze the $x$ component first. In the $x$ component, there are two free variables, $x_2(t)$ and $x_3(t)$. If we can specify one of the variables, the equation becomes a second order non-homogeneous ordinary differential equation, which is easy to solve. Since $x_{\text{zmp}}$ is a cosine function achieved in the previous section, another cosine function is applied here to describe $x_3$, i.e.,

$$x_3(t) = A\cos(\omega t + \varphi) \tag{4.5}$$

To solve for $A$, $\omega$ and $\varphi$, only boundary conditions need to be plugged in, which have already been calculated in the previous part. We can now derive the differential equation for $x_2(t)$ and solve for $x_2(t)$. The techniques and analytic solutions of solving differential equations will not be discussed here.

Starting to analyze the $y$ component, we found it relatively easy to solve this issue. Since $y_1$ and $y_3$ are set to constant, the only variable is $y_2$. Therefore, the following differential equation can be derived. At this moment, the trajectories of all three masses during Single Support Phase are determined.

$$\ddot{y}_2(t) - \frac{g}{h} y_2(t) = \frac{m_1 g}{m_2 g} y_1 + \frac{m_3 g}{m 2_g} y_3 - \frac{M_{\text{totalg}}}{m_2 g} y_{\text{zmp}}(t) \tag{4.6}$$

### 4.2.2 Double Support Phase

Double Support Phase is far more complicated than Single Support Phase. In this phase, the motions of two feet are purely rotational. The front foot rotates around the toe and the back foot rotates around the heel. This

involves the motion of both feet in $x$ and $z$ directions. Assuming that the change of height of the trunk is very small, i.e., $\dot{y}_2 = 0$, we now have the ZMP differential equations as follows:

$$\begin{aligned} m_1x_1(\ddot{z}_1+g)+m_2gx_2+m_3x_3(\ddot{z}_3+g)-m_1z_1\ddot{x}_1-m_2z_2\ddot{x}_2-m_3z_3\ddot{x}_3 \\ = x_{\text{zmp}}[m_1(\ddot{z}_1+g)+m_2g+m_3(\ddot{z}_3+g)] \\ m_1y_1(\ddot{z}_1+g)+m_2gy_2+m_3y_3(\ddot{z}_3+g)-m_2z_2\ddot{y}_2 \\ = y_{\text{zmp}}[m_1(\ddot{z}_1+g)+m_2g+m_3(\ddot{z}_3+g)] \end{aligned} \tag{4.7}$$

In these equations, $x_1$, $x_3$, $z_1$ and $z_3$, which are shown in figure 4.2 can be formulated as follows due to pure rotation:

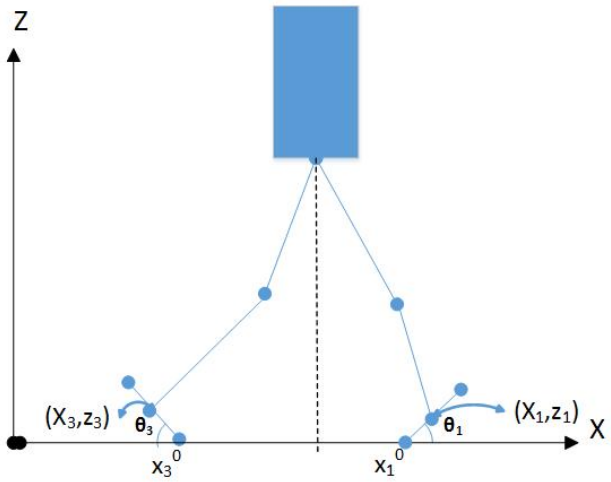



Figure 4.2: Notation for feet rotation during Double Support

$$\begin{aligned} x_1 &= x_1^0+\frac{1}{2}l_{\text{f}}\cos(\theta_1), \qquad & z_1 &= \frac{1}{2}l_{\text{f}}\sin(\theta_1) \\ x_3 &= x_3^0-\frac{1}{2}l_{\text{f}}\cos(\theta_3), \qquad & z_3 &= -\frac{1}{2}l_{\text{f}}\sin(\theta_3) \end{aligned} \tag{4.8}$$

We must first determine $x_1$, $x_3$, $z_1$ and $z_3$ to solve $x_2$ and $y_2$. From the equations above, we need to define two variables $\theta_1$ and $\theta_3$. Here we use linear function to define these variables because in our experimental implementation, linear functions are sufficient to describe those two variables, i.e.,

$$\begin{aligned} \theta_1 &= f_1t+f_0 \\ \theta_3 &= g_1t+g_0 \end{aligned} \tag{4.9}$$

The boundary conditions can be used here that $\theta_1(0) = \theta_{\mathrm{i}}, \theta_1(t_{\mathrm{ds}}) = 0$ and $\theta_3(0) = 0, \theta_3(t_{\mathrm{ds}}) = \theta_e$. By plugging these items back into the differential equations (4.4), we can derive the analytic function for $x_2$ and $y_2$ with respect to time.

### 4.2.3 End Phase

Since the only thing different at the End Phase is the $y$ component, we do not need to use any additional functions to describe unconstrained variables. We may just use the differential equation (4.6) and solve for the $y$ component at the End Phase. Together with all the trajectories generated in previous parts, we achieve the trajectories for all three masses.

## 4.3 Joint Trajectory and Joint Velocities

By using inverse kinematics, all the joint angles can be calculated. The details for inverse kinematics will be discussed in **Chapter 6**. Also, we can derive all the joint velocities for our model since the Jacobian is full rank when a leg is not in singular configuration, which is specified for the model of Reemc Robot. The derivation will be in **Chapter 6**. Now we achieve a kinematic solution to the feet rotation planner. The visualization will be presented and discussed in **Chapter 7**. In the next chapter, we present a dynamic solution. Control methods will be applied to robot walking. The corresponding Matlab code for **Chapter 3** and **Chapter 4** will be attached in **Appendix D**.

# CHAPTER 5

# ROBOTIC CONTROL

The previous chapter focused on a kinematic approach for walking planning. This chapter will focus on a dynamic approach. With reference trajectory generated, we can easily start the simulation. However, the reference trajectory cannot guarantee success of simulation due to modelling error and other disturbance. With little disturbance, the robot may fall. A control system is needed here to cancel out the modelling error and disturbance. We used similar methods in [37]. Since we added the feature of feet rotation, we used the three-mass model discussed in **Chapter 2** instead. In this section, we will first propose a control system in continuous time. Then we will move to a discrete time system. We will find an optimal control input to the discrete time system by treating the problem as infinite horizon LQR problem in discrete time.

The control method we use is preview control, which was first proposed in [38] and uses future information to control current input. The input is designed to change the position of the trunk. Here, we use the following notations:

$$p_1 = \begin{bmatrix} x_{\text{sup}} \\ y_{\text{sup}} \\ z_{\text{sup}} \end{bmatrix}, p_3 = \begin{bmatrix} x_{\text{sw}} \\ y_{\text{sw}} \\ z_{\text{sw}} \end{bmatrix}, p_{\text{base}} = \begin{bmatrix} x_{\text{base}} \\ y_{\text{base}} \\ z_{\text{base}} \end{bmatrix} \tag{5.1}$$

As in [37], we define $p_{\text{base}}$ as the position of the boody of the robot, $p_{\text{sup}}^{\text{base}}$ as the position of the support leg with respect to the body of the robot and $p_{\text{sw}}^{\text{base}}$ as the position of the swing leg with respect to the body of the robot. Therefore, the position of swing mass and support mass can now be expressed as

$$\begin{aligned} p_2 &= p_{\text{base}} + p_{\text{sup}}^{\text{base}} \\ p_3 &= p_{\text{base}} + p_{\text{sw}}^{\text{base}} \end{aligned} \tag{5.2}$$

According to the equation of ZMP (2.1) and (2.2),

$$\begin{aligned} p_x = & \frac{m_{\text{base}}(x_{\text{base}} + x_{\text{body}}^{\text{base}}) + m_{\text{leg}}(x_{\text{base}} + x_{\text{sup}}^{\text{base}}) + m_{\text{leg}}(x_{\text{base}} + x_{\text{sw}}^{\text{base}})}{M} \\ & - \frac{m_{\text{base}} h\ddot{x}_{\text{base}} + m_{\text{leg}} z_e(\ddot{x}_{\text{base}} + \ddot{x}_{\text{sw}})}{Mg} \\ p_y = & \frac{m_{\text{base}}(y_{\text{base}} + y_{\text{body}}^{\text{base}}) + m_{\text{leg}}(y_{\text{base}} + y_{\text{sup}}^{\text{base}}) + m_{\text{leg}}(y_{\text{base}} + y_{\text{sw}}^{\text{base}})}{M} \\ & - \frac{m_{\text{body}} h\ddot{y}_{\text{base}}}{Mg} \end{aligned} \tag{5.3}$$

These two equations can written as follows:

$$\begin{aligned} p_x &= \frac{m_{\text{base}} + 2m_{\text{leg}}}{M} x_{\text{base}} - \frac{E_z}{g}\ddot{x}_{\text{base}} + E_x - \frac{m_{\text{leg}} z_e}{Mg}\ddot{x}_{\text{sup}} + \frac{m_{\text{leg}}}{M}(x_{\text{sup}}^{\text{base}} + x_{\text{sw}}^{\text{base}}) \\ p_y &= \frac{m_{\text{base}} + 2m_{\text{leg}}}{M} y_{\text{base}} - \frac{E_z}{g}\ddot{x}_{\text{sw}} + \frac{m_{\text{leg}}}{M}(y_{\text{sup}}^{\text{base}} + y_{\text{sw}}^{\text{base}}) \\ E_{\text{z}} &= \frac{m_{\text{base}} h + m_{\text{leg}} z_e}{M}, E_{\text{x}} = \frac{m^{\text{base}} x_{\text{body}}^{\text{base}}}{M}, E_{\text{y}} = \frac{m_{\text{base}} y_{\text{body}}^{\text{base}}}{M} \end{aligned} \tag{5.4}$$

$E_{\text{x}}$, $E_{\text{y}}$ and $E_{\text{z}}$ are variables that only involve the base location. Now we may define the input as the third derivative of the position of the body [17], i.e.,

$$\begin{aligned} \dddot{x}_{\text{base}} &= u_x^{\text{body}} \\ \dddot{y}_{\text{base}} &= u_y^{\text{body}} \end{aligned} \tag{5.5}$$

Using the third order derivative as input, we can directly control the robot in jerk dimension. Using the jerk dimension input, we can control the position, speed and acceleration of the robot. Now, we will move from continuous time to discrete time by changing variables: $k = \lfloor \frac{t}{\Delta t} \rfloor$. We can define the state variable and input as

$$X_{\text{base}}[k] = \begin{bmatrix} \vec{x}_{\text{base}}[k] \\ \vec{y}_{\text{base}}[k] \end{bmatrix}, u_{\text{base}}[k] = \begin{bmatrix} u_x^{\text{base}}(k\Delta t) \\ u_y^{\text{base}}(k\Delta t) \end{bmatrix}, p_{\text{zmp}}^{\text{out}}[k] = \begin{bmatrix} p_x^{\text{out}}(k\Delta t) \\ p_y^{\text{out}}(k\Delta t) \end{bmatrix} \tag{5.6}$$

$$\vec{x}_{\text{base}} = \begin{bmatrix} x_{\text{base}} \\ \dot{x}_{\text{base}} \\ \ddot{x}_{\text{base}} \end{bmatrix}, \vec{y}_{\text{base}} = \begin{bmatrix} y_{\text{base}} \\ \dot{y}_{\text{base}} \\ \ddot{y}_{\text{base}} \end{bmatrix} \tag{5.7}$$

The optimal controller for discrete time system is designed to meet the demand of previewable control. The transition equations can now be defined

as in [37]:

$$\begin{aligned} X_{\mathrm{b}}[k+1] &= AX_{\mathrm{base}}[k] + Bu_{\mathrm{base}}[k] \\ p^{\mathrm{out}}_{\mathrm{zmp}}[k] &= DX_{\mathrm{base}}[k] + E + F[k] \end{aligned} \tag{5.8}$$

where matrix

$$\begin{aligned} A &= \begin{bmatrix} 1 & \Delta t & \frac{\Delta^2 t}{2} & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{\Delta^2 t}{2} \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\ B &= \begin{bmatrix} \frac{\Delta^3 t}{6} & 0 \\ \frac{\Delta^2 t}{2} & 0 \\ \Delta t & 0 \\ 0 & \frac{\Delta^3 t}{6} \\ 0 & \frac{\Delta^2 t}{2} \\ 0 & \Delta t \end{bmatrix} \end{aligned} \tag{5.9}$$

$A$, $B$, $C$ are introduced in [37], which can also be used in our three-mass model. From (5.4),

$$\begin{aligned} D &= \begin{bmatrix} \frac{m_{\mathrm{base}}+2m_{\mathrm{leg}}}{M} & 0 & -\frac{E_z}{g} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{m_{\mathrm{base}}+2m_{\mathrm{leg}}}{M} & 0 & -\frac{E_z}{g} \end{bmatrix} \\ E &= \begin{bmatrix} E_x \\ E_y \end{bmatrix} \\ F[k] &= \begin{bmatrix} -\frac{m_{\mathrm{leg}} z_e}{Mg}\ddot{x}_{\mathrm{sw}}[k] + \frac{m_{\mathrm{leg}}}{M}(x^{\mathrm{base}}_{\mathrm{sup}}[k] + x^{\mathrm{base}}_{\mathrm{sw}}[k]) \\ -\frac{m_{\mathrm{leg}} z_e}{Mg}\ddot{y}_{\mathrm{sw}}[k] + \frac{m_{\mathrm{leg}}}{M}(y^{\mathrm{base}}_{\mathrm{sup}}[k] + y^{\mathrm{base}}_{\mathrm{sw}}[k]) \end{bmatrix} \end{aligned} \tag{5.10}$$

According to the equations above, $D$ and $E$ are constants and can be determined before walking. The only thing to complete a closed loop control is $F[k]$. Therefore, we can rewrite the output $p^{\mathrm{out}}_{\mathrm{base}}$ as

$$\begin{aligned} p^{\mathrm{out}}_{\mathrm{base}}[k] &= p^{\mathrm{out}}_{\mathrm{zmp}}[k] - E - F[k] \\ p^{\mathrm{out}}_{base}[k] &= DX_{\mathrm{base}}[k] \end{aligned} \tag{5.11}$$

This system can be better controlled when we use error as variables since

the primary goal is to minimize the actual ZMP positions and reference ZMP positions. The state variables can be defined as the following:

$$\begin{aligned} \Delta X_{\text{base}}[k] &= X_{\text{base}}[k] - X_{\text{base}}[k-1] \\ \Delta u_{\text{base}}[k] &= u_{\text{base}}[k] - u_{\text{base}}[k-1] \\ X'_{\text{base}}[k] &= \begin{bmatrix} p^{\text{out}}_{\text{base}}[k] \\ \Delta X_{\text{base}}[k] \end{bmatrix} \end{aligned} \tag{5.12}$$

The new transition equations can be defined as

$$\begin{aligned} X'_{\text{base}}[k+1] &= \Phi X'_{\text{base}}[k] + \Gamma \Delta u_{\text{base}}[k] \\ p^{\text{out}}_{\text{base}}[k] &= \tilde{D} X'_{\text{base}}[k] \end{aligned} \tag{5.13}$$

$$\Phi = \begin{bmatrix} I & DA \\ 0 & A \end{bmatrix}, \Gamma = \begin{bmatrix} DB \\ B \end{bmatrix}, \tilde{D} = \begin{bmatrix} I & 0 \end{bmatrix} \tag{5.14}$$

In order to achieve an optimal solution, we must define a cost function,

$$J(x[k], u) = \sum_{\text{i=k}}^{\infty} [X'^{\text{T}}_{\text{base}}[i] Q_b X'_{\text{base}} + W(u[i])] \tag{5.15}$$

$Q_{\text{b}}$ is a $3 \times 3$ symmetric non-negative definite matrix. $\Delta u_{\text{base}}$ is incremental input.

$$\begin{aligned} W[u[i]] &= \Delta u^{\text{T}}_{\text{base}}[i] R \Delta u_{\text{base}} \\ u_{\text{base}}[k] &= K_{\text{sb}} \sum_{\text{i=0}}^{\infty} e_{\text{base}}[i] - K_{\text{xb}} X_{\text{base}}[k] + \sum_{\text{i=1}}^{\text{N}_{\text{L}}} G_b[i] P^{\text{ref}}_{\text{base}}[k+i] \\ e_b ase[k] &= \begin{bmatrix} e_{\text{bx}}[k] \\ e_{\text{by}}[k] \end{bmatrix} \end{aligned} \tag{5.16}$$

In this equation, $R$ is a non-negative definite matrix. $K_{\text{sb}}$, $K_{\text{xb}}$ and $G_{\text{b}}[i]$ are gains calculated from $Q_{\text{b}}$ and $W$ and parameters from $A$, $B$, $D$, $E$ and $F[k]$ from equation (5.9) and (5.10). In other words, $e_b$ represents the ZMP error between ZMP reference and ZMP output from the system. $N_L$ is the number of future information steps that are used to determine current input. $P^{\text{ref}}_{\text{base}}$ is the reference ZMP trajectory with respect to the base frame. The goal is to obtain $K_{\text{sb}}$, $K_{\text{xb}}$and $G_{\text{base}}[k]$. The procedure is derived from [17,39].

$$
\begin{aligned}
K_b &= (H_b + \Gamma^{\mathrm{T}} S \Gamma)^{-1} \Gamma^{\mathrm{T}} S \Phi \\
&= \begin{bmatrix} K_{\mathrm{sb}} & K_{\mathrm{xb}} \end{bmatrix} \\
G_b[i] &= -(H_{\mathrm{b}} + \Gamma^{\mathrm{T}} S \Gamma)^{-1} \Gamma^{\mathrm{T}} ((\Phi + \Gamma K_b)^{\mathrm{T}})^{\mathrm{i}-1} S^{\mathrm{T}} \Gamma
\end{aligned}
\tag{5.17}
$$

$S$ is the solution of the Riccati equation. $Q_b$ is the weighting matrix, which is conventionally defined as $Q_b = diag\{1, 1, 0...0\}$. $H_b$ is also conventionally defined as $H_b = diag\{10^{-6}, 10^{-6}...10^{-6}\}$. To solve for the Riccati equation,

$$
\begin{aligned}
S_k &= \Phi^{\mathrm{T}}(S_{\mathrm{k+1}} - S_{\mathrm{k+1}} \Gamma (\Gamma^{\mathrm{T}} S_{\mathrm{k+1}} \Gamma + H_b)^{-1} \Gamma^{\mathrm{T}} S_{\mathrm{k+1}}) \Phi_{\mathrm{k}} + Q_b \\
S_\infty &= \Phi^{\mathrm{T}}(S_\infty - S_\infty \Gamma (\Gamma^{\mathrm{T}} S_\infty \Gamma + H_b)^{-1} \Gamma^{\mathrm{T}} S_\infty) \Phi_{\mathrm{k}} + Q_b
\end{aligned}
\tag{5.18}
$$

An analytic solution can be achieved.

$$
O = \begin{bmatrix} \Phi^{-1} & \Phi^{-1} \Gamma H_b^{-1} \Gamma^{\mathrm{T}} \\ Q_b \Phi^{-1} & \Phi^{\mathrm{T}} + Q_b \Phi^{-1} \Gamma H_b^{-1} \Gamma^{\mathrm{T}} \end{bmatrix}
\tag{5.19}
$$

$H$ can be decomposed into $O = VDV^{-1}$, where $D$ is a diagonal matrix and $W$ is composed of eigenvectors of $O$.

$$
\begin{aligned}
V &= \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \\
S_\infty &= V_{21} V_{11}^{-1}
\end{aligned}
\tag{5.20}
$$

The details on generating current ZMP positions will be discussed in the following chapter. Together, a control system that tries to minimize the error between actual ZMP trajectory and reference ZMP trajectory is designed.

# CHAPTER 6

# SIMULATION TECHNICAL

## 6.1 Reemc Model

Figure 6.1: The Reem-C Robot

The experiment is based on the Reemc Robot produced by PAL. This robot is shown in figure 6.1. As a bipedal humanoid robot, it has two arms and two legs. It has six degrees of freedom for both feet, seven degrees of freedom for both legs, two degrees of freedom for the heads and two degrees of freedom for the torso. Also, the root or base frame of the robot has six degrees of freedom since it is in $SE(3)$. In total, the robot has thirty-six degrees of freedom. For simplicity, since we are only interested in the leg locomotion, we set the torso joints and head joints to be fixed. The robot now has thirty two degrees of freedom. The first step is to grab all the specific data for modelling, which is specified in Table 6.1.

Table 6.1: Parameters of robot model

| Variable | Value |
|---|---|
| $l_{\mathrm{s}}$ | 0.35m |
| $l_{\mathrm{t}}$ | 0.35m |
| $l_{\mathrm{f}}$ | 0.224m |
| $u$ | 0.147m |
| $\eta$ | 0.5 |
| $m_{\mathrm{leg}}$ | 16kg |
| $M_{\mathrm{tot}}$ | 80kg |

This model is described in URDF format and can be uploaded in ROS. In detail, we replace the hand with a rectangular box for simplicity, since we are not going to use hands. This will reduce the DOF by replacing the fingers.

In **Chapter 2**, we introduced that our simplified model has six DOF in configuration space. Here, we can formulate the six DOF for Reemc. Since the $y$ components of both feet are fixed and the rotation is fixed to the y axis for both feet, this will reduce the DOF for both feet to three. The inverse kinematic package and Jacobian calculation in later sections will still be based on six DOF for each leg since the legs have six DOF in configuration space. The walking parameters will be determined during simulation, i.e., **Chapter 7**.

## 6.2 Real Time ZMP Position

In **Chapter 2**, we have already discussed the mathematical expression for ZMP positions. In order to calculate the ZMP positions for a walking robot, for example, the one we will use in the next chapter, we need to know the mass, position, linear velocity and linear acceleration for all bodies of the robot. In real time, it is very computationally consuming since the robot has almost a hundred parts including small sensors. Also, we need to get the acceleration, which requires each part of the robot being installed an acceleration sensor. Most robots do not have an acceleration sensor for each body, which makes the job more complicated. An alternative needs to be found.

Since the feet are the only thing in contact with the environment and support the robot, as discussed in [40–42], if we can know the forces and

torques at the ankles of the robot, we can calculate the ZMP positions accordingly [43].

There are two sensors mounted in Reemc, an IMU sensor, which detects the linear velocity and linear acceleration in 3D environment [44]. Another is the force torque sensor (FTS). Sensors which are mounted at the sole of the foot to detect the center of feet are not possible in this case. The more practical use of sensors are FTS in Gazebo. From FTS, we can determine torques and forces at each ankle, namely $\vec{F}_{\text{left}}$, $\vec{F}_{\text{right}}$, $\vec{\tau}_{\text{left}}$ and $\vec{\tau}_{\text{right}}$, with each vector has a dimension of three. The ZMP position can now be formulated as follows for one ankle:

$$\begin{aligned} X_{\text{zmp}} &= \frac{\tau_y^{\text{pitch}}}{F_z} + \frac{z_{\text{ankle}} F_x}{F_z} \\ Y_{\text{zmp}} &= -\frac{\tau_x^{\text{pitch}}}{F_z} + \frac{z_{\text{ankle}} F_y}{F_z} \end{aligned} \tag{6.1}$$

When measuring the ZMP for a biped robot, we need to do a weighted average of the two ZMPs from the two ankles [45, 46].

$$\begin{aligned} X_{\text{zmp}} &= \frac{\tau_y^{\text{right}} + \tau_y^{\text{left}} + z_{\text{ankle}}^{\text{right}} F_x^{\text{right}} + z_{\text{ankle}}^{\text{left}} F_x^{\text{left}}}{F_z^{\text{right}} + F_z^{\text{left}}} \\ Y_{\text{zmp}} &= -\frac{\tau_x^{\text{right}} + \tau_x^{\text{left}} - z_{\text{ankle}}^{\text{right}} F_y^{\text{right}} - z_{\text{ankle}}^{\text{left}} F_y^{\text{left}}}{F_z^{\text{right}} + F_z^{\text{left}}} \end{aligned} \tag{6.2}$$

In real time, the variables on the right-hand sides of equation (6.2) are easy to get; thus, calculating the actual ZMP positions is quick and accurate. This design fully meets the requirement for high-speed calculation and real-time control. Both FTS and IMU sensors can be designed and attached to the robot as Gazebo Plugin in simulation.

## 6.3 Numerical Solver

In previous chapters, we introduced several nonlinear equations, i.e., equations (3.16), (4.1) and (4.2). Analytic solution is very hard to achieve. Therefore we use IPOPT as nonlinear solvers to solve this this issues. The IPOPT is a nonlinear optimizer, which is fast to run and easy to implement [47]. In

our case, this is to solve the following problem [48]:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \|f(x)\| \\ \text{s.t.} \quad & g(x) = 0 \\ & x^L \leq x \leq x^U \end{aligned} \tag{6.3}$$

In the above problem, $g(x)$ is the set of constraints, for example, (4.1) and (4.2). The mathematical background can be found in [47] and will not be addressed here. We will discuss some implementation detail here. The nonlinear optimizer works best when it is given a good boundary and initial value. To get the best solution, we first simulate and use nonlinear solver in Matlab to generate a guide for boundary conditions and initial values. After generating these values, we calculated the Jacobian and Hessian of the original nonlinear equations. In order to solve for unknowns, we set $f(x)$ to be a vector of unknown variables.

Another important aspect of the solution is to give a good tolerance. The tolerance can be set as small as possible, yet sometimes it gives solutions that are not possible. We settled the tolerance to be in millimeter scale, i.e., one millimeter. Comparing to the scale of the robot, which is in meters, we can see that the tolerance is relatively small and can be neglected. The error will be detailed in **Chapter 7** when a simulation is run.

## 6.4 Inverse Kinematic

The inverse kinematic is to solve from end effector positions to joint angles. The mathematical formulation is outlined in **Chapter 4** of [49]. Here there are several options for inverse kinematic of Reemc Robot, which are numerical solution, analytic solution and sampling based solution. The first two methods are quite popular; however, the third will still provide some insights into inverse kinematic formulation.

### 6.4.1 Numerical Solution

There are many ways to solve the inverse kinematic numerically. Lots of open source libraries are provided online. We can even use IPOPT from the

previous section to solve this problem. Here we will discuss an open source library called TRAC-IK [50]. This package is implemented based on the open source Orocos Kinematics and Dynamics Library (KDL). This package has fast convergence speed and smaller error tolerance. In detail, we implemented this to our Reemc Module to solve the joint angles numerically. However, this inverse kinematic solver may fail under some circumstance. A method to work around this issue is to repeat the solving process several times. If the solver fails to give a solution, the configuration cannot be achieved. In our case, we found all the feasible configurations can be solved in under three iterations.

### 6.4.2 Analytic Solution

Since the end effector of a leg is in $SE(3)$, it has six DOF. The leg has six joints, which also have six DOF. Therefore, for legs only, we can achieve an analytical solution to this inverse kinematic problem.

Lots of texts have discussed forward kinematics, among which the Denavit-Hartenberg (DH) convention is used most [49]. This convention gives us an easy way to formulate kinematic chains. To move from the base to the end effector, we can derive a homogeneous transform. For each joint, we can derive a transformation matrix. Based on DH convention, the transformation can be derived as follows, where $c$ is an abbreviation for $cos$ and $s$ is an abbreviation for $sin$.

$$ {}^{i}A_{i+1} = \begin{bmatrix} c_{\theta_\mathrm{i}} & -s_{\theta_\mathrm{i}} c_{\alpha_\mathrm{i}} & s_{\theta_\mathrm{i}} s_{\alpha_\mathrm{i}} & a_\mathrm{i} c_{\theta_\mathrm{i}} \\ s_{\theta_\mathrm{i}} & c_{\theta_\mathrm{i}} c_{\alpha_\mathrm{i}} & -c_{\theta_\mathrm{i}} s_{\alpha_\mathrm{i}} & a_\mathrm{i} s_{\theta_\mathrm{i}} \\ 0 & s_{\alpha_\mathrm{i}} & c_{\alpha_\mathrm{i}} & d_\mathrm{i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.4} $$

The overall transformation matrix from base frame to end effector frame can be derived by multiplying all the transformation matrices defined above.

$$ {}^{0}T_3 = {}^{0}A_1 \cdot {}^{1}A_2 \cdot {}^{2}A_3 \tag{6.5} $$

The overall transform can be represented as follows. It is expressed in

terms of Euler angles ZYX convention.

$$
{}^{0}T_{3} = \begin{bmatrix} c_1c_2 & c_1s_2s_3 - c_3s_2 & s_1s_3 + c_1s_2s_3 & p_x \\ c_2s_1 & c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 & p_y \\ -s_2 & c_2s_3 & c_2c_3 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.6}
$$

By equating (6.5) and (6.6), we can derive the mathematical expression for all six joint angles. A detailed derivation can follow similar methods described in [51, 52] where the former derived the analytical solution for a robot arm with five DOF and the later derived the analytical solution for six DOF. The details of the derivation are too long and beyond our focus. Thus, the derivation and solution will not be addressed here.

The advantage for analytic solution is that the result can be calculated extremely fast. However, the difficulty lies in that the DOF of the end effector must be the same as the DOF of the robot arm, i.e., the leg in our case.

### 6.4.3 Sampling Based Solution

There are many sampling based strategies for path planning [53]. We can also use similar methods for inverse kinematic. This method is flexible and can solve for position only or position and orientation both. This method is fast and easy to implement when the system does not require high level of accuracy and can tolerate some error.

To sample the configuration space (C-Space), we give each joint a random value and record the corresponding end effector position and orientation. We may first generate a reachability database using octomap and then build a kD-tree inside each cell from the octomap [54]. This can also be viewed as extending the database for a limb from $R(3)$ to $SE(3)$. Here we will first give the algorithm for building octomap, i.e., the database in $R(3)$. Suppose $N_{\text{SAMPLE}}$ is needed.

**Algorithm 1** Generate Position Orientated Database

```
1: procedure GENERATE DATABASE()
2:     Database D
3:     while SamplesGenerated ≤ N_SAMPLE do
4:         for each q_i in q from C-Space do
5:             Random Generate(q_i)
6:         end for
7:         p(x, y, z) = Forward Kinematics(q);
8:         cell = D.find(p)
9:         cell.insert(pair(p, q))
10:    end while
11:    return D
12: end procedure
```

The database generated above provides the necessary condition for reachibility. The $p$ in each cell will be sorted using kD-tree data structure. To inquire about the position, we may simply locate the cell where the inquiry point stays and find the nearest neighbor in the kD-tree of the cell. The joint angles can be found according to the pair.

To inquire about position only, we only need 3D-Tree. However, with the strong capability of kD-tree, we can also add orientation to the kD-tree. The resulting 6D-tree will organize all the data within a cell. To parametrize the orientation with position, we need a good metric in $SE(3)$.

There are many ways to parametrize the orientation in $SE(3)$. However, none of them are perfect. There always exists a singular point for each metric. We can only find a suitable metric that prevents a singular case during the implementation. One way to do this is to use quaternion $Q = (w, x, y, z)$. A detailed definition and properties of quaternion can be found in the Appendix of [55]. For a configuration $q = (X, R) \in SE(3)$, the distance can be formulated as follows [56]:

$$\rho(q_0, q_1) = w_t \|X_0 - X_1\| + w_r \|f(R_0, R_1\|) \tag{6.7}$$

The rotation distance and translation distance are weighted by parameter $w_r$ and $w_t$. The translation distance can be easily calculated by using Euclidean distance. However, rotation distance where $R_0, R_1 \in SO(3)$ needs to

be measured approximately. Some metrics have been proposed [57]. Choudhury and Lynch used the following metric to measure rotation distance [58]:

$$\rho_r = \| \log(Q_1^{-1} Q_2) \| \tag{6.8}$$

In this equation, $Q_1 = (w_1, x_1, y_1, z_1)$ and $Q_2 = (w_2, x_2, y_2, z_2)$ are unit quaternions. Kuffner provided an alternative of calculating approximate distance in [56] as follows:

$$\begin{aligned} \rho_r &= w_r(1 - \|\lambda\|) \\ \lambda &= Q_1 \cdot Q2 = w_1 w_2 + x_1 x_2 + y_1 y_2 + z_1 z_2 \end{aligned} \tag{6.9}$$

This is quite an effective metric since it obeys the triangle inequality and is easy to compute. Another parametrization is Modified Rodrigues parameters, which is parametrized as three variables. The Modified Rodrigues parameters can be transformed from quaternions and to quaternions as follows. Suppose $R = (R_1, R_2, R_3)$ is the Modified Rodrigues arameter and $Q = (w, x, y, z)$ is a unit quaternion.

$$R_1 = \frac{x}{w+1} \quad R_2 = \frac{y}{w+1} \quad R_3 = \frac{z}{w+1} \tag{6.10}$$

The inverse transform can be formulated as follows:

$$\begin{aligned} w &= -\frac{R_1^2 + R_2^2 + R_3^2 - 1}{R_1^2 + R_2^2 + R_3^2 + 1} \quad x = \frac{2R_1}{R_1^2 + R_2^2 + R_3^2 + 1} \\ y &= \frac{2R_2}{R_1^2 + R_2^2 + R_3^2 + 1} \quad z = \frac{2R_3}{R_1^2 + R_2^2 + R_3^2 + 1} \end{aligned} \tag{6.11}$$

The advantage of using Modified Rodrigues parameters is that it has only one singular point. This parameter is frequently used in altitude estimation [59]. With the metrics above, we can define the distance in 6D-tree. Every time we inquire about a configuration, we may first locate the cell according to the position in $\mathbb{R}^3$. The 6D-tree for that cell can be fetched. Using the metric defined, we can find the nearest neighbor in 6D-tree. The search function returns with joint angles from 6D-tree.

The best part is that this method will immediately tell whether a configuration is valid or not by using the octomap, which will avoid some illegal configurations generated by numerical methods. However, this method becomes slow when the resolution of the octomap is set to a value that is too

small. This method also requires space to store the database.

## 6.5 Joint Velocity

In **Chapter 4**, we have already derived the reference joint angles. In this section, we discuss joint velocity. When applying control methods, as in **Chapter 5**, knowing reference angles is sometimes not enough. We may also need joint velocities to command and control the robot. For example, to run simulation in Gazebo, we can also aid controlling the robot by sending joint velocities.

From **Chapter 4**, we have already derived the joint angle trajectories with respect to time, i.e., $x_{\text{left}}(t), x_{\text{right}}(t), x_{\text{trunk}}(t), z_{\text{left}}(t)$ and $z_{\text{right}}(t)$. Now, we can solve for the joint velocities.

We can calculate linear velocity, of end effector from joint velocities [49]. Here we suppose $x$ is the position of the end effector, i.e., foot.

$$\begin{aligned} \mathbf{J_v} &= \frac{\partial x}{\partial t} \\ &= \frac{\partial x}{\partial t}\frac{\partial t}{\partial q} \\ \dot{x} &= \sum_{i=1}^{\text{n}} \frac{\partial x}{\partial q_i}\dot{q}_i \\ &= \mathbf{J_v}\dot{q} \end{aligned} \tag{6.12}$$

The Jacobian $\mathbf{J_v}$ has already been determined by the robot model, which can be derived through forward kinematics. Since the leg has three DOF, which is exactly the same as the DOF of the end effector, the dimension of the Jacobian is $3 \times 3$. That is to say that Jacobian $\mathbf{J_v}$ will be full rank when the leg is not in singular configuration. The inverse of Jacobian and we can solve for the linear velocity and angular velocity as in equation (6.13). Both joint velocity and joint angles will be sent to simulation in next chapter.

$$\dot{q} = {\mathbf{J_v}}^{-1}\dot{x} \tag{6.13}$$

# CHAPTER 7

# SIMULATION RESULT

## 7.1 Walking Parameters

First we need to determine the walking parameters, such as the step length, etc. The variables are listed in table 7.1.

Table 7.1: Parameters of robot walking

| Variable | Value |
|---|---|
| $N_{\text{step}}$ | 12 |
| $t_{\text{start}}$ | 0.7s |
| $t_{\text{s}}$ | 1s |
| $t_{\text{end}}$ | 0.8s |
| $DSR$ | 0.1 |
| $l_{\text{step}}$ | 0.6m |
| $h$ | 0.55m |
| $w$ | 0.23m |

These parameters can be changed if the resulting system can be solved in **Chapter 4**.

## 7.2 Kinematic Solution

This section focuses on the kinematic solution for my planner. As in **Chapter 3** and **Chapter 4**, the goal is to generate reference ZMP trajectories, reference trajectories for three masses and reference joint angles.

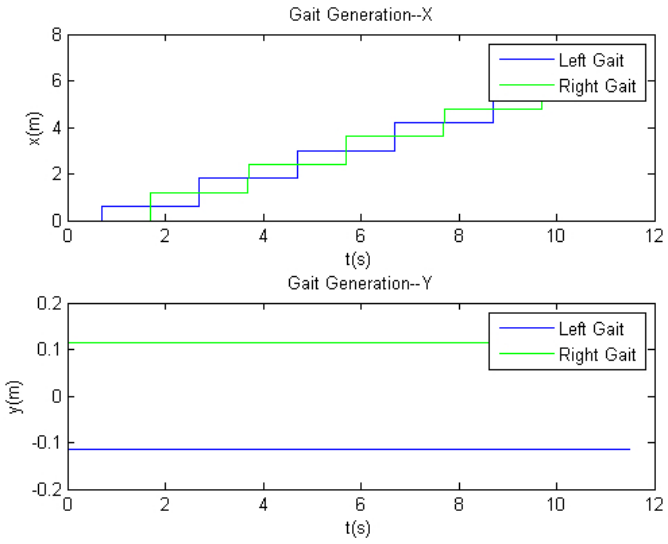



Figure 7.1: Gait generation

### 7.2.1 Gait Generation

As shown in figure 7.1, the left leg and right leg move alternately, yet they only move in x direction. The positions in y direction are kept constant. This position will only change when the robot wants to make a turn.

### 7.2.2 ZMP Trajectory Generation

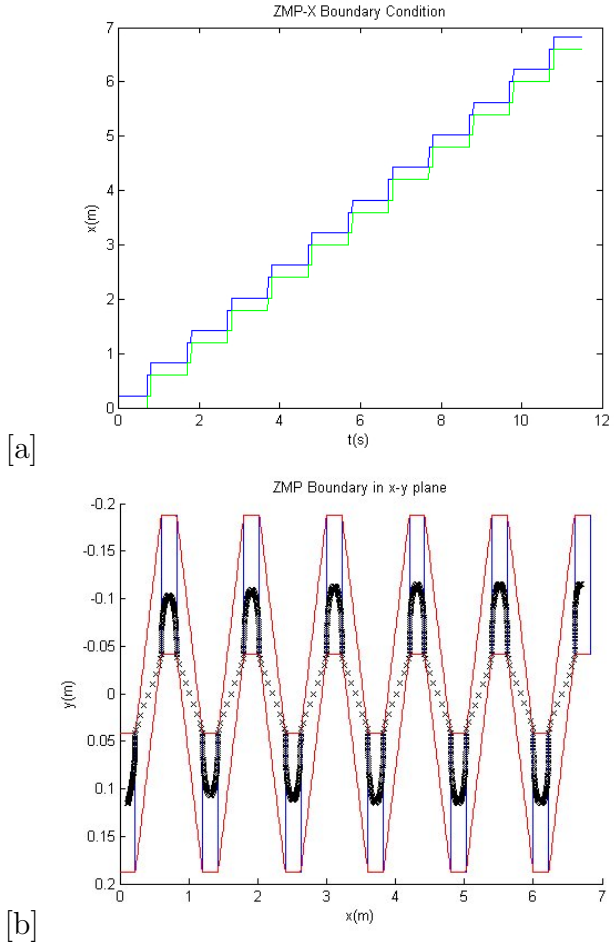
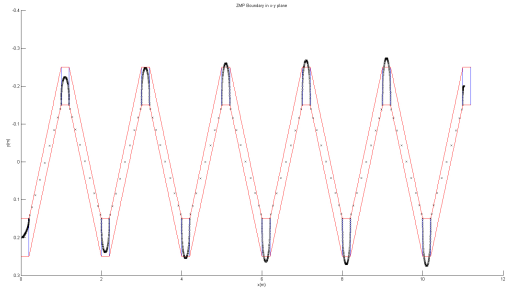



Figure 7.2: (a) represents the ZMP boundary with respect to time; (b) represents the ZMP boundary in XY plane

As shown in figure 7.2, ZMP boundary conditions can be generated according to the gaits planned in the previous section. The figure to the left is the ZMP condition with respect to time while the right one represents ZMP condition in XY plane where the red and blue lines specify the region of the support polygon.

[a]



[b]

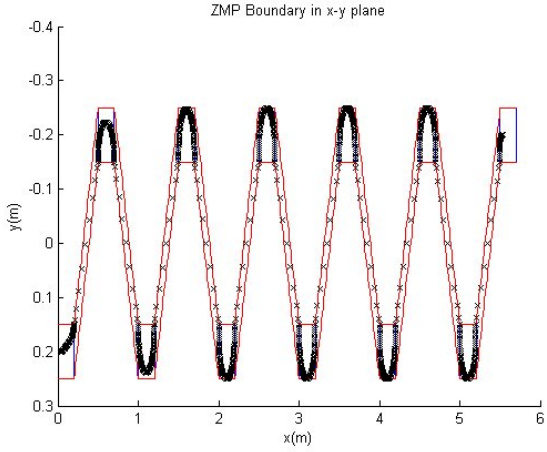
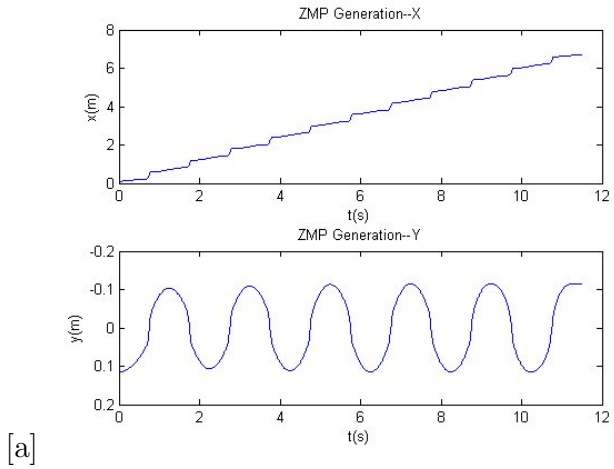


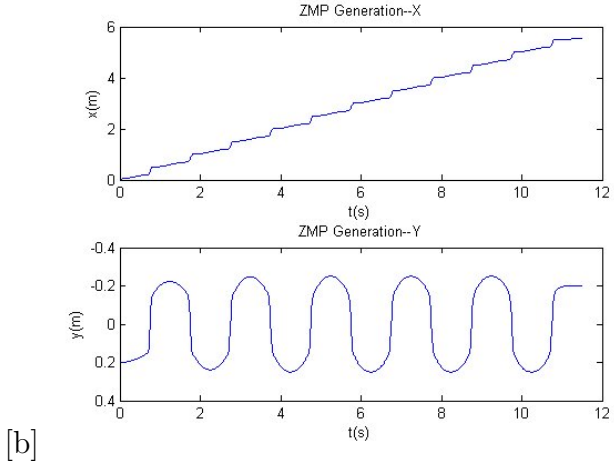

Figure 7.3: (a) represents the ZMP boundary with respect to time; (b) represents the ZMP boundary in XY plane

By using interpolation methods discussed in **Chapter 3**, we can calculate ZMP trajectory according to the ZMP conditions. However, sometimes ZMP may exceed the boundary conditions as illustrated in figure 7.3 a. The interpolation points may stay outside the support polygon.

Using the correction method, we achieved 7.3 b. As shown, all of the ZMP points now stay inside the support polygon. Therefore, if we correct the reference ZMP trajectories, the robot can maintain balance.




[a]

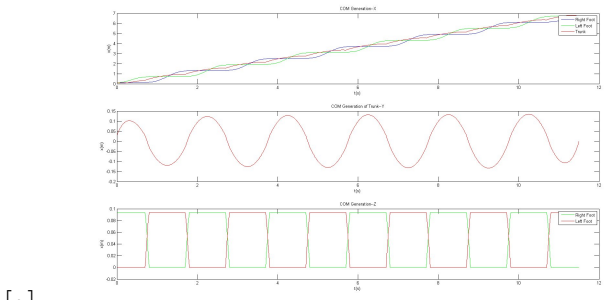



[b]

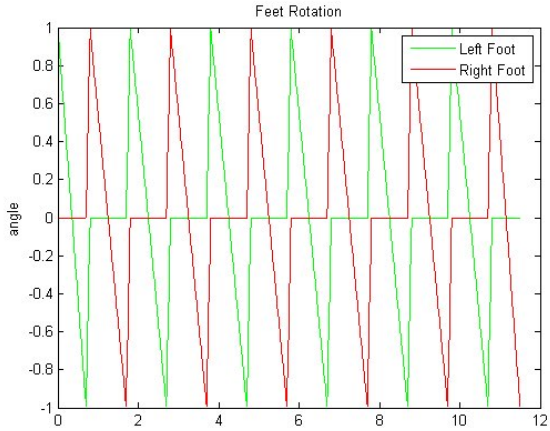
Figure 7.4: (a) represents the ZMP trajectory with respect to time; (b) represents the ZMP trajectory after correction

Figure 7.4 shows the ZMP trajectories with respect to time. Figure 7.4 (a) shows the ZMP trajectories without correction, while (7.4 (b) shows the ZMP trajectories after correction.

### 7.2.3 COM Generation



[a]




[b]

Figure 7.5: (a) shows X and $y$ components of COM trajectories; (b) shows the $z$ component of feet masses

In **Chapter 4**, methods have been introduced to calculate COM trajectories. This is to generate trajectory for all three masses with respect to time. We can now use this result to calculate reference joint angles in the next section.

## 7.3 Dynamic Solution

From previous sections, we can calculate reference joint angles using inverse kinematic. The first step was realized in Gazebo, which is a physical simulator in ROS. Using planning methods, we change the coordinates of the trunk to near positions to keep balance. The temporary result can be viewed in `https://www.youtube.com/watch?v=vXrrb168jiA&list=PLXQvyFrEzAia8z5QDql-vuowaboL7xrYy&index=2`.

# CHAPTER 8

# CONCLUSION AND FUTURE WORK

The goal of this thesis is to add feet rotation to biped robot walking. In our heel-toe motion planner, we first design the gait for each step. Then we can achieve the ZMP boundary conditions where all ZMP points should stay inside the support polygon. From ZMP trajectories, we can generate the COM trajectories. At this point, we have calculated all reference ZMP trajectory and reference joint angles. The next step is to control the robots to walk in a physical simulator. By using controlling methods, we can cancel out errors brought by modelling. The current result shows that the robot has made its first step. With this first step, the robot can continue walking with feet rotation.

This method is fast and easy to implement and it can be fit into any biped robots. However, there are some drawbacks in the current method. First, all walking parameters should be chosen carefully. In order to solve nonlinear equations in **Chapter 4**, parameters including step length and trunk height have to be given appropriate values so that the nonlinear equations can be solved. Second, in trying to solve COM trajectory, the masses for three-mass model have to be appropriate chosen. The values we chose are described in **Appendix A**.

Future work will focus on multiple aspects. Moving from a kinematic solution to a dynamic solution is another tough problem. In our planner, we simply planned the robot to walk in a straight line. Future work may include designing a planner that can enable the robot to make a turn. Simple actions that humans can easily do need to be planned carefully for biped robots. In addition, our planner currently had not been tested when there is some disturbance. To design a more robust control system, the robot has to function under disturbance, such as wind. Finally, due to instability of Gazebo, we may also try to simulate in a real robot to improve our planner.

# APPENDIX A

# ZMP CORRECTION AND STABILITY

## A.1 ZMP Correction

ZMP correction is used to correct ZMP trajectory when the ZMP goes outside the support polygon. The system cannot tolerate such an error since it exceeds the ZMP boundary. The original quadratic function can be expressed as a quadratic function.

$$y(t) = at^2 + bt + c \tag{A.1}$$

where $a$, $b$ and $c$ are constants. The quadratic function can have its vertex at position $(-\frac{b}{2a}, c - \frac{b^2}{4a})$. Therefore, we suppose the correction function $\delta(t)$ must have the following properties:

1) If the boundary condition is satisfied, $\delta(t) = 0$. If the boundary condition is not satisfied, $\delta(t)$ will be added to the original fitting function.

2) The boundary condition can be expressed as: $\dot{\delta}(t = 0) = 0$, $\dot{\delta}(t = \Delta_{01}) = 0$, $\delta(t = 0) = 0$ and $\delta(t = \Delta_{01}) = 0$

3) $\delta(t)$ is smooth upon region $[0, \Delta_{01}]$.

4) $|\delta(t = -\frac{b}{2a})| < |\max(error)| = |y^* - y_{\max\,/\,\min}|$

A simple function to use is quintic function. Using the properties above, such a function can be found easily. This correction function can be assured to have the ZMP bounded for the following reasons:

1) When solving $\dot{\delta} = 0$, the solution should be symmetric about the real axis. By using a quintic function, $\dot{\delta} = 0$ has three solutions. According to property 2), two solutions have already been designed. The last one must be real.

2) According to property 2), $\delta(t = 0)$ and $\delta(t = \Delta_{01})$ must be either maximum or minimum. However, if no maximum or minimum exists between $(0, \Delta_{01})$, $\delta(t = 0)$ and $\delta(t = \Delta_{01})$ must be maximum, minimum or minimum,

maximum. This conflicts with property 2) that $\delta(t = 0) = \delta(t = \Delta_{01}) = 0$ except that $\delta(t) = 0$. However, $\delta(t) = 0$ conflicts with property 4). So there must be a maximum/Minimum point between $(0, \Delta_{01})$.

Now we can conclude that the quintic function meets such a requirement. This function can also be applied when any deviation occurs. For example, when the y-zmp or the velocity is not at the designated value, an offset can be applied to the function to converge to the designed value, thus completing a closed loop control.

## A.2 Stability for End Phase

End-Phase is unique. The function can be expressed as

$$y(t) = A + Be^{-\alpha t} \tag{A.2}$$

The first order derivative of this function will converge to zero when time goes to infinity and when $\alpha > 0$. With small B, the ZMP could still reside in the support polygon at maximum overshoot. This happens when the first derivative of the $y$ component of ZMP is not too large.

# REFERENCES


[1] J. Lack, "Planar multicontact locomotion using hybrid zero dynamics," M.S. thesis, Georgia Institute of Technology, Atlanta, 2013.

[2] F. Iida and R. Tedrake, "Minimalistic control of a compass gait robot in rough terrain," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 1985–1990.

[3] R. Wittmann, A. C. Hildebrandt, A. Ewald, and T. Buschmann, "An estimation model for footstep modifications of biped robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 2572–2578.

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

[5] J. Englsberger, C. Ott, M. A. Roa, A. Albu-Schffer, and G. Hirzinger, "Bipedal walking control based on capture point dynamics," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2011, pp. 4420–4427.

[6] A. Farooq and D. J. N. Limebeer, "Path following of optimal trajectories using preview control," in *Proceedings of the 44th IEEE Conference on Decision and Control*, Dec 2005, pp. 2787–2792.

[7] K. Takaba, "A tutorial on preview control systems," in *SICE 2003 Annual Conference (IEEE Cat. No.03TH8734)*, vol. 2, Aug 2003, pp. 1388–1393 Vol.2.

[8] T. B. Sheridan, "Three models of preview control," *IEEE Transactions on Human Factors in Electronics*, vol. HFE-7, no. 2, pp. 91–102, June 1966.

[9] H. Audren, J. Vaillant, A. Kheddar, A. Escande, K. Kaneko, and E. Yoshida, "Model preview control in multi-contact motion-application to a humanoid robot," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 4030–4035.

[10] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 42–56, Jan 2003.

[11] J. Englsberger, C. Ott, and A. Albu-Schffer, "Three-dimensional bipedal walking control based on divergent component of motion," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 355–368, April 2015.

[12] M. Vukobratovic and B. Borovac, "Zero-moment point thirty five years of its life," *Int. J. Human. Robot.*, vol. 1, p. 157, 2004.

[13] P. Sardain and G. Bessonnet, "Forces acting on a biped robot. center of pressure-zero moment point," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans archive*, vol. 34, no. 5, pp. 630–637, September 2004.

[14] T. Yoshikawa and O. Khatib, "Compliant humanoid robot control by the torque transformer," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 3011–3018.

[15] T. Takenaka, T. Matsumoto, and T. Yoshiike, "Real time motion generation and control for biped robot -1st report: Walking gait pattern generation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 1084–1091.

[16] K. Nishiwaki and S. Kagami, "High frequency walking pattern generation based on preview control of zmp," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 2667–2672.

[17] S. Kajita, M. Morisawa, K. Harada, K. Kaneko, F. Kanehiro, K. Fujiwara, and H. Hirukawa, "Biped walking pattern generator allowing auxiliary zmp control," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 2993–2999.

[18] D. Stonier and J. H. Kim, "Zmp analysis for realisation of humanoid motion on complex topologies," in *2006 IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, Oct 2006, pp. 247–252.

[19] C. Zhu and A. Kawamura, "Walking principle analysis for biped robot with zmp concept, friction constraint, and inverted pendulum model," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 1, Oct 2003, pp. 364–369 vol.1.

[20] K. Harada, S. Kajita, F. Kanehiro, K. Fujiwara, K. Kaneko, K. Yokoi, and H. Hirukawa, "Real-time planning of humanoid robot's gait for force controlled manipulation," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 1, April 2004, pp. 616–622 Vol.1.

[21] S. Tonneau, N. Mansard, C. Park, D. Manocha, F. Multon, and J. Pettr, "A reachability-based planner for sequences of acyclic contacts in cluttered environments," *in ISSR*, 2015.

[22] H. H. Zhao, W. L. Ma, A. D. Ames, and M. B. Zeagler, "Human-inspired multi-contact locomotion with amber2," in *2014 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, April 2014, pp. 199–210.

[23] S. Tonneau, A. Prete, J. Pettr, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic planner for multiped robots," *Rpport LAAS n 16024*, 2016.

[24] M. Kudruss, M. Naveau, O. Stasse, N. Mansard, C. Kirches, P. Soueres, and K. Mombaur, "Optimal control for whole-body motion generation using center-of-mass dynamics for predefined multi-contact configurations," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, pp. 684–689.

[25] A. Werner, B. Henze, D. A. Rodriguez, J. Gabaret, O. Porges, and M. A. Roa, "Multi-contact planning and control for a torque-controlled humanoid robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 5708–5715.

[26] D. Hadfield-Menell, A. D. Dragan, P. Abbeel, and S. J. Russell, "Cooperative inverse reinforcement learning," *CoRR*, vol. abs/1606.03137, 2016. [Online]. Available: http://arxiv.org/abs/1606.03137

[27] A. Venkatraman, R. Capobianco, L. Pinto, M. Hebert, D. Nardi, and J. A. Bagnell, *Improved Learning of Dynamics Models for Control*. Cham: Springer International Publishing, 2017, pp. 703–713. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-50115-4_61

[28] J. H. Choi and J. W. Grizzle, "Planar bipedal walking with foot rotation," in *Proceedings of the 2005, American Control Conference, 2005.*, June 2005, pp. 4909–4916 vol. 7.

[29] T. Sato, S. Sakaino, and K. Ohnishi, "Real-time walking trajectory generation method with three-mass models at constant body height for three-dimensional biped robots," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 2, pp. 376–383, Feb 2011.

[30] T. Kim and J.-W. Kim, "Planning walking patterns of a biped robot with udeas optimization," in *2007 International Conference on Control, Automation and Systems*, Oct 2007, pp. 2693–2698.

[31] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to Humanoid Robotics*. New York, NY: Springer, 2013.

[32] Y. F. Li and X. B. Chen, "On the dynamic behavior of a force/torque sensor for robots," *IEEE Transactions on Instrumentation and Measurement*, vol. 47, no. 1, pp. 304–308, Feb 1998.

[33] D. Galdeano, A. Chemori, S. Krut, and P. Fraisse, "Optimal pattern generator for dynamic walking in humanoid robotics," in *10th International Multi-Conferences on Systems, Signals Devices 2013 (SSD13)*, March 2013, pp. 1–6.

[34] S. Feng and Z. Sun, *Biped Robot Walking Using Three-Mass Linear Inverted Pendulum Model*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 371–380. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88513-9_40

[35] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the Next Millennium (Cat. No.01CH37180)*, vol. 1, 2001, pp. 239–246 vol.1.

[36] L. Lanari, S. Hutchinson, and L. Marchionni, "Boundedness issues in planning of locomotion trajectories for biped robots," in *2014 IEEE-RAS International Conference on Humanoid Robots*, Nov 2014, pp. 951–958.

[37] S. Shimmyo, T. Sato, and K. Ohnishi, "Biped walking pattern generation by using preview control based on three-mass model," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 11, pp. 5137–5147, Nov 2013.

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

[39] J. McMahan, "Linear systems and optimal control condensed notes," November 2010. [Online]. Available: http://www4.ncsu.edu/~jamcmaha/control/control.pdf

[40] A. Goswami, "Postural stability of biped robots and the foot rotation indicator point," *International Journal of Robotics Res.*, vol. 18, no. 6, pp. 523–533, 1999.

[41] T. Sugihara and Y. Nakamura, "High mobility control of humanoid robots based on an analogy of zmp-cog model and carted inverted pendulum model," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 34, no. 5, pp. 1244–1245, September 2004.

[42] T. Sugihara and Y. Nakamura, "High mobility control of humanoid robots based on an analogy of zmp-cog model and carted inverted pendulum mode," in *Robotics Society of Japan annual conference*, vol. 24, no. 1, Japan, 2006, pp. 74–83.

[43] M. Shimojo, T. Araki, A. Ming, and M. Ishikawa, "A zmp sensor for a biped robot," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 1200–1205.

[44] G. Angelosanto, "Kalman filtering of imu sensor for robot balance control," M.S. thesis, Massachusetts Institute of Technology, Cambridge, 2008.

[45] C. Yuan, R. J. Yan, J. Wu, S. H. Kim, K. S. Shin, and C. S. Han, "Design and evaluation of a three-axis force/torque sensor for humanoid robot balance control," in *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, Oct 2014, pp. 227–232.

[46] C. Yuan, L. P. Luo, K. S. Shin, and C. S. Han, "Design and analysis of a 6-dof force/torque sensor for human gait analysis," in *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*, Oct 2013, pp. 1788–1793.

[47] A. W. L. T. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[48] Y. Kawajir, F. Margot, C. Laird, S. Vigerske, and A. Wachter, "Introduction to ipopt: A tutorial for downloading, installing and using ipopt," Massachusetts Institute of Technology, MA, Tech. Rep. 24226, Nov. 2013.

[49] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Dynamics and Control*. Hoboken, NJ: John Wiley Sons, 2004.

[50] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, pp. 928–935.

[51] A. Q. Gan, E. Oyama, E. M. Rosales, and H. Hu, "A complete analytical solution to the inverse kinematics of the pioneer 2 robotic arm," *Robotica*, vol. 23, pp. 123–129, 2005.

[52] L. Nie and Q. Huang, "Inverse kinematics for 6-dof manipulator by the method of sequential retrieval," in *Proceedings of 2012 International Conference on Mechanical Engineering and Material Science, 2012. MEMS 2012.*, Nov. 2012, pp. 255–258.

[53] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, June 2016.

[54] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: an efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 2, pp. 189–206, Apr. 2013.

[55] A. Principles of Robot Motion: Theory and Implementation, *Howie Choset and Kevin Lynch and Seth Hutchinson and George Kantor and Wolfram Burgard and Lydia Kavraki and Sebastian Thrun.* Cambridge, MA: MIT Press, 2005.

[56] J. J. Kuffner, "Effective sampling and distance metrics for 3d rigid body path planning," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 4, April 2004, pp. 3993–3998 Vol.4.

[57] B. C. Andrei and K. R. Vijay, "Euclidean metrics for motion generation on se(3)," *Departmental Papers (MEAM)*, no. 150, 2002. [Online]. Available: http://repository.upenn.edu/meam_papers/150

[58] P. Choudhury and K. Lynch, "Rolling manipulation with a single control," in *Control Applications, 2001. (CCA '01). Proceedings of the 2001 IEEE International Conference on*, 2001, pp. 1089–1094.

[59] P. Wu, Y. Ge, and H. Yuan, "Research on attitude updating algorithm based on modified rodrigues parameters for low cost attitude and heading reference system," in *The 2010 IEEE International Conference on Information and Automation*, June 2010, pp. 1235–1239.