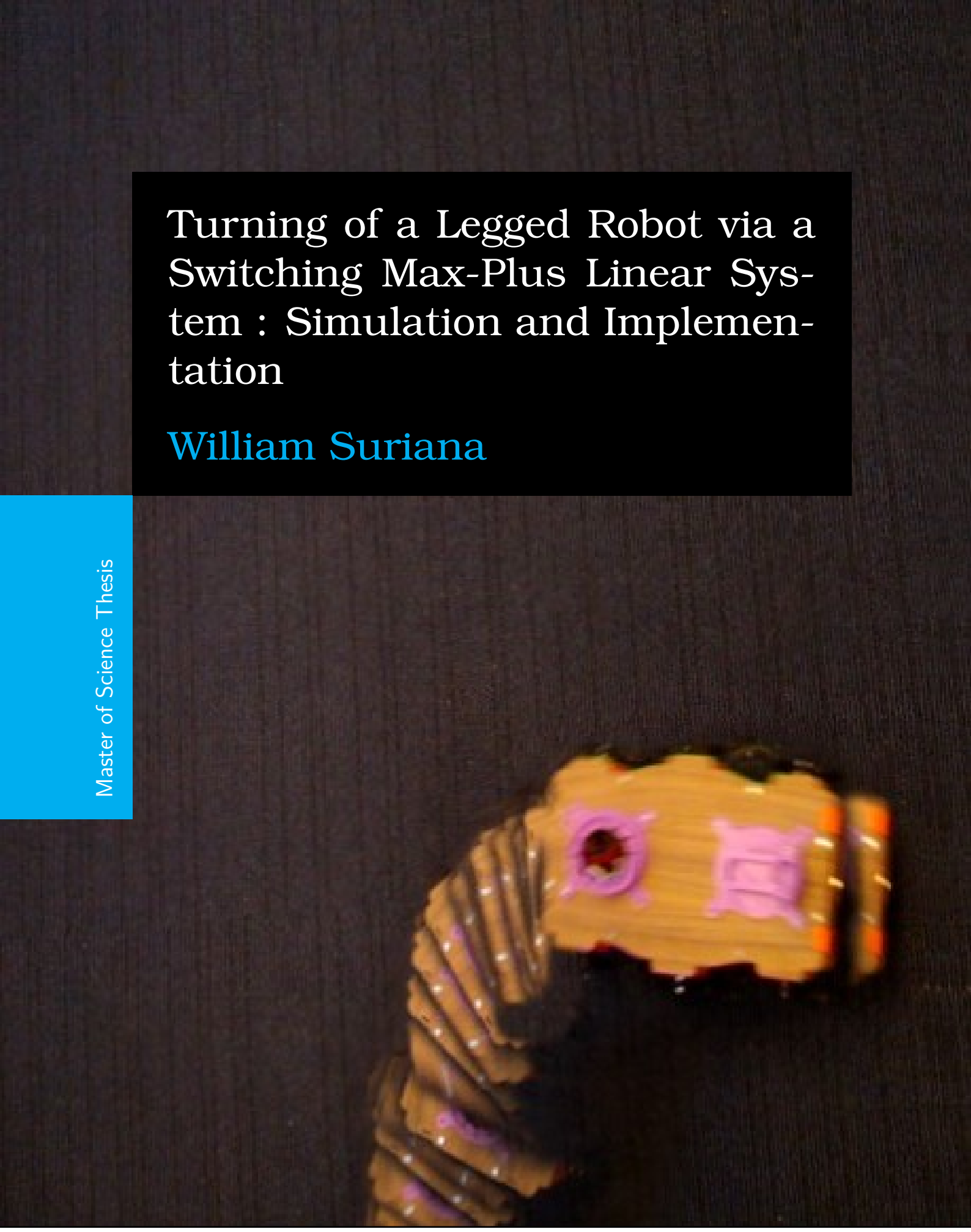
Turning of a Legged Robot via a Switching Max-Plus Linear System : Simulation and Implementation

William Suriana

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Turning of a Legged Robot via a Switching Max-Plus Linear System : Simulation and Implementation

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

William Suriana

October 23, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by The Zebro team and Indonesian Endowment Fund for Education (LPDP). Their cooperation is hereby gratefully acknowledged.

Abstract

Legged locomotion is an example of a discrete event system (DES) with only synchronization and no concurrency. This particular kind of DES can be framed in max-plus algebra, which consists of maximization and addition as its basic operations. In max-plus algebra, those operations are called as max-plus addition and max-plus multiplication respectively, symbolized by $\oplus$ and $\otimes$. Max-plus algebra itself can be used for both scalar and matrices operations [1, 2, 3]. Utilizing the max-plus operations between matrices, max-plus linear (MPL) system can be constructed. An MPL system consists of system matrices, and vectors as both states and inputs. The MPL system can be seen to be equivalent to the state space system in linear algebra.

Switching max-plus linear (SMPL) system is a class of MPL system where the element of the system matrices can be changed by a switching function. There are two variants of SMPL system, the implicit and the explicit SMPL system. In the implicit SMPL system, the current states depend on both previous states and themselves. The current states only depend on the previous states in the explicit SMPL system. The implicit SMPL system can be transformed into explicit SMPL system. Legged locomotion can be modeled as an autonomous implicit SMPL system, as there is no input used. In modeling legged locomotion as an SMPL system, the touchdown and lift-off time instant of every leg is set as the states. The SMPL system is then used to schedule when each leg should touchdown and lift-off the ground. Based on the touchdown and lift-off time instant of a leg, swing and stance period of the leg can be chosen. The swing period of a leg starts with lift-off and ends with touchdown. The stance period of a leg starts with touchdown and ends with lift-off. The total of both periods is called the cycle time. The SMPL system then incorporates the gait and the desired swing and stance period, to schedule the touchdown and lift-off time instant of every leg.

The SMPL system of legged locomotion for straightforward motion is already proposed in [4]. The legged robot which is considered in [4] is called Zebro. Zebro is a robot with six half-circular compliant legs, developed in Technische Universiteit Delft. In this report, the trajectory-following SMPL system for Zebro is proposed. With the trajectory-following SMPL system, a Zebro is able to move forward in all directions by means of the legs scheduling. It means that with the trajectory-following SMPL system, Zebro is able to turn while moving forward. To turn, Zebro needs to have different linear velocity on the left and right side as

it has no steering system. Then, Zebro turns in a similar way as the differential-drive mobile robot. While the Zebro is turning with the trajectory-following SMPL system, the stance period of the inner legs are longer by τ_p seconds compared to the stance period of the outer legs. On the other hand, the swing period inner legs are shorter by τ_p seconds compared to the swing period of the outer legs. The cycle time of every leg is the same consequently. In the trajectory-following SMPL system, the cycle time is the same as the eigenvalue of the system matrices. The eigenvector represents the steady-state condition of the states. The chosen gait and the direction of Zebro can be inferred from the eigenvector.

The motion of the Zebro is predicted by the switching kinematic algorithm in this report. The switching kinematic algorithm is based on the method proposed in [5] to model the kinematics of the wheeled mobile robot. Sheth-Uicker convention is utilized to assign the coordinate frames and describe the relation between coordinate frames. Based on the relation between coordinate frames, first Jacobian matrices of the legs are formed. Each jacobian matrix describes the relationship between the motion of a leg and the motion of Zebro. Here, the considered leg motion is only the leg motion during the stance period. During the swing period, the leg motion does not affect the motion of Zebro. The jacobian matrices are combined in the sensed forward solution, to predict the motion of Zebro if there are several legs on stance period. The switching kinematic algorithm will detect which legs are on the stance period during a duration of time, and the change of angle of the respective legs. The switching kinematic algorithm then forms a suitable sensed forward solution.

The results of the simulation show that the Zebro is able to turn by the trajectory-following SMPL system. To induce turning with a smaller radius, the parameter τ_p is increased. In general, if the τ_p is increased, the turning radius of Zebro is smaller. Still, the Zebro does not always turn to the intended direction. The turning of the Zebro is also not smooth for the quadruped gait. It is caused by the tilting of the Zebro. To predict the trajectory of Zebro, a switching kinematic algorithm can be used. The switching kinematic algorithm can predict the trajectory of Zebro if a certain lower boundary of τ_p is exceeded. The Zebro can also turn to the intended direction if that lower boundary is exceeded. However, the results of the simulation still cannot be verified by the results of the implementation. The results of the implementation are not conclusive yet.

keywords : max-plus algebra, SMPL, kinematics, Zebro

Table of Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to express my gratitude towards my supervisor dr.ir.T.J.J van den Boom for his assistance during the writing of this thesis. With his advices, there is a clear direction for the progress of my research. Thanks for giving me the opportunity to do this research. Secondly, I would like to thank the Zebro team and dr.ir.E.Hakkennes for letting me to test my research with the DeciZebro. It is such a great experience to really test the designed algorithm with the real Zebro, and to be welcomed as a part of the team. I really hope that, in return, the algorithm in this report can really help the development of the Zebro in the Zebro team. I also would like to give special thanks to Laurens for letting me modify his C++ code, and helped me in applying this research into the DeciZebro.

Furthermore, this research is a part of my study in the TU Delft. Therefore, I would like to thank LPDP (Indonesia Endowment Fund for Education) for granting me a full scholarship to study in the TU Delft. During these two years, I also met a lot of friends which supported me a lot. Therefore, I would like to give a lot of thanks to friends from PK-35, friend from DCSC, my flatmates, my Indonesian friends, and all my friends which might not be included there. I owe you a lot.

Finally, I would like to thank my parents and my family for their continuous support during these two years. Thanks for always supporting me here during my study, and for supporting me to take this option.

Delft, University of Technology — William Suriana
October 23, 2017

“ This too shall pass .”

— *a Persian adage*

Chapter 1

Introduction

1-1 Background

Approximately, half of the earth's surface is inaccessible for wheeled and tracked vehicles[7]. Legged locomotion is more adequate than wheeled and tracked locomotion for various types of surface. Utilizing legged locomotion, a robot can be useful for many tasks, such as disaster recovery, nature exploration, and exploration for mining. Those tasks are performed on uneven and rough surfaces, where wheeled and tracked robots will not be effective. Therefore, there is an importance of developing the legged robot for those kinds of tasks.

There are many legged robots which have already been developed, with various forms [8, 9, 10, 6, 11, 12, 13]. Delft Center for Systems and Control (DCSC) of Delft University of Technology (TU Delft) also have already developed a legged robot which is called Zebro (Zesbenige Robot). Zebro is a six-legged (hexapod) autonomous robot designed on the foundations of RHex [6]. All of Zebro's legs are half-circular, and each having only one independently actuated revolute degree of freedom. With independent actuation of Zebro's legs, various gaits can be utilized by Zebro for its locomotion. The gaits of legged locomotion can be modeled as discrete event systems (DES), due to the intrinsic periodic nature and synchronization requirement of legged locomotion [14].

Modeling DES in conventional algebra results in a nonlinear description of the system. However, there exists a subclass of DES for which this model becomes linear if it is formulated in max-plus algebra [15]. Max-plus algebra uses only maximization and addition for its operations. With these operations, DES with only synchronization and no concurrency or choice occur can be modeled [15]. These subclass of DES are called max-plus linear (MPL) DES. Moreover, MPL systems can be extended to switching max-plus linear (SMPL) systems where different modes of operation can be modeled [16]. SMPL systems are already used to model the legged locomotion of Zebro [14, 4]. With SMPL systems, several gaits can be included in the model as a different mode of operation for each gait [14, 4].

With the development of Zebro, it is hoped that Zebro can navigate autonomously from a starting point to a goal, following a trajectory. However, there is no research yet about

trajectory-following navigation of Zebro. To construct a trajectory-following navigation strategy for Zebro, turning method of Zebro has to be summarized. There are already some works done regarding the turning method of legged robots, with different kind of methods [8, 9, 10, 6, 11, 12, 13]. Due to structural limitation of Zebro, the most probable method of turning is the differential drive, similar with RHex [6].

Differential drive is a commonly-used turning method for some wheeled robots. In differential drive robot, the linear velocity of a side of a wheeled robot is set to be different than the opposite side. With different linear velocity between the right and left side, the wheeled robot will produce turning moment. The same principle can be used by legged robots and will be applied for Zebro. Due to the complex interaction between the legs and the ground, it is hard to model the dynamics for motion prediction of the Zebro. Therefore, kinematics approach will be used to predict the motion of the Zebro.

However, in the remaining system, differential drive is not done by means of the SMPL system. Differential drive is done by introducing offset to the lift-off and touchdown angle of the related legs. There are several disadvantages of this method. First, the turning motion cannot be predicted from the properties of the SMPL system. This disadvantage arises because angle offset is not a part of the SMPL system. Second, the introduced angle offset is bounded by the structure of the half-circular compliant legs. Therefore, to control the trajectory-following Zebro, the control input is bounded by the bound of the offset. Thus, a trajectory-following SMPL system is proposed by this thesis as a candidate method to induce the turning. The relation between the SMPL system and the kinematics of the robot is also established in this thesis. To analyze the performance of the trajectory-following SMPL system, 3D simulation is done using the trajectory-following SMPL system.

1-2 Assumptions

There are several assumptions considered in this thesis :

- **The considered robot is Zebro** The idea in this thesis is proposed with regards to the morphology of Zebro. The idea might not be suitable for another legged robot with different morphology
- **Turning motion is defined as forward-turning motion** The turning-in-place motion is not considered as part of turning in this thesis
- **All legs are utilized in a defined gait** In every defined gait, there is no stationary leg
- **All legs roll without slipping** The slipping kinematics of all legs are not considered in the kinematic approach
- **The robot moves on flat and stationary surface** This assumption is introduced so the basic features of the trajectory-following SMPL system can be explained easier

1-3 Problem Statement

In this thesis, the discussed problems are including :

- **How can a Zebro turn by means of SMPL system?**
- **How can the turning motion of Zebro be predicted?**
- **How is the turning performance of the Zebro?**

The problems are discussed from Chapter 3 until the last chapter. The solutions to the problems will also be explained in those chapters.

1-4 Contribution

The main contribution of this thesis is a new trajectory-following SMPL system. The trajectory-following SMPL system and its properties are explained in chapter 3, section 3-4. To know how the Zebro moves with the trajectory-following SMPL system, the kinematics of the Zebro is explored in chapter 4, section 4-4. The kinematics of Zebro is proposed based on the kinematics of wheeled mobile robot [5], explained in chapter 4, section 4-1 until section 4-3. Then, planar 2D simulation is done in Chapter 4 to analyze the performance of the new SMPL system. The planar 2D simulation results will be compared to the predicted motion using the kinematics of the Zebro.

1-5 Outline

This report is divided into six chapters. Outside of this chapter, the remaining chapters are explained as below :

- Chapter 2 : Max-Plus Algebra and Max-Plus Linear Systems
  In the second chapter, the foundations of max-plus algebra and max-plus linear systems are explained. The foundations of max-plus algebra are including the scalar and matrices operations, the connection with graph theory, and the eigenvalues and eigenvectors. The structure of max-plus linear system is also discussed here.
- Chapter 3 : Scheduling of Legged Locomotion with SMPL System
  The structure of the remaining SMPL system is going to be discussed in the third chapter. The discussion is including the structure of the SMPL system and its properties. Next, the proposed trajectory-following SMPL system with its properties is also explained.
- Chapter 4 : Kinematic Modeling of Zebro
  The kinematics of the Zebro is proposed in the fourth chapter. First, the kinematic modeling method in [5] is discussed. This kinematic modeling method is used as the basis of the kinematic modeling of Zebro. Then, the switching kinematics of Zebro is explained.

- Chapter 5 : Simulation and Implementation
  In this chapter, the results of the simulation and implementation are the main discussion. To explain the results well, the properties of both simulation and implementation are explained first

- Chapter 6 : Conclusion and Future Works
  To finish the report, the conclusion is presented in the sixth chapter. The conclusion summarizes the material in this report. The possible future works are also discussed here.

Chapter 2

Max-Plus Algebra and Max-Plus-Linear Systems

2-1 Introduction

In this chapter, the concept of max-plus algebra is introduced. The introduced concept consists of properties of max-plus algebra, max-plus-linear (MPL) systems, and switching max-plus-linear (SMPL) systems. The purpose is to explain the basic principles behind the SMPL systems of legged locomotion.

Properties of max-plus algebra are first discussed in the section 2-2 . The properties of max-plus algebra which are discussed consist of basic operations, matrix definitions, matrix operations, connection with graph theory, and properties of eigenvalues and eigenvectors in max-plus algebra. In the second section, the properties of MPL and SMPL systems are discussed. The properties of MPL and SMPL systems are explained based on the properties of max-plus algebra.

2-2 Max-Plus Algebra

2-2-1 Basic Operations and Properties

Max-plus algebra is a commutative idempotent semiring [1] which consists of two basic operations, $\oplus$ and $\otimes$ [1, 2, 3]. The operations $\oplus$,$\otimes$:$\mathbb{R}_{\max} \times \mathbb{R}_{\max} \to \mathbb{R}_{\max}$ are defined as:

$$x \oplus y := \max(x, y) \tag{2-1}$$

$$x \otimes y := x + y \tag{2-2}$$

for x,y$\in \mathbb{R}_{\max}$:=$\mathbb{R} \cup \{\infty\}$. In max-plus algebra, the identity element is defined as $e := 0$, and the zero element is defined as ε:=-∞. The addition and multiplication in linear algebra can be compared to $\oplus$ and $\otimes$ in max-plus algebra as there are several similarities between them

[15]. The important difference between linear algebra and max-plus algebra is there is no inverse element in $\oplus$ operation.

Besides the basic operations, power operation also exists in max-plus algebra. The power operation has the highest order of evaluation in max-plus algebra, followed by $\otimes$ and $\oplus$. Power operation is defined as

$$x^{\otimes^r} := rx \tag{2-3}$$

The operations are done with regards to several properties :

- Associativity

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z$$
$$x \otimes (y \otimes z) = (x \otimes y) \otimes z$$

- Commutavity

$$x \oplus y = y \oplus x$$
$$x \otimes y = y \otimes x$$

- Distributivity of $\otimes$

$$x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$$

for $\forall x, y, z \in \mathbb{R}_{max}$.

2-2-2 Matrix Definitions and Operations

As in linear algebra, identity matrix $E \in \mathbb{R}_{\max}^{n \times n}$ and zero matrix $\mathcal{E} \in \mathbb{R}_{\max}^{n \times m}$ are exist in max-plus algebra. They are constructed by max-plus-algebraic identity element and max-plus-algebraic zero element respectively, and defined as

$$[\mathcal{E}]_{ij} = \varepsilon \tag{2-4}$$

$$[E]_{ij} = \begin{cases} e & \text{if i =j} \\ \varepsilon & \text{otherwise} \end{cases} \tag{2-5}$$

with i and j denote the index of row and column of the matrices respectively . The $\oplus$, $\otimes$, and max-plus-algebraic power operation in matrices are defined as

$$[A \oplus B]_{ij} = a_{ij} \oplus b_{ij} := \max(a_{ij}, b_{ij}) \tag{2-6}$$

$$[A \otimes C]_{ij} = \bigoplus_{k=1}^{m} a_{ik} \otimes c_{kj} := \max_{k=1,\dots,m} (a_{ik} + c_{kj}) \tag{2-7}$$

$$\underbrace{D^{\otimes k} := D \otimes D \otimes \dots \otimes D}_{k- \text{ times}} \tag{2-8}$$

for matrices A, B, $D \in \mathbb{R}_{\max}^{n \times m}$ and $C \in \mathbb{R}_{\max}^{m \times p}$. Note that a_{ij}, b_{ij}, c_{ij} denote the element of matrices A, B, C for row i and column j. Exception is made for matrices max-plus-algebraic power of zero, as it is defined as $D^{\otimes^0} := E_n$.

2-2-3 Connection with Graph Theory

Max-plus algebra and graphs are closely related, and equivalency exists between them.[1, 17]. The equivalency particularly exists between max-plus algebra and a class of graph, directed graph. There are properties of directed which are correlated to the properties of max-plus algebra. To explain those properties, a directed graph is defined first

Definition 2-2.1. Directed graph A directed graph $\mathcal{G}$ is an ordered pair ($\mathcal{V}$,$\mathcal{A}$), where $\mathcal{V}$ is a set of elements called nodes and $\mathcal{A}$ is a set of ordered pairs of nodes or arcs

An arc can be denoted as $(i, j) \in \mathcal{A}$, with i as the initial node and j the final node. An arc with same initial and final nodes, (i, i), is called a self-loop arc. Nodes can be sequenced as $(i_1, i_2, ..., i_k)$ with k as the order of the final node. If every arcs (i_k, i_{k+1}) in the sequence of nodes are contained in $\mathcal{A}$, then the sequence of nodes are called a path. If for any two different nodes in a directed graph there exists a path, the directed graph is strongly connected. An arc itself can be assigned a numerical value as its weight. Weighted arcs can be correlated to elements of a matrix in the max-plus algebraic sense. A class of directed graph, precedence graph, is defined to correlate a directed graph and a matrix in the max-plus algebraic sense.

Definition 2-2.2. Precedence graph The precedence graph of a matrix $A \in \mathbb{R}_{\max}^{n\times m}$, denoted by $\mathcal{G}(A)$, is a weighted directed graph with nodes 1,2,...,n and a weighted arc (j, i) for every $a_{ij} \neq \varepsilon$

In a precedence graph $\mathcal{G}(A)$, a weight of a particular path can be calculated by summing all the weights of the arcs that form the path. The weight of the particular path can also be calculated from the correlated max-plus matrix A. The summation of all a_{ij} which corresponds to the arcs in a particular path is equal to the weight of the path. Furthermore, the maximum weight of all paths of $\mathcal{G}(A)$ of length k from initial node j to final node i is calculated as

$$(A^{\otimes k})_{ij} = \bigoplus_{i_1,i_2,...,i_{k-1}} a_{i_1 i_2} \otimes a_{i_2 i_3} \otimes ... \otimes a_{i_{k-1} i_j} \tag{2-9}$$

If a path exists between any two different nodes in a precedence graph, the precedence graph is strongly connected. It is equivalent to having weight for all elements in the matrix A. An important property of matrix A can be explained with this concept

Definition 2-2.3. Irreducibility A matrix $A \in \mathbb{R}_{\max}^{n\times n}$ is called irreducible if its precedence graph $\mathcal{G}(A)$is strongly connected

In the max-plus algebraic sense, the irreducibility of matrix A can be proven by means of max-plus calculation [15]

$$(A \oplus A^{\otimes 2} \oplus ... \oplus A^{\otimes n-1})_{ij} \neq \varepsilon$$

for all i, j with i $\neq$ j. The concept of irreducibility helps us to understand two important theorems in max-plus algebra

Theorem 2-2.1 (**Theorem 3.17 of[1]**). *Consider the following systems of linear equations in the max-plus algebra:*

$$x = A \otimes x \oplus b \tag{2-10}$$

with $A \in \mathbb{R}_{max}^{n\times n}$ *and* b, $x \in \mathbb{R}_{max}^{n\times 1}$. *Now let*

$$A^{\star} := \bigoplus_{p=0}^{\infty} A^{\otimes p} \tag{2-11}$$

If $A^{\star}$ *as the Kleene-star product of* A *exists in* $\mathbb{R}_{max}^{n\times n}$ *then*

$$x = A^{\star} \otimes b \tag{2-12}$$

solves the system of max-plus linear equations $2-10$.

The existence of $A^{\star}$ in $2-11$ depends on the precedence graph $\mathcal{G}(A)$. If there are only diagonal elements of non-positive values or no diagonal element in $\mathcal{G}(A)$, then $A^{\star}$ exists. Moreover, $A^{\star}$ is unique if the diagonal elements of $\mathcal{G}(A)$ are negative. Theory $2-2.1$ can be expanded into

Theorem 2-2.2 (**Theorem 3.20 of [1]**). *If matrix* A *of the precedence graph* $\mathcal{G}(A)$ *has diagonal elements with non-positive values , then*

$$A^{\star} = E \oplus A \oplus A^{\otimes 2} \oplus ... \oplus A^{\otimes n-1} \tag{2-13}$$

where n *is the dimension of* A

Theorem 2-2.2 implies that if matrix A has diagonal elements with non-positive values, then it is irreducible. It can be observed that the calculation of $A^{\star}$ in Theorem 2-2.2 is similar to the calculation proof of irreducible matrix. Hence, for the irreducible A, $A^{\star}$ can be found in finite $p = n-1$. With this bound, calculation of $A^{\star}$ is possible. In general, irreducible matrix itself is an example of nilpotent matrix

Definition 2-2.4. Nilpotent Matrix The matrix $A \in \mathbb{R}_{\max}^{n\times n}$ is called nilpotent if there exists a finite positive integer p_0 , with $p \geq p_0$, such that $A^{\otimes p} = \mathcal{E}$.

2-2-4 Eigenvalues and Eigenvectors

As in linear algebra, eigenvalues and eigenvectors can be found from a square matrix. Eigenvalues and eigenvectors in max-plus algebra are defined as

Definition 2-2.5. Max-plus-algebraic eigenvalues and eigenvectors Let $A \in \mathbb{R}_{\max}^{n\times n}$. If there exist $\lambda \in \mathbb{R}_{\max}$ and $v \in \mathbb{R}_{\max}^{n}$ with $v \neq \mathcal{E}_{n\times 1}$ such that $A \otimes v = \lambda \oplus v$ then we say that λ is a max-plus-algebraic eigenvalue of A and that v is a corresponding max-plus-algebraic eigenvector of A

For every square matrix with entries in $\mathbb{R}_{\max}$, there is always be,at least, a max-plus-algebraic eigenvalue [1]. Different with linear algebra, for $A \in \mathbb{R}_{\max}^{n\times n}$, there might be less than n max-plus-algebraic eigenvalues. For every irreducible matrices, there is only a max-plus-algebraic eigenvalue [1]. As in linear algebra, the max-plus-algebraic multiplication of max-plus-algebraic eigenvectors with a real number will result in max-plus-algebraic eigenvectors.

In max-plus-algebra, there exists a property called cyclicity of a a matrix. Cyclicity of a matrix is related to the max-plus-algebraic eigenvalues, and defined as:

Definition 2-2.6. Cyclicity of a matrix A matrix A is said to be cyclic if there exists a max-plus-algebraic eigenvalue $\lambda \in \mathbb{R}_{\max}$, and integers $c \in \mathbb{N}$ and $k_0 \in \mathbb{N}$ such that

$$\forall p \geq k_0 : A^{\otimes p+c} = \lambda^{\otimes c} \otimes A^{\otimes p} \tag{2-14}$$

The smallest possible c for this equation is called the cyclicity of a matrix A, with k_0 is called the coupling time.

Irreducibility and cyclicity of a matrix are related, as every irreducible matrix is a cyclic matrix [1, 2].

2-3 Max-Plus-Linear Systems

Based on max-plus algebra, a discrete event system (DES) with synchronization and no concurrency can be modeled as max-plus linear (MPL) systems or switching max-plus-linear systems (SMPL). MPL and SMPL systems are similar with a state-space model of a system in linear algebra, with respect to the form. The next states are derived from max-plus addition and multiplication of the previous states, inputs, and the matrices. As in linear algebra, the matrices represent the model of the system. In subsection 2-3-1, MPL systems will be explained in details. Furthermore, SMPL systems will be explained in subsection 2-3-2

2-3-1 Max-Plus-Linear Systems

MPL systems can be divided into period-invariant MPL systems and period-variant MPL systems.In general, period-invariant MPL systems can be modeled as [1, 2, 3]:

$$x(k) = A \otimes x(k-1) \oplus B \otimes u(k) \tag{2-15}$$
$$y(k) = C \otimes x(k) \tag{2-16}$$

with $A \in \mathbb{R}_{\max}^{n \times n}$,$B \in \mathbb{R}_{\max}^{n \times m}$, and $C \in \mathbb{R}_{\max}^{l \times n}$ as the system matrices. The number of states,inputs, and ouputs are represented by, n m, l, respectively. The vector $x(k)$, $x(k-1) \in \mathbb{R}_{\max}^{n \times 1}$ are the current and previous-states vector of the system, respectively. The vector $u(k) \in \mathbb{R}_{\max}^{m \times 1}$ is the input vector of the system. The vector $y(k) \in \mathbb{R}_{\max}^{l \times 1}$ is the output vector of the system. In MPL systems, the elements of the state, input, and output vectors are event times, and k is the cycle counter of the events.

The description of period-invariant MPL systems in equation 2-15 is in explicit form as the value of $x(k)$ does not depend on $x(k)$ itself. If the value of $x(k)$ depends on the $x(k)$ itself, the period-invariant MPL systems are said to be in implicit form. The implicit period-invariant MPL systems are defined as

$$x(k) = A_0 \otimes x(k) \oplus A_1 \otimes x(k-1) \oplus B \otimes u(k) \tag{2-17}$$
$$y(k) = C \otimes x(k) \tag{2-18}$$

with A_0,$A_1 \in \mathbb{R}_{\max}^{n \times n}$ describe as the system matrices that relate $x(k)$ with itself and $x(k-1)$ respectively. The implicit and explicit form of MPL systems are still closely related to each

other. In fact, the implicit form of MPL systems can be transformed into explicit form using Theorem $2-2.1$ and Theorem $2-2.2$. The transformation can be defined as

$$\begin{aligned} x(k) &= A_0 \otimes x(k) \oplus A_1 \otimes x(k-1) \\ &= A_0^{\star} \otimes A_1 \otimes x(k-1) \\ &= A \otimes x(k-1) \end{aligned} \tag{2-19}$$

for period-invariant MPL systems. The period-variant MPL systems are mostly similar with period-invariant MPL systems for both explicit and implicit forms, except the system matrices might change for every cycle k.

2-3-2 Switching Max-Plus-Linear Systems

For a system with different operating modes, SMPL systems can be used to model the DES instead of MPL systems. SMPL systems, for explicit period-invariant form, are defined as [16]

$$x(k) = A^{l(k)} \otimes x(k-1) \oplus B^{l(k)} \otimes u(k) \tag{2-20}$$

$$y(k) = C^{l(k)} \otimes x(k) \tag{2-21}$$

with $A^{l(k)} \in \mathbb{R}_{\max}^{n \times n}$,$B^{l(k)} \in \mathbb{R}_{\max}^{n \times m}$, and $C^{l(k)} \in \mathbb{R}_{\max}^{l \times n}$ as the system matrices for mode *l(k)*. The mode *l(k)* of the system can be determined by a switching function [18]

$$l(k) = \phi(x(k-1), ..., x(0), l(k-1), u(k)) \tag{2-22}$$

With the switching function, the mode of the system can be switched so change in the structure of the system can be modeled. SMPL systems, for implicit period-invariant form, are defined as [16]

$$x(k) = A_0^{l(k)} \otimes x(k) \oplus A_1^{l(k)} \otimes x(k-1) \oplus B^{l(k)} \otimes u(k) \tag{2-23}$$

$$y(k) = C^{l(k)} \otimes x(k) \tag{2-24}$$

The explicit and implicit period-variant SMPL systems have similar forms compared to the period-invariant ones, except the switching function is also determined by the cycle k. The implicit model of SMPL systems also can be transformed into explicit form if Theorem $2-2.1$ is satisfied [18]. The SMPL systems might also subject to change based on the time instant τ at the moment. Then, both explicit and implicit SMPL systems can be defined as

$$x(k|t) = A^{l(k|t)} \otimes x(k-1|t) \oplus B^{l(k|t)} \otimes u(k|t) \tag{2-25}$$

$$y(k|t) = C^{l(k|t)} \otimes x(k|t) \tag{2-26}$$

$$\tag{2-27}$$

and

$$x(k|t) = A_0^{l(k|t)} \otimes x(k|t) \oplus A_1^{l(k|t)} \otimes x(k-1|t) \oplus B^{l(k|t)} \otimes u(k|t) \tag{2-28}$$

$$y(k|t) = C^{l(k|t)} \otimes x(k|t) \tag{2-29}$$

2-4 Summary

In this chapter, the concept of max-plus algebra and its properties are already introduced. The connection of max-plus algebra and graph theory is utilized in the explanation. Based on the max-plus algebra, the MPL and SMPL system are also explained. The knowledge of max-plus algebra, MPL, and SMPL system is essential in understanding the modeling of legged locomotion with SMPL in the next chapter.

Chapter 3

Scheduling of Legged Locomotion with Switching Max-Plus-Linear (SMPL) System

3-1 Introduction

Gait reference generator is a fundamental part in controlling the movement of legged robots. The gait reference generator decides reference path for each leg in a legged robot. The reference path of a leg itself is decided according to how the legged robot must move. By doing so, the gait reference generator enforces the legged robot to follow the desired trajectory. Moreover, the reference path of every leg must be synchronized, so the legged locomotion is kinematically stable. Therefore, synchronization of legs is also an important element of the gait reference generator. Scheduling of legged locomotion is a way to achieve both purposes. Here, SMPL is proposed as an effective scheduling system for legged robots. This chapter will mainly discuss the SMPL system for legged locomotion, both the SMPL for forward-only and trajectory-following locomotion.

To introduce, the existing gait reference generators are discussed in 3-2. There are two gait reference generators explained, central pattern generators (CPGs) and Buehler clock [19, 6]. In section 3-3, the SMPL system for forward-only locomotion is discussed. The discussion is including the properties of the corresponding SMPL system, such as max-plus eigenstructure, coupling time, and gait transition. Next, the SMPL system for trajectory-following locomotion is explained. The relevant properties of this SMPL system are also discussed here. Finally, in section 3-5 the full control structure needed for the legged locomotion system is explained.

3-2 Recent Modeling Method

In this report, there are two events considered during the legs movement. Those events can be explained as

- Touchdown is the event when a leg touches the ground again after it moves in the air
- Lift-off is defined as the event when a leg leaves the ground after it moves on the ground

Between the two events, there are two defined periods. The periods are defined based on the starting and ending events. Stance period of a leg will start with touchdown and end with lift-off of the leg. Swing period of a leg will start with lift-off and end with touchdown of the leg.

3-2-1 Central Pattern Generators (CPGs)

CPGs have been used for many times for legged robots, viewed as a standard model of gait reference generator. There are several useful properties of CPGs as gait reference generator[19]:

1. CPGs model the limit cycles on cross products of circles. Therefore, CPGs are suitable for periodic task such as legged locomotion
2. CPGs are suitable for distributed implementation
3. There are fewer control parameters needed for CPGs
4. CPGs enable the use of feedback signal

In CPGs, the periodic motion is modeled by an abstract phase $\theta_i \in \mathbb{S}^1$ with i denotes the position of the leg and $\mathbb{S}^1$ the circle. The phase θ represents the reference angle which the motor needs to sweep. The range of the phase can extend from 0 to 2π. From this subsection to the end of the chapter, leg angle is assumed to be linear with the motor angle (phase). If there are n legs, the n leg phase states $\theta = [\theta_1 ... \theta_n]^T \in \mathbb{T}^n$ are generated with equation

$$\dot{\theta}(\tau) = V + h(\theta(\tau)) \tag{3-1}$$

where $\mathbb{T}^n$ is the n-torus (the Cartesian product of n circles), $V \in \mathbb{R}^n$ is the phase velocity reference, and function $h(\theta(\tau))$ synchronizes each phase by setting the coupling between each phases. Function $h(\theta(\tau))$ enables the design of a certain gait. With the generated phase θ, reference trajectories of each actuator can be generated with

$$q_{ref}(\tau) = g(p, \theta(\tau)) \tag{3-2}$$

with $q_{ref}(\tau) \in \mathbb{R}^n$ as the reference trajectories of each actuator at current time instant $\tau \in \mathbb{R}$ and p as the set of parameters that modulate phases $\theta(\tau)$ into a real motion. Later, the tracking controller will use $q_{ref}(\tau)$ to control the legged robots.

There are several disadvantages of CPGs as gait reference generator[4]:

1. The needs of solving differential equations continuously, in real-time
2. The equation 3-1 will exhibit transient behavior which adds complexity into the system

With the linearity and the nonexistence of the differential equations in SMPL system, later it will be shown that SMPL system can be proposed as an alternative to CPGs

3-2-2 Buehler Clock

Buehler clock is first used as a reference trajectory for legs of RHex robot[6]. The Buehler clock is modeled as continuous piecewise linear functions indicating the phase reference for each leg for a specific time. In figure $3-1$, Buehler clock model for tripod gait of hexapod

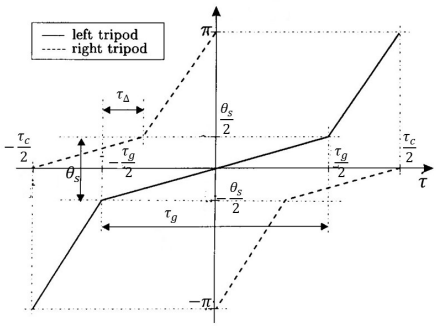


Figure 3-1: The Buehler Clock model of a hexapod robot[6]

robots is shown. The left tripod gait represents the phase reference for legs 1,4, and 5. The right tripod gait represents the phase reference for legs 2,3, 6. The numbering of the legs is referred to figure $3-2$. From figure $3-1$, the mathematical model of the Buehler clock can be written as

$$\theta_{\text{left tripod}}(\tau) = \begin{cases} \frac{\theta_s}{\tau_g}\bar{\tau} & \text{if} -\frac{\tau_g}{2} < \bar{\tau} < \frac{\tau_g}{2} \\ \frac{\pi-\theta_s}{\tau_c-\tau_g}(\bar{\tau} - \frac{\tau_g}{2}) + \frac{\theta_s}{2} & \text{if} \bar{\tau} \geq \frac{\tau_g}{2} \\ \frac{\pi-\theta_s}{\tau_c-\tau_g}(\bar{\tau} + \frac{\tau_g}{2}) - \frac{\theta_s}{2} & \text{if} \bar{\tau} \leq \frac{\tau_g}{2} \end{cases} \tag{3-3}$$

$$\theta_{\text{right tripod}}(\tau) = \theta_{\text{left tripod}}(\tau + \frac{\tau_c}{2}) \tag{3-4}$$

with τ_c as the cycle time, τ_g as the stance time, and θ_s as the stance phase. For $\bar{\tau}$, it can be defined as

$$\bar{\tau} = ((\tau + \frac{\tau_c}{2})\text{modulo}\tau_c) - \frac{\tau_c}{2}$$

The Buehler clock model represent a simple gait reference generator than CPGs. There is no differential equation involved in Buehler clock model, in contrast with CPGs. Later, it will be shown that SMPL model of legged locomotion system can be viewed as generalization of Buehler clock[4]

3-3 SMPL for Straightforward-Only Legged Locomotion

In this report, the considered legged robot is Zebro robot. With six half-circular compliant legs, the indexing of the legs is needed to identify the discussed legs. The indexing of the legs is indicated in the figure below

The indexing of the legs will be used in the discussion about the SMPL system and the gait.

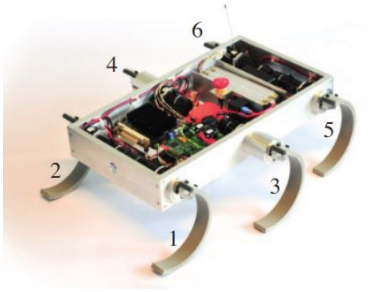


Figure 3-2: Zebro with the leg index numbering assumed in this paper [4, 6]

3-3-1 SMPL System

To model legged locomotion system with SMPL, the states of the model needs to be defined first. As a discrete event system (DES), the states of legged locomotion system can be defined as the time instant of events. The events are touchdown and lift-off of each leg. Then, let t_i be the time instant of touchdown event of leg i and l_i be the time instant of lift-off event of leg i. Therefore t_i and l_i can be defined as[14]

$$t_i(k) = l_i(k) + \tau_f \tag{3-5}$$
$$l_i(k) = t_i(k-1) + \tau_g \tag{3-6}$$

with k as the cycle counter of the events, τ_f as the minimal swing time. The equations $3-5$ until $3-6$ can be used to synchronize the legs movement in legged locomotion, combining with max-plus algebra. For example, to synchronize the legs of humanoid robot, this equation can be applied

$$\begin{aligned} t_{\text{left}}(k) &= l_{\text{left}}(k) + \tau_f \\ &= l_{\text{left}}(k) \otimes \tau_f \end{aligned} \tag{3-7}$$
$$\begin{aligned} t_{\text{right}}(k) &= l_{\text{right}}(k) + \tau_f \\ &= l_{\text{right}}(k) \otimes \tau_f \end{aligned} \tag{3-8}$$
$$\begin{aligned} l_{\text{left}}(k) &= \max(t_{\text{left}}(k-1) + \tau_g, t_{\text{right}}(k-1) + \tau_\Delta) \\ &= t_{\text{left}}(k-1) \otimes \tau_g \oplus t_{\text{right}}(k-1) \otimes \tau_\Delta \end{aligned} \tag{3-9}$$
$$\begin{aligned} l_{\text{right}}(k) &= \max(t_{\text{right}}(k-1) + \tau_g, t_{\text{left}}(k) + \tau_\Delta) \\ &= t_{\text{right}}(k-1) \otimes \tau_g \oplus t_{\text{left}}(k) \otimes \tau_\Delta \end{aligned} \tag{3-10}$$

with τ_Δ as the double stance time. With this synchronization, the left leg of the humanoid robot will lift off-first. Then, the right leg of the humanoid robot can only lift off τ_Δ seconds after the touchdown of the left leg. The equations $3-7$ until $3-10$ can be translated into

$$x(k) = \begin{bmatrix} \varepsilon & \varepsilon & \tau_f & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \tau_f \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \tau_\Delta & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \otimes x(k) \oplus \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \tau_g & \tau_\Delta & \varepsilon & \varepsilon \\ \tau_\Delta & \tau_g & \varepsilon & \varepsilon \end{bmatrix} \otimes x(k-1) \tag{3-11}$$

with $x(k) = [t_{\text{left}}(k)\ t_{\text{right}}(k)\ l_{\text{left}}(k)\ l_{\text{right}}(k)]^T$. For multi-legged robots, the matrices in equation $3-11$ will have various forms and size. The size of the matrices will depend on the number of legs and the form of the matrices will depend on the gait which is used by the legged robots. Therefore, there is a need to formalize the construction of the model as in equation $3-11$. For a n-legged robots, let l_1,...,l_m be the sets of integers which defined as[20]

$$\bigcup_{p=1}^{m} \ell_p = \{1, ..., n\} \text{for}$$
$$\forall i \neq j, \ell_i \cap \ell_j = \emptyset$$

Each l_p will contain a set of legs which recirculate simultaneously (lift-off together). Each l_p can be ordered to form a gait which can be defined as[20]

$$\mathcal{G} = \ell_1 \prec \ell_2 \prec ... \prec \ell_m \tag{3-12}$$

Therefore, the relation in $3-12$ can be interpreted as the order of recirculation of each group of legs. If the gait is already defined, then equation $3-11$ can be formalized as[21]

$$x(k) = \left[\begin{array}{c|c} \mathcal{E} & \tau_f \otimes E \\ \hline P & \mathcal{E} \end{array}\right] \otimes x(k) \oplus \left[\begin{array}{c|c} \mathcal{E} & \mathcal{E} \\ \hline \tau_g \otimes E \oplus Q & \mathcal{E} \end{array}\right] \otimes x(k-1) \tag{3-13}$$

with $x(k) = [t_1(k)...\ t_n(k)\ l_1(k)...\ l_n(k)]^T$. A cycle k is finished if l_m in $\mathcal{G}$ is already done. The chosen gait is defined in matrices P and Q. Those matrices will synchronize the legged locomotion as in equations $3-7$ until $3-10$, with regards to the chosen gait. Therefore, with this synchronization, the legs in l_2 can only lift-off τ_Δ seconds after the legs in l_1 touchdown. The chosen gait can be translated into matrices P and Q with[21]

$$[P]_{p,q} = \begin{cases} \tau_\Delta & \forall j \in \{1, ..., m-1\}; \forall p \in l_{j+1}; \forall q \in l_j \\ \varepsilon & \text{otherwise} \end{cases} \tag{3-14}$$

$$[Q]_{p,q} = \begin{cases} \tau_\Delta & \forall p \in l_1; \forall q \in l_m \\ \varepsilon & \text{otherwise} \end{cases} \tag{3-15}$$

The model in $3-13$ can be simplified into form

$$x(k) = A_0 \otimes x(k) \oplus A_1 \otimes x(k-1) \tag{3-16}$$

This model can be regarded as an implicit SMPL system. The model does not only depend on the previous cycle states but also on the current cycle states. With various kind of gait which can be applied, A_0 and A_1 can have various kind of modes with respect to the defined gait. The implicit SMPL system in equation $3-16$ then can be transformed into an explicit SMPL systems as

$$\begin{aligned} x(k) &= A_0 \otimes x(k) \oplus A_1 \otimes x(k-1) \\ &= A_0^* \otimes A_1 \otimes x(k-1) \\ &= A \otimes x(k-1) \end{aligned} \tag{3-17}$$

This transformation is only possible if Theorem 2−2.1[1] is fulfilled. However, for the existence of an explicit SMPL system of legged locomotion, there is an additional requirement[21].

Lemma 3-3.1. *A sufficient condition for A_0^* to exist is that the matrix P is nilpotent in the max-plus sense*

Lemma $3-3.1$ exists as diagonal elements of non-positive values are not desirable in matrix A_0. The existence of non-positive values in matrix A_0 implies that τ_f and τ_Δ are non-positive, which does not make sense. Therefore, to fulfill Theorem $3-3.1$, matrix P must be nilpotent. The proof of Lemma $3-3.1$ is included in the Appendix.

The matrix P itself can be proven as a nilpotent matrix. By the construction of the matrix P, $\mathcal{G}(P)$ is a strongly connected precedence graph. It can be observed that the column index which is contained in l_j will be used as the row index in l_{j+1}. By this way, for any two different nodes, there exists a path from a node to another node. Then, the matrix P is irreducible. The irreducibility of matrix P implies that it is a nilpotent matrix. Because of the irreducibility, matrix $A_0^\star$ can be calculated as

$$A_0^\star = A_0 \oplus A_0^{\otimes 2} \oplus ... \oplus A_0^{\otimes n-1} \tag{3-18}$$

3-3-2 Eigenstructure and Coupling Time of the System Matrices

From matrix A in equation $3-17$, several important max-plus-algebraic properties can be inferred. Those properties are max-plus-algebraic eigenvalue, max-plus-algebraic eigenvector, coupling time, and cyclicity. To derive those properties, two assumptions need to be considered

$$\begin{aligned} &\text{B1} : \tau_f, \tau_g > 0 \\ &\text{B2} : (\tau_f \otimes \tau_\Delta)^{\otimes m} \geq \tau_f \otimes \tau_g \end{aligned}$$

[20]The assumption B1 is straightforward from the fact that the swing and stance time of every legs will always be more than 0 if the legged robots move. With assumption B2, τ_f, τ_g, τ_Δ have limited set of possibilities. The derivation of max-plus-algebraic eigenvalue and eigenvector can be explained as

Lemma 3-3.2. *If assumption A1 hold then the unique max-plus-algebraic eigenvalue of matrix A in system $3-17$ is*

$$\lambda := (\tau_f \otimes \tau_\Delta)^{\otimes m} \oplus \tau_f \otimes \tau_g \tag{3-19}$$

and the max-plus-algebraic eigenvector $v \in \mathbb{R}_{max}^{2n\times 1}$ *is*

$$\forall j \in \{1, ..., m\}; \forall q \in l_j : [v]_q := \tau_f \otimes (\tau_f \otimes \tau_\Delta)^{\otimes j-1} \tag{3-20}$$

$$[v]_{q+n} := (\tau_f \otimes \tau_\Delta)^{\otimes j-1} \tag{3-21}$$

with n *as the number of the states.*

From Lemma $3-3.2$, it is clear that there is only a max-plus-algebraic eigenvalue with a corresponding max-plus-algebraic eigenvector. Therefore, matrix A in equation $3-17$ will always be an irreducible matrix[1]. From equation $3-19$ in Lemma $3-3.2$, max-plus-algebraic eigenvalue of the A matrix is the total cycle time of the gait. The corresponding max-plus-algebraic eigenvector in equation $3-20$ describes the steady-state behaviour[4, 1]. With knowledge of the max-plus-algebraic eigenvalue and eigenvector, the coupling time and cyclicity can be derived using equation $2-14$ in the previous chapter. It is possible as the matrix A in systems $3-17$ is irreducible. The resulting coupling time and cyclicity are

Lemma 3-3.3. *If assumptions A1 and A2 hold then the coupling time of matrix A in equation* $3-17$ *is* $k_0 = 2$ *with cyclicity* $c = 1$

The coupling time explains the transient behaviour of legged robots with system $3-17$. With coupling time of 2, the legged robots with system $3-17$ will achieve steady-state condition within 2 cycles after a gait transition or perturbation.

3-3-3 Gait Transitions

From Chapter 2, it is already explained that different gait will result in different matrices A_0 and A_1 of SMPL system $3-16$. However, the consequence of switching gait has not been explained yet. To understand the consequence of switching gait, the representation of a gait needs to be understood. With how a gait is defined in $3-12$, a certain gait can be represented in several ways. For example, a tripod gait can be represented as

$$\mathcal{G} = \{1, 4, 5\} \prec \{2, 3, 6\}$$
$$\mathcal{G} = \{2, 3, 6\} \prec \{1, 4, 5\}$$

and pentapod gait can be represented as

$$\mathcal{G} = \{1\} \prec \{2\} \prec \{3\} \prec \{4\} \prec \{5\} \prec \{6\}$$
$$\mathcal{G} = \{6\} \prec \{5\} \prec \{4\} \prec \{3\} \prec \{2\} \prec \{1\}$$
$$\mathcal{G} = \{4\} \prec \{3\} \prec \{2\} \prec \{1\} \prec \{6\} \prec \{5\}$$

Each representation of a particular gait will result in the same legged locomotion behavior. However, different representation will result in different gait transition characteristic if the gait is switched. For example, if the gait is switched from tripod gait to pentapod gait

$$\{1, 4, 5\} \prec \{2, 3, 6\} \prec \{1\} \prec \{2\} \prec \{3\} \prec \{4\} \prec \{5\} \prec \{6\}$$

will have different gait transition characteristic compared with

$$\{1, 4, 5\} \prec \{2, 3, 6\} \prec \{4\} \prec \{3\} \prec \{2\} \prec \{1\} \prec \{6\} \prec \{5\}$$

for constant τ_f, τ_g, and τ_Δ. When the gait is switched, the stance time of each leg will change. The amount of stance time change will depend on the chosen representation of the gaits involved. In the example above, the stance time of leg 4 is larger in the first kind of transition than the second kind.

To know the amount of change in stance time during gait transition, the properties of max-plus-algebraic eigenvalue can be utilized. If assumptions A1 and A2 are fulfilled, the equation $3-19$ becomes $\lambda{:=}(\tau_f \otimes \tau_\Delta)^{\otimes m}$. The assumption A2 is also considered here as coupling time needs to be minimized for the transition. Therefore, τ_f, τ_g, τ_Δ are set with regards to assumption A2. It is already known that the max-plus-algebraic eigenvalue of matrix A in system $3-17$ is the total cycle time of a gait, with l_m as the last set of legs. Therefore, the time instant of legs in set l_i will lift-off is[22]

$$l_{l_i} = \tau_f \otimes \tau_\Delta^{\otimes i} \tag{3-22}$$

for $0 < i < j \leq m$. If the legs in set l_i are moved into set l_j in the new gait, the time instant of the legs will lift-off is

$$l_{l_j} = \tau_f \otimes \tau_\Delta^{\otimes j} \tag{3-23}$$

Therefore, the amount of change in stance time is equal the difference between l_{l_i} and l_{l_j}[22]

$$\Delta_l = \tau_f \otimes \tau_\Delta^{\otimes j-i} \tag{3-24}$$

If this is the case, then the legs involved will stance for extra Δ_l seconds during the transition to synchronize. It might also be the case that $0 < j < i \leq m$. Then, if the legs in set l_i are moved into set l_j in the new gait, then

$$\Delta_l = \tau_f \otimes \tau_\Delta^{\otimes i-j} \tag{3-25}$$

In this case, the legs involved will postpone the lift-off for Δ_l seconds. To achieve optimal gait transition, the variance of the stance time during transition should be minimized. Therefore, optimal gait transition can be achieved by minimizing Δ_l. The minimization of Δ_l during gait transition can be realized by choosing a representation of a gait which resemble the previous gait[21]. This decision will cause the difference between l and j is minimal, so Δ_l is minimal.

3-4 SMPL for Trajectory-Following Legged Locomotion

In the section 3-3, the remaining SMPL system has already been discussed. It is already proven that the remaining SMPL system can guarantee synchronization of legs. However, the remaining SMPL system was only tested in Zebro for straight-line motion [4]. On the other hand, to design a trajectory-following legged locomotion system, the legged robot needs to turn. In our case, Zebro does not have any steering system. Then, without any added part, Zebro can only turn using a method called differential drive. With differential drive, the linear velocity of a side of the Zebro will be different to another side. The difference of the linear velocity causes the Zebro turns while moving forward. This method is already tested in RHex [6].

There are two possible ways to generate specific linear velocity for each side of the Zebro. The first way is by adding offset in the phases of the legs. For the same stance and swing

period, legs on the left side of Zebro will have different angular velocity than the right side. The linear velocity of a side of Zebro will be different to another side as the result. However, the possible offset added is bounded due to the shape of the legs. The second way is by determining different stance time for each side of the Zebro. This way offers less bound on the determined stance time, as long as the generated angular velocity of each leg does not violate the constraint. Therefore, it is more suitable for control purpose. To apply different stance time for each side of the Zebro, a new SMPL system is proposed in this section.

3-4-1 SMPL Model

To induce different stance time for each side, a parameter τ_p is introduced. The parameter τ_p is defined as difference time between left and right legs. In a turning process, τ_p is added to the stance time of the inner legs. Consequently, the inner legs have longer stance time and slower angular velocity. The linear velocity of the inner side will be slower than the outer side. For the outer legs, τ_p is added to the swing time. The addition of τ_p to the swing time of the outer legs guarantees that the total cycle time for each leg is same. With this factor, the calculation of the states are cyclic. Apart from τ_p, the other parameters are the same as the remaining SMPL system. There is an important convention which is desired by using the trajectory-following SMPL system. The convention is that the legs in the same sequence will lift-off together on every condition.

With the new parameter, t_i and l_i of the legs can be classified into two types when the legged robot is turning. They are t_i^i and l_i^i of the inner legs, and t_i^o and l_i^o of the outer legs. For independent legs, equation 3-5 and 3-6 can be derived further into

$$t_i^i(k) = l_i^i(k) + \tau_f \tag{3-26}$$
$$t_i^o(k) = l_i^o(k) + \tau_f + \tau_p \tag{3-27}$$
$$l_i^i(k) = t_i^i(k-1) + \tau_g + \tau_p \tag{3-28}$$
$$l_i^o(k) = t_i^o(k-1) + \tau_g \tag{3-29}$$

To compare the trajectory-following SMPL system with the straightforward-only SMPL system, the humanoid robot example is repeated for turning motion. For a turning left humanoid robot, the left leg is the inner leg and the right leg is the outer leg. The states of both legs are then defined as

$$\begin{aligned} t_{\text{left}}(k) &= l_{\text{left}}(k) + \tau_f \\ &= l_{\text{left}}(k) \otimes \tau_f \end{aligned} \tag{3-30}$$
$$\begin{aligned} t_{\text{right}}(k) &= l_{\text{right}}(k) + \tau_f + \tau_p \\ &= l_{\text{right}}(k) \otimes \tau_f \otimes \tau_p \end{aligned} \tag{3-31}$$
$$\begin{aligned} l_{\text{left}}(k) &= \max(t_{\text{left}}(k-1) + \tau_g + \tau_p, t_{\text{right}}(k-1) + \tau_\Delta) \\ &= t_{\text{left}}(k-1) \otimes \tau_g \otimes \tau_p \oplus t_{\text{right}}(k-1) \otimes \tau_\Delta \end{aligned} \tag{3-32}$$
$$\begin{aligned} l_{\text{right}}(k) &= \max(t_{\text{right}}(k-1) + \tau_g, t_{\text{left}}(k) + \tau_p) \\ &= t_{\text{right}}(k-1) \otimes \tau_g \oplus t_{\text{left}}(k) \otimes \tau_p \end{aligned} \tag{3-33}$$

From equation 3-32 and 3-33, terms $t_{\text{right}}(k-1) \otimes \tau_\Delta$ and $t_{\text{left}}(k) \otimes \tau_p$ can be observed. These terms appear because the swing time of the left leg is shorter than the right leg. Therefore, after the left leg touchdowns, there are τ_p seconds difference with the lift-off of the right leg. For multi-legged robot, these terms appear to guarantee that all legs in l_{j+1} lift-off after all legs in l_j have already touchdown. The outer legs in l_j will touchdown τ_p seconds after the inner legs.

Observing the equation 3-32 to 3-33, it is obvious that the inner legs and the outer legs need to denoted with different notation. With different notation, trajectory-following SMPL system can be described easier mathematically. The gait parameterization can be redefined as

$$\begin{aligned} \bigcup_{p=1}^{m} \ell_p = \{1, ..., n\} \quad & \text{for} \\ \forall i \neq j, \ell_i \cap \ell_j = \emptyset & \\ \text{iff} \quad \bar{\tau}_g^{2k-1} \neq \bar{\tau}_g^{2k}, k \in 1, ..., \frac{n}{2} & \\ \ell_p^i \subset \ell_p \quad & \text{for inner legs} \\ \ell_p^o \subset \ell_p \quad & \text{for outer legs} \\ \ell_p^i \cup \ell_p^j = \ell_p, \ell_p^i \cap \ell_p^j = \emptyset & \\ \text{iff} \quad \bar{\tau}_g^{2k-1} = \bar{\tau}_g^{2k}, k \in 1, ..., \frac{n}{2} & \\ \ell_p^i = \emptyset, \ell_p^o = \emptyset & \end{aligned}$$

In the gait parameterization, if turning motion exists, the inner and outer legs are defined as subsets of l_p. The set of inner legs are denoted by l_p^i and outer legs by l_p^o. These sets only exist if the linear velocity of the left side and right side of Zebro are different. The elements of sets l_p^i and l_p^o also depend on the direction of turn. The difference is defined by the inequality of true stance time of even indexed legs $\bar{\tau}_g^{2k}$ and odd indexed legs $\bar{\tau}_g^{2k-1}$. Here, the true stance time of leg i is defined as $\bar{\tau}_g^i$. If the existing motion is straightforward, the sets l_p^i and l_p^o are empty and only set l_p exists.

The trajectory-following SMPL itself does not only consider turning motion. Straight-forward motion is also still considered. Therefore, trajectory-following SMPL can be seen as a general form of SMPL system for legged locomotion. For a n-legged trajectory-following robot, the SMPL system in 3-13 can be modified into

$$\begin{aligned} x(k) &= \left[\begin{array}{c|c} \mathcal{E} & \tau_f \otimes M \\ \hline R & \mathcal{E} \end{array}\right] \otimes x(k) \oplus \left[\begin{array}{c|c} E & \mathcal{E} \\ \hline (\tau_g \otimes N) \oplus T & E \end{array}\right] \otimes x(k-1) \\ &= G_0 \otimes x(k) \oplus G_1 \otimes x(k-1) \end{aligned} \tag{3-34}$$

In this SMPL system, matrices R, T, M, and N are introduced. The introduced matrices replace P, Q, and two max-plus identity matrices respectively. The matrices R and T are constructed in a similar way as matrices P and Q. To construct matrices R and T, these rules are used

$$[R]_{p,q} = \begin{cases} \tau_\Delta & \forall j \in \{1,...,m-1\}; \forall p \in \ell^o_{j+1}; \forall q \in \ell^o_j; \bar{\tau}^{2k-1}_g \neq \bar{\tau}^{2k}_g, k \in 1,...,\frac{n}{2} \\ \tau_p & \forall j \in \{1,...,m-1\}; \forall p \in \ell^i_{j+1}; \forall q \in \ell^i_j; \bar{\tau}^{2k-1}_g \neq \bar{\tau}^{2k}_g, k \in 1,...,\frac{n}{2} \\ \tau_\Delta & \forall j \in \{1,...,m-1\}; \forall p \in \ell_{j+1}; \forall q \in \ell_j; \bar{\tau}^{2k-1}_g = \bar{\tau}^{2k}_g, k \in 1,...,\frac{n}{2} \\ \varepsilon & \text{otherwise} \end{cases} \tag{3-35}$$

$$[T]_{p,q} = \begin{cases} \tau_\Delta & \forall p \in \ell^o_1; \forall q \in \ell^o_m; \bar{\tau}^{2k-1}_g \neq \bar{\tau}^{2k}_g, k \in 1,...,\frac{n}{2} \\ \tau_p & \forall p \in \ell^i_1; \forall q \in \ell^i_m; \bar{\tau}^{2k-1}_g \neq \bar{\tau}^{2k}_g, k \in 1,...,\frac{n}{2} \\ \tau_\Delta & \forall p \in \ell_1; \forall q \in \ell_m; \bar{\tau}^{2k-1}_g = \bar{\tau}^{2k}_g, k \in 1,...,\frac{n}{2} \\ \varepsilon & \text{otherwise} \end{cases} \tag{3-36}$$

All conditions in the rules above correspond to the gait parameterization in this subsection. The structure of matrices R and T depend on the chosen gait and the generated motion. Every combination of a particular gait and a particular direction will yield a particular structure of matrices R and T. Turning left, turning right, and straightforward motion will result in a different structure of matrices R and T respectively. Different from matrices R and T, matrices M and N do not depend on the chosen gait. The matrices M and N are constructed by these rules

$$[M]_{p,q} = \begin{cases} \tau_p & \forall j \in \{1,...,m\}; \forall p \in \ell^o_j; \forall p = q; \bar{\tau}^{2k-1}_g \neq \bar{\tau}^{2k}_g, k \in 1,...,\frac{n}{2} \\ e & \forall j \in \{1,...,m\}; \forall p \in \ell^i_j; \forall p = q; \bar{\tau}^{2k-1}_g \neq \bar{\tau}^{2k}_g, k \in 1,...,\frac{n}{2} \\ e & \forall p = q; \bar{\tau}^{2k-1}_g = \bar{\tau}^{2k}_g, k \in 1,...,\frac{n}{2} \\ \varepsilon & \text{otherwise} \end{cases} \tag{3-37}$$

$$[N]_{p,q} = \begin{cases} e & \forall j \in \{1,...,m\}; \forall p \in \ell^o_j; \forall p = q; \bar{\tau}^{2k-1}_g \neq \bar{\tau}^{2k}_g, k \in 1,...,\frac{n}{2} \\ \tau_p & \forall j \in \{1,...,m\}; \forall p \in \ell^i_j; \forall p = q; \bar{\tau}^{2k-1}_g \neq \bar{\tau}^{2k}_g, k \in 1,...,\frac{n}{2} \\ e & \forall p = q; \bar{\tau}^{2k-1}_g = \bar{\tau}^{2k}_g, k \in 1,...,\frac{n}{2} \\ \varepsilon & \text{otherwise} \end{cases} \tag{3-38}$$

From rule 3-37 and 3-38, the structure of matrices M and N depend only on the direction of motion. As in the straightforward-only SMPL system, matrices M and N are max-plus identity matrices E for straightforward motion. For turning motion, matrices M and N are diagonal matrices. The diagonal element of matrices M and N depend on the direction of turn. For example, if the Zebro turns left, then the even indexed diagonal elements in matrix M are τ_p. The odd indexed diagonal elements in matrix M are e. The opposite applies on matrix N.

The SMPL system in 3-34 is still described in implicit form. To simplify the calculation of the states, the SMPL system 3-34 needs to be transformed into an explicit form. In lemma 3-3.1, the sufficient condition for the transformation is already stated for the straightforward-only SMPL system. To check the validity of Lemma 3-3.1 for trajectory-following SMPL system, repetitive product of G_0 is calculated

$$G_0^{\otimes p} = \begin{cases} \left[\begin{array}{c|c} \mathcal{E} & \tau_f^{\otimes k} \otimes M^{\otimes k} \otimes R^{\otimes (k-1)} \\ \hline R^{\otimes k} \otimes \tau_f^{\otimes (k-1)} \otimes M^{\otimes (k-1)} & \mathcal{E} \end{array}\right] & \text{if k is odd} \\ \left[\begin{array}{c|c} \tau_f^{\otimes k} \otimes M^{\otimes k} \otimes R^{\otimes k} & \mathcal{E} \\ \hline \mathcal{E} & R^{\otimes k} \otimes \tau_f^{\otimes k} \otimes M^{\otimes k} \end{array}\right] & \text{if k is even} \end{cases} \tag{3-39}$$

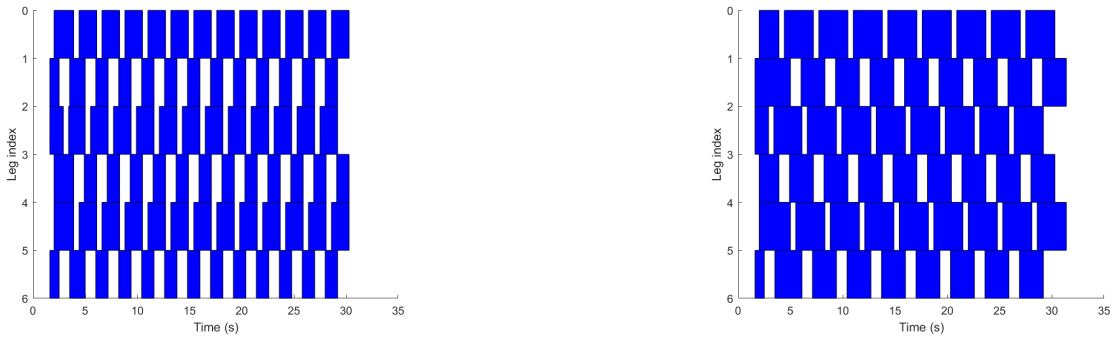
Here, the Kleene-product of G_0, G_0^*, exists only if either matrix R or M is nilpotent. As the matrix M consist of diagonal finite elements, matrix M will not be nilpotent in any case. Therefore, corresponding to Lemma 3-3.1 , matrix G_0^* exists if matrix R is nilpotent. Lemma 3-3.1 holds, with matrix R replacing matrix P. The matrix G_0^* can be calculated with

$$G_0^* = \bigoplus_{p=0}^{n-1} G_0^{\otimes p} \tag{3-40}$$

The implicit trajectory-following SMPL system 3-34 then can be transformed into

$$\begin{aligned} x(k) &= G_0 \otimes x(k) \oplus G_1 \otimes x(k-1) \\ &= G_0^\star \otimes G_1 \otimes x(k-1) \\ &= G \otimes x(k-1) \end{aligned} \tag{3-41}$$

The trajectory-following SMPL system will result in the touchdown and lift-off time instant of every leg as the states. The states of the trajectory-following SMPL can be illustrated by a gantt chart. Here, the gantt chart of the states are presented for tripod $\{1,4,5\},\{2,3,6\}$ and quadruped $\{1,4\},\{3,6\},\{2,5\}$ as the gait. The chosen parameters are τ_f, τ_p, τ_g, and τ_Δ equals to 0.5, 0.5, 0.8, and 0.1.



(a) Tripod **(b)** Quadruped

Figure 3-3: The gantt chart of the states of the trajectory-following SMPL system with tripod and quadruped as the gait. The blue bar denotes the stance period, and the white bar denotes the swing period

3-4-2 Eigenstructure of System Matrices

With the addition of parameter τ_p, the eigenvalue and eigenvector of the trajectory-following SMPL system are different compared to straightforward-only SMPL system. As the addition of τ_p changes how touchdown and lift-off time instant are calculated, τ_p affects the value of eigenvalue and eigenvector. The eigenvalue and eigenvector of the proposed SMPL system are defined with the following lemma :

Lemma 3-4.1. *Consider two assumptions :*

$$A1 : \tau_f > 0, \quad \tau_g > 0$$
$$A2 : (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes m} > (\tau_g \otimes \tau_f \otimes \tau_p)$$

Assuming that assumption A1 holds, and for turning motion, the max-plus-algebraic eigenvalue of matrix G is uniquely defined as

$$\lambda := (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes m} \tag{3-42}$$

and the max-plus-algebraic eigenvector $v \in \mathbb{R}_{max}^{2n\times 1}$ is defined as

$$\forall j \in \{1, ..., m\}; \forall q^i \in l_j^i; \forall q^o \in l_j^o : \tag{3-43}$$
$$[v]_q^i := \tau_f \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \tag{3-44}$$
$$[v]_q^o := (\tau_f \otimes \tau_p) \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \tag{3-45}$$
$$[v]_{q+n}^{i,o} := (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \tag{3-46}$$

In the previous work [20], it is proven that the eigenvalue represents the total cycle time and the eigenvector represents the time instant of touchdown and lift-off of the legs in steady state condition. The variable states $x(k)$ can be substituted by the $\lambda \otimes v$, as the current time instant of touchdown and lift-off of the legs is equal to the summation of previous time instant and total cycle time. Therefore, variable states $x(k-1)$ can also be substituted by v. As the cycle time, the eigenvalue is always larger than zero. With the substitution, the SMPL system in 3-34 is transformed into :

$$\lambda \otimes v = \lambda \otimes \left[\begin{array}{c|c} \mathcal{E} & \tau_f \otimes M \\ \hline R & \mathcal{E} \end{array}\right] \otimes v \oplus \left[\begin{array}{c|c} E & \mathcal{E} \\ \hline (\tau_g \otimes N) \oplus T & E \end{array}\right] \otimes v \tag{3-47}$$
$$\lambda \otimes v = (\lambda \otimes \left[\begin{array}{c|c} \mathcal{E} & \tau_f \otimes M \\ \hline R & \mathcal{E} \end{array}\right] \oplus \left[\begin{array}{c|c} E & \mathcal{E} \\ \hline (\tau_g \otimes N) \oplus T & E \end{array}\right]) \otimes v \tag{3-48}$$
$$\lambda \otimes \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \left[\begin{array}{c|c} E & (\lambda \otimes \tau_f) \otimes M \\ \hline \lambda \otimes R \oplus (\tau_g \otimes N) \oplus T & E \end{array}\right] \otimes \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{3-49}$$

The validity of equation 3-49 then is used to proof that the lemma 3-3.2 holds true. Equation 3-49 can be divided into two equations :

$$\lambda \otimes v_1 = v_1 \oplus \lambda \otimes \tau_f \otimes M \otimes v_2 \tag{3-50}$$
$$\lambda \otimes v_2 = (\lambda \otimes R \oplus \tau_g \otimes N \oplus T) \otimes v_1 \tag{3-51}$$

Equation 3-50 and equation 3-51 are the first and second row of equation 3-49 respectively. Here, v_1 consist of $[v]_q^i$ and $[v]_q^o$ and v_2 consists of $[v]_{q+n}^{i,o}$. The term v_2 can be removed from the right side of equation 3-51 as it will always be less than $(\lambda \otimes R \oplus \tau_g \otimes N \oplus T) \otimes v1$. Equation 3-50 then can be derived for two cases. For the first case, the eigenvector of inner legs are considered. In this case, equation 3-50 is derived as :

$$\lambda \otimes v_1 = v_1 \oplus \lambda \otimes \tau_f \otimes e \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \tag{3-52}$$

$$\lambda \otimes \tau_f \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} = \tau_f \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \oplus \lambda \otimes \tau_f \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \tag{3-53}$$

$$\lambda \otimes \tau_f \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} = \lambda \otimes \tau_f \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \tag{3-54}$$

From equation 3-52, it can be noted that $[M]_{p,q}$ is equal to e for inner legs. The equations 3-53 and 3-54 hold true as the eigenvalue is always larger than zero. In the second case, equation 3-50 is derived with the eigenvector of outer legs, as :

$$\lambda \otimes v_1 = v_1 \oplus \lambda \otimes \tau_f \otimes \tau_p \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \tag{3-55}$$

$$\lambda \otimes v_1 = (\tau_f \otimes \tau_p) \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \oplus \lambda \otimes \tau_f \otimes \tau_p \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \tag{3-56}$$

$$\lambda \otimes (\tau_f \otimes \tau_p) \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} = \lambda \otimes \tau_f \otimes \tau_p \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \tag{3-57}$$

Different than equation 3-52, $[M]_{p,q}$ is equal to τ_p in equation 3-55. Again, the equation 3-56 and 3-57 hold true as the eigenvalue is always larger than zero. Finally, equation 3-50 holds true for all possible cases. Now, equation 3-51 is also derived for two cases. In the first case, the row indices of equation 3-51 which correspond to $\forall j \in \{1, ..., m-1\}$, $\forall p \in \ell_{j+1}$ are considered. The derivation can be described as :

$$\begin{aligned}
\lambda \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j} &= (\lambda \otimes R \oplus \tau_g \otimes N \oplus T) \otimes v_1 \\
&= [\lambda \otimes R \otimes v_1]_p \oplus [\tau_g \otimes N \otimes v_1]_p \oplus [T \otimes v_1]_p \\
&= (\bigoplus_{q \in \ell_j} \lambda \otimes [R]_{p,q} \otimes [v_1]_q) \oplus (\bigoplus_{q \in \ell_{j+1}} \tau_g \otimes [N]_{p,q} \otimes [v_1]_q) \oplus ([T]_p \otimes v_1)
\end{aligned} \tag{3-58}$$

The derivation of equation 3-58 can be divided into three parts :

$$\begin{aligned}
\bigoplus_{q \in \ell_j} \lambda \otimes [R]_{p,q} \otimes [v_1]_q &= (\lambda \otimes \tau_\Delta \otimes \tau_f \otimes \tau_p \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1}) \\
&\quad \oplus (\lambda \otimes \tau_p \otimes \tau_f \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1}) \\
&= \lambda \otimes \tau_\Delta \otimes \tau_f \otimes \tau_p \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \\
&= \lambda \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j}
\end{aligned} \tag{3-59}$$

$$\begin{aligned}
\bigoplus_{q \in \ell_{j+1}} \tau_g \otimes [N]_{p,q} \otimes [v_1]_q &= (\tau_g \otimes e \otimes \tau_f \otimes \tau_p \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j}) \\
&= \tau_g \otimes \tau_f \otimes \tau_p \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j}
\end{aligned} \tag{3-60}$$

$$[T]_p \otimes v_1 = \varepsilon \tag{3-61}$$

It is important to note that in equation 3-59 max-plus summation is done for $q \in \ell_j$, corresponding to how matrix R is formed in rule 3-35. It is different than equation 3-60, where max-plus summation is done for $q \in \ell_{j+1}$ as $q = p$ in matrix N. Equation 3-61 is equal to ε as all elements of $[T]_p$ are ε for $\forall p \in \ell_{j+1}$. Combining equation 3-58, 3-59, 3-60, and 3-61, equation 3-58 can be further derived as :

$$\begin{aligned} \lambda \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j} &= \lambda \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j} \oplus \tau_g \otimes \tau_f \otimes \tau_p \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j} \\ &= \lambda \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j} \end{aligned} \tag{3-62}$$

In equation 3-62, λ is always larger than $\tau_g \otimes \tau_f \otimes \tau_p$ corresponding to assumption A2. Next, the remaining rows are now considered for the derivation of equation 3-51, as the second case. The remaining rows consist of rows $p \in \ell_1$, corresponding to how matrix T is formed. The derivation can be described as :

$$\lambda \otimes [v_2]_p = (\lambda \otimes R \oplus \tau_g \otimes N \oplus T) \otimes v_1 \tag{3-63}$$

$$\lambda \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes 1-1} = \lambda \otimes [R]_p \otimes v_1 \oplus [\tau_g \otimes N \otimes v_1]_p \oplus [T \otimes v_1]_p \tag{3-64}$$

$$\lambda = \lambda \otimes [R]_p \otimes v_1 \oplus (\bigoplus_{q \in l_1} \tau_g \otimes [N]_{p,q} \otimes [v_1]_q) \oplus (\bigoplus_{q \in l_m} [T]_{p,q} \otimes [v_1]_q) \tag{3-65}$$

The equation 3-65 then can be divided into three parts :

$$\lambda \otimes [R]_p \otimes v_1 = \varepsilon \tag{3-66}$$

$$\begin{aligned} \bigoplus_{q \in \ell_1} \tau_g \otimes [N]_{p,q} \otimes [v_1]_q &= (\tau_g \otimes e \otimes \tau_f \otimes \tau_p \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes 1-1}) \\ &= \tau_g \otimes \tau_f \otimes \tau_p \end{aligned} \tag{3-67}$$

$$\begin{aligned} \bigoplus_{q \in \ell_m} [T]_{p,q} \otimes [v_1]_q &= (\tau_\Delta \otimes \tau_f \otimes \tau_p \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes m-1}) \\ &\oplus (\tau_p \otimes \tau_f \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes m-1}) \\ &= \tau_\Delta \otimes \tau_f \otimes \tau_p \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes m-1} \\ &= (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes m} \end{aligned} \tag{3-68}$$

In the equation 3-66, all elements of $[R]_p$ are equal to ε for rows $p \in \ell_1$. Similar to previous case, max-plus summation is done for $q \in \ell_1$ as $q = p$ in matrix N. Whereas, max-plus summation is done for $q \in \ell_m$ in matrix T parallel to how matrix T is formed. Substituting equation 3-66, 3-67, and 3-68 to equation 3-65, the eigenvalue is obtained as

$$\lambda = (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes m}$$

Therefore, the lemma 3-4.1 is proven with all the equations in this subsection. It is also proven that the eigenvalue and eigenvector still represent the total cycle time and the steady-state condition as in the straightforward motion. In the steady-state condition, the desired

difference between each time instants is described. Knowing the desired difference of time instants, the chosen gait can be known. Furthermore, in turning motion, the direction of turning can also be inferred from the eigenvector. The direction of turn can be indicated by observing the elements of $[v]_q^i$ and $[v]_q^o$. If the elements of $[v]_q^i$ are odd-indexed, then the Zebro is turning left. Otherwise, the Zebro is turning right. The chosen gait can be concluded from the number of sets containing the same element in $[v]_q$, and the index of the elements in that set. The elements of $[v]_{q+n}^{i,o}$ which correspond to the legs in a particular ℓ_p will be the same. It implies that all legs in a particular ℓ_p will lift-off at the same time instant. From the definition of the eigenvalue, it is also can be concluded that the value of $\bar{\tau}_g$ depends on τ_p, τ_f, and τ_Δ instead of the τ_g itself. The τ_g will only act as the lower boundary in choosing the other parameters. The inner and outer true stance period, $\bar{\tau}_g^i$ and $\bar{\tau}_g^o$, then can be defined as

$$\bar{\tau}_g^i = \lambda - \tau_f \tag{3-69}$$
$$\bar{\tau}_g^o = \lambda - (\tau_f + \tau_p) \tag{3-70}$$

However, it is also possible that the legs in a particular ℓ_p do not lift-off at the same time instant. This phenomenon happens when the assumption A2 in Lemma 3-4.1 is violated. Then, the eigenvalue and the eigenvector of trajectory-following SMPL systems are described in a different way than in Lemma 3-4.1.

Lemma 3-4.2. *Consider the assumptions :*

$$A3 : \tau_p > \tau_\Delta$$

If the assumption A2 is violated and assumption A3 holds, and for turning motion, the max-plus-algebraic eigenvalue of matrix G is defined as

$$\lambda := \tau_g \otimes \tau_f \otimes \tau_p \tag{3-71}$$

and the max-plus-algebraic eigenvector $v \in \mathbb{R}_{max}^{2n\times 1}$ is defined as

$$\forall j \in \{2, ..., m\}; \forall q^i \in \ell_j^i; \forall q^o \in \ell_j^o : \tag{3-72}$$
$$[v]_q^i := \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{j-1} \otimes \tau_f \tag{3-73}$$
$$[v]_q^o := \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{j-1} \otimes \tau_f \otimes \tau_p \tag{3-74}$$
$$[v]_{q+n}^{i,o} := \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{j-1} \tag{3-75}$$
$$\forall q^i \in \ell_1^i; \forall q^o \in \ell_1^o : \tag{3-76}$$
$$[v]_q^{i,o} := \tau_g \otimes \tau_f \otimes \tau_p \tag{3-77}$$
$$[v]_{q+n}^i := \tau_g \otimes \tau_p \tag{3-78}$$
$$[v]_{q+n}^o := \tau_g \tag{3-79}$$

In proving the Lemma 3-4.2, the equation 3-50 and 3-51 are again derived. In contrast with equation 3-52 to 3-68, the derivation of equation 3-50 and 3-51 are done with respect to the eigenvalue and eigenvector in Lemma 3-4.2. Similar with the proof of Lemma 3-4.1, the proof is derived for several cases. To simplify the writing, $[v]_q$ and $[v]_{q+n}$ are also written as v_1 and v_2 instead. First, the equation 3-50 is derived for $\forall p \in \ell_1^i$

$$\lambda \otimes v_1 = v_1 \oplus \lambda \otimes \tau_f \otimes e \otimes \tau_g \otimes \tau_p \tag{3-80}$$
$$\lambda \otimes \tau_g \otimes \tau_p \otimes \tau_f = \lambda \otimes \tau_f \otimes \tau_g \otimes \tau_p \tag{3-81}$$

The equation 3-81 holds true as λ will always be larger than v_1. Then, the equation 3-50 is derived for $\forall p \in \ell_1^o$

$$\lambda \otimes v_1 = v_1 \oplus \lambda \otimes \tau_f \otimes \tau_p \otimes \tau_g \tag{3-82}$$
$$\lambda \otimes \tau_g \otimes \tau_p \otimes \tau_f = \lambda \otimes \tau_f \otimes \tau_p \otimes \tau_g \tag{3-83}$$

With both equation 3-81 and 3-83, equation 3-50 is proven to be true for $\forall p \in \ell_1$. Next, equation 3-50 is derived for $\forall p \in \ell_j^i$ with $\forall j \in 2, ..., m$.

$$\lambda \otimes v_1 = v_1 \oplus \lambda \otimes \tau_f \otimes e \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{j-1} \tag{3-84}$$
$$\lambda \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{j-1} \otimes \tau_f = \lambda \otimes \tau_f \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{j-1} \tag{3-85}$$

The last case for the derivation of equation 3-50 is for $\forall p \in \ell_j^o$ with $\forall j \in 2, ..., m$

$$\lambda \otimes v_1 = v_1 \oplus \lambda \otimes \tau_f \otimes \tau_p \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{j-1} \tag{3-86}$$
$$\lambda \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{j-1} \otimes \tau_f \otimes \tau_p = \lambda \otimes \tau_f \otimes \tau_p \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{j-1} \tag{3-87}$$

Observing both equation 3-85 and 3-87, equation 3-50 holds true for $\forall p \in \ell_j$ with $\forall j \in 2, ..., m$. The equation 3-50 then is proven for all cases. Thereafter, equation 3-51 needs to be proven so Lemma 3-4.2 can be verified. There are four cases considered in proving equation 3-51. For the first and second case, equation 3-51 is derived into equation 3-88 first. Equation 3-88 is similar with 3-58, except for how v_2 is defined on the left side

$$\begin{aligned}\lambda \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{j-1} &= (\lambda \otimes R \oplus \tau_g \otimes N \oplus T) \otimes v_1 \\ &= [\lambda \otimes R \otimes v_1]_p \oplus [\tau_g \otimes N \otimes v_1]_p \oplus [T \otimes v_1]_p \\ &= (\bigoplus_{q \in \ell_j} \lambda \otimes [R]_{p,q} \otimes [v_1]_q) \oplus (\bigoplus_{q \in \ell_{j+1}} \tau_g \otimes [N]_{p,q} \otimes [v_1]_q) \oplus ([T]_p \otimes v_1)\end{aligned} \tag{3-88}$$

In the first case, rows $\forall p \in \ell_2$ are considered. Equation 3-88 is then derived into three parts

$$\begin{aligned}\bigoplus_{q\in\ell_1}\lambda\otimes[R]_{p,q}\otimes[v_1]_q &= \lambda\otimes\tau_\Delta\otimes\tau_g\otimes\tau_p\otimes\tau_f\\ &\oplus\lambda\otimes\tau_p\otimes\tau_g\otimes\tau_p\otimes\tau_f\\ &=\lambda\otimes\tau_p\otimes\tau_g\otimes\tau_p\otimes\tau_f\end{aligned}\tag{3-89}$$

$$\begin{aligned}\bigoplus_{q\in\ell_2}\tau_g\otimes[N]_{p,q}\otimes[v_1]_q &= \tau_g\otimes e\otimes\tau_g\otimes(\tau_p^{\otimes 2}\otimes\tau_f)^{2-1}\otimes\tau_f\otimes\tau_p\\ &=(\tau_g\otimes\tau_p\otimes\tau_f)^2\otimes\tau_p\end{aligned}\tag{3-90}$$

$$[T]_p\otimes v_1=\varepsilon\tag{3-91}$$

In the equation 3-89, the assumption A3 is applied to get the maximal value. The result of equation 3-89 is the same as the result of equation 3-90 as the consequence of how the eigenvalue is defined. The equation 3-88 is then derived as

$$\lambda\otimes\tau_g\otimes(\tau_p^{\otimes 2}\otimes\tau_f)=\lambda\otimes\tau_p\otimes\tau_g\otimes\tau_p\otimes\tau_f\tag{3-92}$$

The result of equation 3-92 shows that the equation 3-51 holds true for $\forall p\in\ell_2$. Again, the equation 3-88 is derived into three parts for $\forall p\in\ell_j$ with $\forall j\in 3,...,m$

$$\begin{aligned}\bigoplus_{q\in\ell_{j-1}}\lambda\otimes[R]_{p,q}\otimes[v_1]_q &= \lambda\otimes\tau_\Delta\otimes\tau_g\otimes(\tau_p^{\otimes 2}\otimes\tau_f)^{j-2}\otimes\tau_f\otimes\tau_p\\ &\oplus\lambda\otimes\tau_p\otimes\tau_g\otimes(\tau_p^{\otimes 2}\otimes\tau_f)^{j-2}\otimes\tau_f\\ &=\lambda\otimes\tau_p\otimes\tau_g\otimes(\tau_p^{\otimes 2}\otimes\tau_f)^{j-2}\otimes\tau_f\end{aligned}\tag{3-93}$$

$$\bigoplus_{q\in\ell_j}\tau_g\otimes[N]_{p,q}\otimes[v_1]_q=\tau_g\otimes e\otimes\tau_g\otimes(\tau_p^{\otimes 2}\otimes\tau_f)^{j-1}\otimes\tau_f\otimes\tau_p\tag{3-94}$$

$$[T]_p\otimes v1=\varepsilon\tag{3-95}$$

Using assumption A3, the result of equation 3-93 is obtained. For this case, the result of equation 3-88 is

$$\lambda\otimes\tau_g\otimes(\tau_p^{\otimes 2}\otimes\tau_f)^{j-1}=\lambda\otimes\tau_p\otimes\tau_g\otimes(\tau_p^{\otimes 2}\otimes\tau_f)^{j-2}\otimes\tau_f\tag{3-96}$$

The equation 3-51 is again proven with the result of equation 3-96. The equation 3-51 now needs to be proven for the last two cases. The last two cases are for $\forall p\in\ell_1$. For $\forall p\in\ell_1^i$, equation 3-51 is derived into

$$\begin{aligned}\lambda\otimes\tau_g\otimes\tau_p &= (\lambda\otimes R\oplus\tau_g\otimes N\oplus T)\otimes v_1\\ &=[\lambda\otimes R\otimes v_1]_p\oplus[\tau_g\otimes N\otimes v_1]_p\oplus[T\otimes v_1]_p\\ &=(\bigoplus_{q\in\ell_j}\lambda\otimes[R]_{p,q}\otimes[v_1]_q)\oplus(\bigoplus_{q\in\ell_{j+1}}\tau_g\otimes[N]_{p,q}\otimes[v_1]_q)\oplus([T]_p\otimes v_1)\end{aligned}\tag{3-97}$$

The result of equation 3-97 is then derived again. The derivation is divided into three parts, as explained below

$$\lambda \otimes [R]_p \otimes v_1 = \varepsilon \tag{3-98}$$

$$\bigoplus_{q \in \ell_1^i} \tau_g \otimes [N]_{p,q} \otimes [v_1]_q = \tau_g \otimes \tau_p \otimes \tau_g \otimes \tau_f \otimes \tau_p \tag{3-99}$$

$$\begin{aligned} \bigoplus_{q \in \ell_m} [T]_{p,q} \otimes [v_1]_q &= \tau_\Delta \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{m-1} \otimes \tau_f \otimes \tau_p \\ &\oplus \tau_p \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{m-1} \otimes \tau_f \\ &= \tau_\Delta \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{m-1} \otimes \tau_f \otimes \tau_p \end{aligned} \tag{3-100}$$

Because of the violation of assumption A2, the result of equation 3-99 is always larger than the result of equation 3-100. Therefore, the result of equation 3-97 is obtained as

$$\lambda \otimes \tau_g \otimes \tau_p = \tau_g \otimes \tau_p \otimes \tau_g \otimes \tau_f \otimes \tau_p \tag{3-101}$$

With the similar steps as in equation 3-97 to 3-100, for $\forall p \in \ell_1^o$, equation 3-51 is derived into

$$\begin{aligned} \lambda \otimes \tau_g &= (\lambda \otimes R \oplus \tau_g \otimes N \oplus T) \otimes v_1 \\ &= [\lambda \otimes R \otimes v_1]_p \oplus [\tau_g \otimes N \otimes v_1]_p \oplus [T \otimes v_1]_p \\ &= (\bigoplus_{q \in \ell_j} \lambda \otimes [R]_{p,q} \otimes [v_1]_q) \oplus (\bigoplus_{q \in \ell_{j+1}} \tau_g \otimes [N]_{p,q} \otimes [v_1]_q) \oplus ([T]_p \otimes v_1) \end{aligned} \tag{3-102}$$

Again, the equation 3-102 is derived into three parts

$$\lambda \otimes [R]_p \otimes v_1 = \varepsilon \tag{3-103}$$

$$\bigoplus_{q \in \ell_1^i} \tau_g \otimes [N]_{p,q} \otimes [v_1]_q = \tau_g \otimes e \otimes \tau_g \otimes \tau_f \otimes \tau_p \tag{3-104}$$

$$\begin{aligned} \bigoplus_{q \in \ell_m} [T]_{p,q} \otimes [v_1]_q &= \tau_\Delta \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{m-1} \otimes \tau_f \otimes \tau_p \\ &\oplus \tau_p \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{m-1} \otimes \tau_f \\ &= \tau_\Delta \otimes \tau_g \otimes (\tau_p^{\otimes 2} \otimes \tau_f)^{m-1} \otimes \tau_f \otimes \tau_p \end{aligned} \tag{3-105}$$

With the results of equation 3-103 to 3-105 and the violation of assumption A2, the result of equation 3-102 is obtained as

$$\lambda \otimes \tau_g = \tau_g \otimes \tau_g \otimes \tau_f \otimes \tau_p \tag{3-106}$$

With the results above, Lemma 3-4.2 is proven for all possible cases. It should be noted that Lemma 3-4.2 is not desirable in this report, as it breaks the timing convention which is used. Therefore, assumption A2 needs to be satisfied in this report.

Furthermore, it is important to know the coupling time of the trajectory-following SMPL system. From several numerical simulations, the coupling time of the trajectory-following SMPL system is shown to be two, with one as the cyclicity. The value of the coupling time is assumed to be true if $\tau_p \otimes \tau_f \otimes \tau_\Delta \geq \tau_g$. However, the max-plus algebraic proof for the value of the coupling time is not obtained yet.

3-5 Control Structure

As the SMPL model only decides the touchdown and lift-off event time instants for every legs, SMPL model alone is not enough to generate locomotion for legged robots. Therefore, a sufficient control structure is needed for the legged robots. The control structure can be defined with the picture below[4]

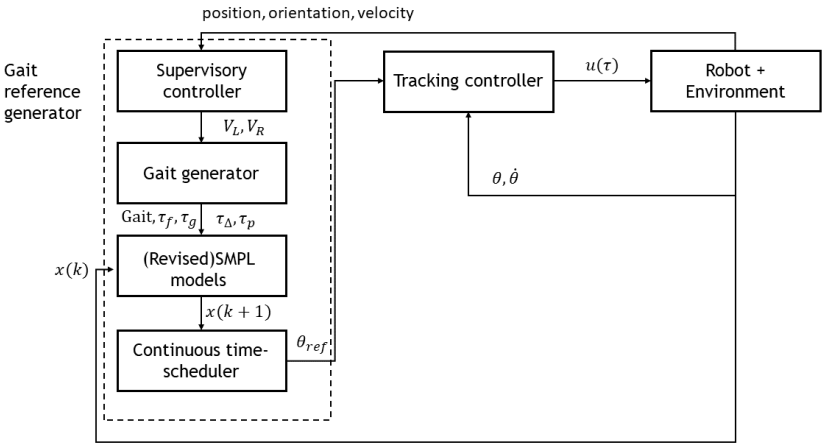


Figure 3-4: Block diagram of the control structure of legged robots

The control structure consists of several function blocks. The first one is the supervisory controller. The supervisory controller chooses the suitable linear velocity for both left and right side of the Zebro. The linear velocity of both sides are decided based on the initial and target position and orientation. The linear velocity of both sides will affect the radius of the turning if it is needed. Thus, if turning is needed, the suitable linear velocity of both sides are calculated by the supervisory controller based on the desired radius.Based on the desired velocity of both sides, the gait generator will decide the gait and the max-plus parameters (τ_f, τ_g, τ_p, and τ_Δ). The chosen gait and the max-plus parameters are chosen so the linear velocity of both sides are as the intended ones.

Based on the gait and the max-plus parameters, SMPL model will decide the touchdown and lift-off event time instant for every leg. The known touchdown and lift-off event time instants will be transformed by continuous time scheduler into reference phases for every leg. The continuous time scheduler can be defined as

$$[\theta_{\text{ref}}]_i = \begin{cases} \frac{\theta_l(t_i(k_{ti})-\tau)+(\theta_t+2\pi)(\tau-l_i(k_{li}))}{t_i(k_{ti})-l_i(k_{li})} & \text{if} \tau \in [l_i(k_{li}), t_i(k_{ti})] \\ \frac{\theta_t(l_i(k_{li}+1)-\tau)+\theta_1(\tau-t_i(k_{ti}))}{l_i(k_{li}+1)-t_i(k_{ti})} & \text{if} \tau \in [t_i(k_{ti}), l_i(k_{li}+1)] \end{cases} \tag{3-107}$$

with $[\theta_{\text{ref}}]_i$ as the reference phase for each leg i, θ_t is the fixed value of phase where the legs touchdown, and θ_l is fixed value of phase where the legs lift-off. In equation 3-107, indices $k_{\text{t}i}$ and $k_{\text{l}i}$ are used to state that the event counter for each leg i is different. The resulting $[\theta_{\text{ref}}]_i$ can be interpreted as the results of interpolation of θ_t and θ_l with current time instant τ.

For each leg, a tracking controller will enforce the reference phase. The tracking controllers are identical for every legs. The tracking controller can be implemented using phase tracking PID controller, which is defined as

$$u(\tau) = K_P(\theta_{\text{ref}}(\tau) - \theta(\tau)) + K_D(\dot{\theta}_{\text{ref}}(\tau) - \dot{\theta}(\tau)) + K_I \int_{\tau_0}^{\tau} (\theta_{\text{ref}}(s) - \theta(s))ds \tag{3-108}$$

with K_P, K_D, and K_I as the proportional, derivative, and integral parameter. The parameters $\theta(\tau)$, $\dot{\theta}(\tau)$, $\theta(s)$ are gained from the real condition by collecting the information from legged robot's sensors. The generated phase of the legs then can be used to predict the linear velocity of the legged robot, by means of the kinematics of half-circular compliant legs.

3-6 Summary

In this chapter, trajectory-following SMPL system is presented as the main proposition. The elements of trajectory-following SMPL are discussed in this chapter. Several properties of the trajectory-following SMPL system are also explained. With the trajectory-following SMPL, Zebro can turn using legs scheduling. To compare both SMPL systems, the straightforward-only SMPL system and its properties are explained. Both SMPL systems use the same control structure, which is also explained in this chapter.

Chapter 4

Kinematic Modeling of Zebro

4-1 Introduction

In chapter 3, the trajectory-following SMPL system is proposed as a gait reference generator for Zebro. Nevertheless, the trajectory-following SMPL system cannot predict position and orientation change of Zebro. The generated time instants of touchdown and lift-off alone are not enough to predict position and orientation change of Zebro. Therefore, another model is needed to predict position and orientation change of Zebro. There are two options which can be utilized. The first option is a dynamic-based model. In the dynamic-based model, the generated torque of each leg is used as the input. The second option is a kinematic-based model. The kinematic-based model uses the angular velocity of each leg as the input. The second option is chosen in this research, based on the compatibility with the trajectory-following SMPL system. In the trajectory-following SMPL system, continuous time scheduler generates reference phase for each leg. The reference phase enforces each leg to touchdown and lift-off at certain angle and certain time instant. Using the continuous time scheduler, the angular velocity of each leg can be predicted and controlled in a straightforward manner. Predicting and controlling the torque of each leg is more complicated with the trajectory-following SMPL system.

Knowing the angular velocity of a leg, the kinematics of a leg can be inferred. In section 4-2, the kinematics of a leg of Zebro is explained. As the swing period has no effect on the kinematics of Zebro, the explanation is only for stance period. Zebro is using half-circular compliant legs with their specific properties. The kinematics of Zebro itself will be derived based on kinematic modeling of a wheeled mobile robot. Therefore, in section 4-3, the kinematic modeling of a wheeled mobile robot is discussed based on [5]. The discussion is intended to describe how a wheeled mobile robot position and orientation can be predicted using the angular velocity of each leg. Based on section 4-3, the switching kinematic algorithm for a Zebro is explained in section 4-4. The kinematics of the Zebro is classified as switching kinematic because of the switching of legs on the stance period after a certain time. The legs which are on the stance period are regarded as the active legs in this report. A certain

gait will have specific legs at a certain time. The switching kinematic algorithm is the main proposition of this chapter.

4-2 Kinematic of A Half-Circular Compliant Leg

The legs of Zebro, and other robots of the RHex-type, are considered to be half-circular compliant legs [23, 24]. The term compliant arises from the fact that the legs are not rigid, but flexible. As the legs rotate, their shape change with regards to their position. The change of shape contributes to the nonlinearity of a half-circular compliant leg. Nonlinearity of a half-circular compliant leg can be observed from the position of the contact point of the leg. The position of the contact point of a leg is not vertically below the position of the corresponding hip. Then, a trigonometric relation exists between the hip position and contact position. The trigonometric relation of hip and contact position induces the nonlinearity. To understand further the morphology of the half-circular compliant leg, first figure 4-1 is presented

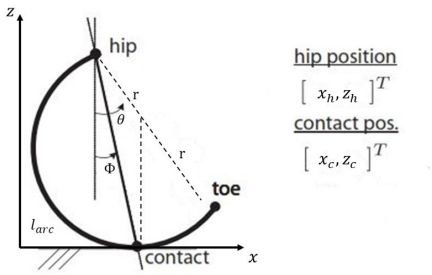


Figure 4-1: The morphology of a half-circular compliant leg

Leg angle ϕ is defined as the angle between a line from hip position to contact position and a vertical line, and the motor angle θ is defined as the angle between a line from hip position to the toe of the leg and a vertical line. The motor angle θ of a leg is intended to follow the corresponding reference phase θ_{ref}. The motor angle can also be defined as the angle between a line from hip position to the center of the leg and a vertical line. The contact point is assumed to be vertically below the center of the leg. Thus, the contact position can be defined as a function of hip position and motor angle. As a convention, the angle between a vertical line and touchdown angle has negative value in this report. The angle between a vertical line and lift-off angle has positive value. The same convention applies to the reference phase. The touchdown and lift-off angle of both leg and motor are illustrated in figure 4-2

In deriving the kinematic of a half-circular compliant leg, there are three possible approaches. These approaches are derived with regards to the nonlinearity of half-circular compliant leg. The first and second approach are presented based on [23]. In the first approach, there are two assumptions used

1. The shape of the leg is always circular with constant total arc length. The radius of the leg always change based on the corresponding leg and motor angle

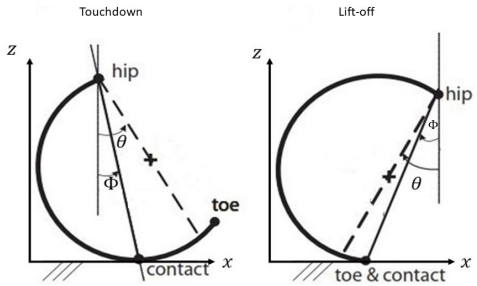


Figure 4-2: Touchdown and lift-off of a half-circular compliant leg

2. The velocity of the contact point against the ground is zero, as slipping is ignored [25]

For the first approach, the radius of a leg r is defined as

$$r = \frac{l_{\text{arc}}}{\pi - \theta} \tag{4-1}$$

Equation 4-1 arises from assumption of constant total arc length. To preserve the total arc length, the radius of the leg needs to be changed based on the motor angle. Using equation 4-1, the horizontal position of the contact point is defined as

$$\begin{aligned} x_c &= x_h - r\sin(\theta) \\ &= x_h - \frac{l_{\text{arc}}}{\pi - \theta}\sin(\theta) \end{aligned} \tag{4-2}$$

The minus term in equation 4-2 comes up from the fact that the horizontal position of the contact point is ahead of the hip at touchdown, while the touchdown angle is negative. Differentiating equation 4-2 over time, the kinematics of a half-circular compliant leg is defined as

$$\dot{x_h} = (\frac{l_{\text{arc}}\sin\theta}{(\pi - \theta)^2} + \frac{l_{\text{arc}}\cos\theta}{\pi - \theta})\dot{\theta} \tag{4-3}$$

From equation 4-3, it can be observed that the kinematics of the leg with the first approach is nonlinear. The second approach uses only the second assumption of the first approach. The radius of the leg is assumed to be constant. With the second approach, the kinematics of a half-circular compliant leg is defined as

$$\dot{x_h} = r\dot{\theta} \quad \cos(\theta) \tag{4-4}$$

Similar to the first approach, the second approach results in a nonlinear description of the kinematics of the leg. The third approach is derived based on the resemblance between the

leg and half of a wheel. Using the kinematics of a wheel for rolling without slipping motion, the kinematics of a half-circular compliant leg is derived as

$$\dot{x_h} = 2r\dot{\theta} \tag{4-5}$$

Equation 4-5 is obviously a linear one, different with the first two approaches. Observing equation 4-3 and 4-4, the nonlinearity of the kinematics of the leg will cause acceleration on the hip. The velocity of the hip will not be constant because of the nonlinear terms, even as the angular velocity of the leg is constant. The acceleration might cause an error in the position and orientation prediction of Zebro.

In this report, the third approach is used. The main reason is that the legs which are used in both simulation and implementation are not compliant. Therefore, with rigid legs, it is more suitable to model the legs as half-wheels. Furthermore, in the first and second approach, terms $\cos(\theta)$ and $\sin(\theta)$ exist. The terms indicate that $\dot{x_h}$ depends on the position of the leg as well for both approaches. However, the kinematic modeling framework which will be used does not take position of legs as inputs. Then, the first and second approach do not suit the framework which will be used.

4-3 Kinematic of Wheeled Mobile Robot

The kinematics of a Zebro can be derived from the kinematic modeling of a wheeled mobile robot with non-steered wheels. The morphology of Zebro is similar to nonholonomic wheeled mobile robot with no steering. The movement of Zebro is also based on differential drive of the legs, similar to some wheeled mobile robots. It is different with how some legged robots move, which are by foot placement. For those reasons, the kinematic of Zebro resembles the kinematics of wheeled mobile robots with non-steered wheels. In this section, the derivation of the general approach to kinematics of wheeled mobile robot is explained. The explained approach is based on [5]. The literature [5] uses Sheth-Uicker conventions to decide the coordinate frame and derive the homogeneous matrices. The homogeneous matrices are used to describe the positional relation between the wheels, the body, and the world frame. With the homogeneous matrices, jacobian matrix of each is derived. The jacobian matrix relates angular velocity of a wheel and the velocity of vehicle. Then, combining jacobian matrices of active legs, the kinematics of a wheeled mobile robot with non-steered wheels is obtained.

4-3-1 Definitions and Conventions

In this research, the kinematic modeling is done based on Sheth-Uicker convention. In the Sheth-Uicker convention, two coordinate frames are assigned at a joint. Each coordinate frame is assigned at each link, as illustrated in figure 4-3. The considered links are the floor, the robot body, and the steering links. The joints are a revolute pair at each steering axis, and a planar pair to model each wheel. Each revolute pair connects a steering link and the robot body. Each planar pair of a wheel connects the floor and a steering link. The wheeled mobile robot then can be considered as multiple closed-link chains. Thus, using Sheth-Uicker convention, there is no confusion in assigning the coordinate frame for multiple closed-link chains

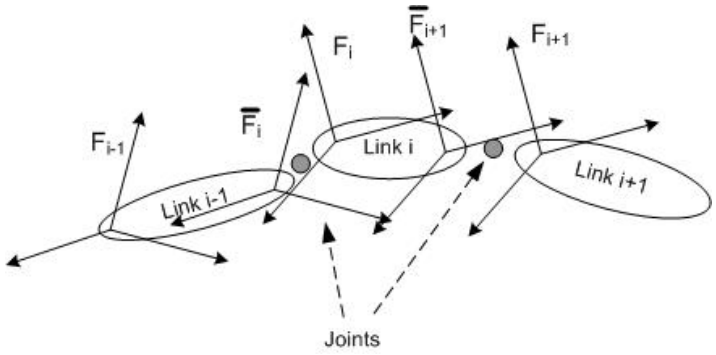


Figure 4-3: Illustration of Sheth-Uicker convention

Based on the existing joints and links, coordinate frames are assigned. The coordinate frame floor, F, is assigned as a stationary reference frame at the floor. Coordinate frame steering S_i and contact point C_i are assigned at the steering link i. The coordinate frame S_i is located near the revolute pair of the steering link i, and the coordinate frame C_i is located near the planar pair of the steering link i. For the robot body, coordinate frame robot R is assigned at the center of the robot body. Coordinate frame hip H_i is also assigned at the robot body, near the revolute pair of the steering link i.

Corresponding to coordinate frame R and C_i, instantaneous coincident coordinate frame $\bar{R}$ and $\bar{C}_i$ are defined. The coordinate frame $\bar{R}$ is instantaneously coincident with the coordinate frame R and stationary to the coordinate frame F. The coordinate frame $\bar{C}_i$ is instantaneously coincident with the coordinate frame C and stationary to the coordinate frame F. As both $\bar{R}$ and $\bar{C}_i$ are instantaneous coincident, the position of both are redefined as the coordinate frame R and C_i move. With both $\bar{R}$ and $\bar{C}_i$, the velocity of each contact point and robot can be defined independent to the position of R and C_i.

To relate two coordinate frames, a transformation matrix is defined. The transformation matrix ${}^{A}\Pi_{B}$ transform the defining coordinate of a point p from the coordinate frame B to the coordinate frame A. This relation can be defined in an equation as

$$ {}^{A}p = {}^{A}\Pi_{B}\,{}^{B}p \tag{4-6} $$

where ${}^{A}p$ and ${}^{B}p$ denote point p defined in coordinate frame A and B respectively. The transformation matrix ${}^{A}\Pi_{B}$ and point ${}^{A}p$ themselves can be defined as

$$ {}^{A}\Pi_{B} = \begin{pmatrix} \cos{}^{A}\psi_{B} & -\sin{}^{A}\psi_{B} & 0 & {}^{A}d_{Bx} \\ \sin{}^{A}\psi_{B} & \cos{}^{A}\psi_{B} & 0 & {}^{A}d_{By} \\ 0 & 0 & 1 & {}^{A}d_{Bz} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4-7} $$

$$ {}^{A}p = \begin{pmatrix} {}^{A}p_{x} \\ {}^{A}p_{y} \\ {}^{A}p_{z} \\ 1 \end{pmatrix} \tag{4-8} $$

with ${}^{A}\psi_{B}$ as the angle between the origin of coordinate frame A and B, ${}^{A}d_{Bx}$ as the distance between the origin of coordinate frame A and B in x direction, and ${}^{A}p_{x}$ as the position of point p with respect to coordinate frame A in x direction. The same convention applies to y and z direction as well. The concept of transformation matrix in 4-7 is used to define the relation between the coordinate frames in wheeled mobile robot. The transformation matrix ${}^{A}\psi_{B}$ also can be differentiated over time to define the velocity of the point, defined as

$$ {}^{A}\dot{p} = {}^{A}\dot{\Pi}_{B}{}^{B}p \tag{4-9} $$

Here, ${}^{A}\dot{p}$ is defined as the velocity of point p defined in coordinate frame A and ${}^{A}\dot{\Pi}_{B}$ is the differentiation of transformation matrix over time. The elements of ${}^{A}\dot{p}$ and ${}^{A}\dot{\Pi}_{B}$ are the differentiation of the elements of ${}^{A}p$ and ${}^{A}\Pi_{B}$ over time. Vector ${}^{A}\dot{p}_{B}$ also can be used to defined the velocity of frame B relative to frame A. There are several useful axioms for the transformation matrix, which are

- Identity: ${}^{A}\Pi_{A} = I$
- Cascade : ${}^{A}\Pi_{B} = {}^{A}\Pi_{X}\ {}^{X}\Pi_{B}$
- Inversion : ${}^{A}\Pi_{B} = ({}^{B}\Pi_{A})^{-1}$
- Zero velocity : ${}^{A}\dot{\Pi}_{B} = 0$
- Velocity : ${}^{A}\dot{\Pi}_{B} = {}^{A}\dot{\Pi}_{X}\ {}^{X}\Pi_{B} + {}^{A}\Pi_{X}\ {}^{X}\dot{\Pi}_{B}$

Because of the Sheth-Uicker convention, the transformation matrix ${}^{A}\Pi_{B}$ is divided into two kinds. They are the constant transformation matrix ${}^{A}T_{B}$ and variable transformation matrix ${}^{A}\Phi_{B}$. The elements of the constant transformation matrix ${}^{A}T_{B}$ have fixed values, whereas the elements of variable transformation matrix ${}^{A}\Phi_{B}$ change. In general, for wheeled mobile robot, the constant transformation matrices are including

- ${}^{F}T_{\bar{R}}$: floor-instantaneous coincident robot body frame transformation matrix
- ${}^{F}T_{\bar{C}_i}$: floor-instantaneous coincident contact point frame transformation matrix
- ${}^{R}T_{H_i}$: robot body-hip transformation matrix for steered wheels
- ${}^{S_i}T_{C_i}$: steering-contact point transformation matrix for steered wheels
- ${}^{R}T_{C_i}$: robot body-contact point transformation matrix for non-steered wheels

The non-steered wheel has ${}^{R}T_{C_i}$ as the transformation matrix between hip and steering is the identity matrix. With no steering, the coordinate frame of hip and steering coincide. As the elements of constant transformation matrices do not change, zero velocity axiom applies to them. Furthermore, the variable transformation matrices are including

- ${}^{\bar{R}}\Phi_{R}$: instantaneous coincident robot body-robot body frame transformation matrix

- ${}^{H_i}\Phi_{S_i}$: hip-steering frame transformation matrix for steered wheels
- ${}^{\bar{C}_i}\Phi_{C_i}$: instantaneous coincident contact point-contact point frame transformation matrix

Those matrices are categorized as variable transformation matrices because of the relative motion between the frames. Then, the velocity axiom applies to those matrices. Identity axiom also applies to those matrices because of the instantaneous coincident property. For non-steered wheels, ${}^{H_i}\Phi_{S_i}$ is an identity matrix.

4-3-2 Wheel Jacobian Matrices

A wheel jacobian matrix is constructed to determine the velocity of robot body caused by the motion of a leg i. First, the transformation matrix ${}^{\bar{R}}\Phi_R$ needs to be defined as the cascade of other transformation matrices. The cascade is needed as the elements of ${}^{\bar{R}}\Phi_R$ are unknown and always change. Transformation matrix ${}^{\bar{R}}\Phi_R$ is defined as

$$ {}^{\bar{R}}\Phi_R = {}^{F}T_{\bar{R}}^{-1}{}^{F}T_{\bar{C}_i}\,{}^{\bar{C}_i}\Phi_{C_i}\,{}^{S_i}T_{C_i}^{-1}{}^{H_i}\Phi_{S_i}^{-1}{}^{R}T_{H_i}^{-1} \tag{4-10} $$

Equation 4-10 can be differentiated over time, applying both velocity axiom and zero velocity axiom. Zero velocity axiom is applied to the constant transformation matrices. The differentiation ${}^{\bar{R}}\dot{\Phi}_R$ is defined as

$$ {}^{\bar{R}}\Phi_R = {}^{F}T_{\bar{R}}^{-1}{}^{F}T_{\bar{C}_i}\,{}^{\bar{C}_i}\dot{\Phi}_{C_i}\,{}^{S_i}T_{C_i}^{-1}{}^{H_i}\Phi_{S_i}^{-1}{}^{R}T_{H_i}^{-1} + {}^{F}T_{\bar{R}}^{-1}{}^{F}T_{\bar{C}_i}\,{}^{\bar{C}_i}\Phi_{C_i}\,{}^{S_i}T_{C_i}^{-1}{}^{H_i}\dot{\Phi}_{S_i}^{-1}{}^{R}T_{H_i}^{-1} \tag{4-11} $$

In equation 4-11, the elements of ${}^{F}T_{\bar{C}_i}$ are unknown. To define ${}^{F}T_{\bar{C}_i}$, the cascade axiom is used as in equation 4-10. Then, transformation matrix ${}^{F}T_{\bar{C}_i}$ is defined as

$$ {}^{F}T_{\bar{C}_i} = {}^{F}T_{\bar{R}}\,{}^{\bar{R}}\Phi_R\,{}^{R}T_{H_i}\,{}^{H_i}\Phi_{S_i}\,{}^{S_i}T_{C_i}\,{}^{\bar{C}_i}\Phi_{C_i}^{-1} \tag{4-12} $$

Equation 4-12 replaces the matrix ${}^{F}T_{\bar{C}_i}$ in equation 4-11, resulting in

$$ {}^{\bar{R}}\Phi_R = {}^{\bar{R}}\Phi_R\,{}^{R}T_{H_i}\,{}^{H_i}\Phi_{S_i}\,{}^{S_i}T_{C_i}\,{}^{\bar{C}_i}\Phi_{C_i}^{-1}{}^{\bar{C}_i}\dot{\Phi}_{C_i}\,{}^{S_i}T_{C_i}^{-1}{}^{H_i}\Phi_{S_i}^{-1}{}^{R}T_{H_i}^{-1} + {}^{\bar{R}}\Phi_R\,{}^{R}T_{H_i}\,{}^{H_i}\Phi_{S_i}\,{}^{H_i}\dot{\Phi}_{S_i}^{-1}{}^{R}T_{H_i}^{-1} \tag{4-13} $$

Applying identity axiom to equation 4-13, the equation becomes

$$ {}^{\bar{R}}\Phi_R = {}^{R}T_{H_i}\,{}^{H_i}\Phi_{S_i}\,{}^{S_i}T_{C_i}\,{}^{\bar{C}_i}\dot{\Phi}_{C_i}\,{}^{S_i}T_{C_i}^{-1}{}^{H_i}\Phi_{S_i}^{-1}{}^{R}T_{H_i}^{-1} + {}^{R}T_{H_i}\,{}^{H_i}\Phi_{S_i}\,{}^{H_i}\dot{\Phi}_{S_i}^{-1}{}^{R}T_{H_i}^{-1} \tag{4-14} $$

From the elements of the resulting matrix, robot body velocity vector ${}^{\bar{R}}\dot{p}_R$ is obtained as

$$
\begin{aligned}
{}^{\bar{R}}\dot{p}_R &= \begin{bmatrix} \cos{}^R\psi_{C_i} & -\sin{}^R\psi_{C_i} & {}^Rd_{C_{iy}} & -{}^Rd_{H_{iy}} \\ \sin{}^R\psi_{C_i} & \cos{}^R\psi_{C_i} & -{}^Rd_{C_{ix}} & {}^Rd_{H_{ix}} \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} {}^{\bar{C}_i}v_{C_{ix}} \\ {}^{\bar{C}_i}v_{C_{iy}} \\ {}^{\bar{C}_i}w_{C_i} \\ {}^{H_i}w_{S_i} \end{bmatrix} \\
&= \hat{J}_i\dot{\hat{q}}_i
\end{aligned} \tag{4-15}
$$

The matrix $\hat{J}_i$ is the pseudo-Jacobian matrix of wheel i, while vector $\dot{\hat{q}}_i$ is the pseudo-velocity vector of wheel i. It is called pseudo-velocity vector as the elements of the vector are not the physical elements of a wheel. To relate the pseudo-velocity of a wheel with the real velocity $\dot{q}_i$, wheel matrix W_i of wheel i is used as in

$$
\dot{\hat{q}}_i = W_i\dot{q}_i \tag{4-16}
$$

Substituting equation 4-16 into equation 4-15, the Jacobian matrix is obtained as

$$
\begin{aligned}
{}^{\bar{R}}\dot{p}_R &= \hat{J}_iW_i\dot{q}_i \\
&= J_i\dot{q}_i
\end{aligned} \tag{4-17}
$$

4-3-3 Sensed Forward Solution

The Jacobian matrix only defines the relation between the velocity of a wheel and the velocity of robot. In practice, there are several active wheels. To understand the relation between the velocity of active wheels with the velocity of the robot, sensed forward solution of the robot is derived as in [5]. First, the real velocity vector $\dot{q}_i$ is divided into the sensed and actuated variable $\dot{q}_{is}$ and the not-sensed and not-actuated variable ($\dot{q}_{in}$). Then, the Jacobian matrix J_i is also divided into the sensed and actuated part J_{is} and the not-sensed and not-actuated part J_{in}. The division of Jacobian matrix J_i is done corresponding to the position of elements in $\dot{q}_{is}$ and $\dot{q}_{in}$ in real velocity vector $\dot{q}_i$. Writing ${}^{\bar{R}}\dot{p}_R$ as $\dot{p}$, equation 4-17 is transformed into

$$
\dot{p} = J_{is}\dot{q}_{is} + J_{in}\dot{q}_{in} \tag{4-18}
$$

for wheel i. Then for i = 1,...,N we can set matrices

$$
\begin{pmatrix} I_1 & -J_{1n} & 0 & ... & 0 \\ I_2 & 0 & -J_{2n} & ... & 0 \\ : & : & ... & ... & 0 \\ I_N & 0 & ... & 0 & -J_{Nn} \end{pmatrix} \begin{pmatrix} \dot{p} \\ \dot{q}_{1n} \\ \dot{q}_{2n} \\ : \\ \dot{q}_{Nn} \end{pmatrix} = \mathrm{diag}(J_{1s}, J_{2s}, ..., J_{Ns}) \begin{pmatrix} \dot{q}_{1s} \\ \dot{q}_{2s} \\ : \\ \dot{q}_{Ns} \end{pmatrix} \tag{4-19}
$$

which in general form can be written as

$$
A_n\dot{p}_n = B_s\dot{q}_s \tag{4-20}
$$

Then, with least-squares method as in [5], the velocities vector of the robot and not-sensed wheel velocities can be calculated with

$$\dot{p}_n = (A_n^T A_n)^{-1} A_n^T B_s \dot{q}_s \tag{4-21}$$

4-4 Switching Kinematic of Zebro

The kinematics of a Zebro can be inferred with the method in 4-3. However, there is a difference between the motion of a wheeled mobile robot and the Zebro. For a wheeled mobile robot, the active legs are always the same. If a wheeled mobile robot has four wheels, then those wheels will always be utilized. The active legs of the Zebro always change, depend on which legs are on stance period. The legs which are on swing period do not affect the motion of the Zebro, as they do not interact with the ground. Therefore, the kinematics of the Zebro can be regarded as switching kinematics. The kinematic relation of Zebro will depend on the chosen gait and the corresponding period of time. Moreover, Zebro has half-circular compliant legs instead of wheels. Thus, several modifications of the method proposed in [5] are considered.

In this research, the kinematics of a half-circular compliant leg is considered as in equation 4-5. This approach suits the proposed method, as it focuses on the resemblance between a half-circular compliant leg and a half wheel. This approach also simplifies the calculation, as the equation 4-5 is linear. The linearity of equation 4-5 also means that the angular velocity $\dot{\theta}$ is the only used input. The motor angle at the particular time instant is not needed as the input. Nevertheless, this approach might cause an error. Using this approach, the obtained linear velocity of a leg is constant. With nonlinearity included, the linear velocity of a leg is changing based on the motor angle.

Based on equation 4-5, the wheel matrix and the real velocity vector of a half-circular compliant leg i can be defined as

$$W_i = \begin{bmatrix} 0 & 0 \\ 2r & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{4-22}$$

$$\dot{q}_i = \begin{bmatrix} w_{ix} \\ w_{iz} \end{bmatrix} \tag{4-23}$$

Combining the wheel matrix and the real velocity vector, the pseudo-velocity vector $\dot{\hat{q}}_i$ is defined as

$$\begin{bmatrix} {}^{\bar{C}_i}v_{C_{ix}} \\ {}^{\bar{C}_i}v_{C_{iy}} \\ {}^{\bar{C}_i}w_{C_i} \\ {}^{H_i}w_{S_i} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 2r & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w_{ix} \\ w_{iz} \end{bmatrix}$$
$$\begin{bmatrix} {}^{\bar{C}_i}v_{C_{ix}} \\ {}^{\bar{C}_i}v_{C_{iy}} \\ {}^{\bar{C}_i}w_{C_i} \\ {}^{H_i}w_{S_i} \end{bmatrix} = \begin{bmatrix} 0 \\ 2rw_{ix} \\ w_{iz} \\ 0 \end{bmatrix} \tag{4-24}$$

Observing the equation 4-24, the second row of 4-22 is chosen as the longitudinal velocity of the Zebro is affected by the angular velocity of the Zebro around the lateral axis w_{ix}. The third row of 4-22 means that the yaw angular velocity of the Zebro is equal to the angular velocity of the rotational slip w_{iz}. The first and last row of 4-22 are zero as the motion of a leg does not result in the lateral velocity of the Zebro, and all legs are not steered. Using the pseudo-Jacobian matrix $\hat{J}_i$ in 4-15 and the wheel matrix, the jacobian matrix of a half-circular compliant leg i is defined as

$$J_i = \begin{bmatrix} -2r\sin{}^{R}\psi_{C_i} & {}^{R}d_{C_{iy}} \\ 2r\cos{}^{R}\psi_{C_i} & -{}^{R}d_{C_{ix}} \\ 0 & 1 \end{bmatrix} \tag{4-25}$$

For the Zebro purpose, all legs have the same radius r. The jacobian matrices of all legs will be different as each leg has its own lateral and longitudinal position with respect to the frame of the robot body. For six-legs Zebro,it is illustrated as

Figure 4-4: The coordinate frame assignments and the dimension of Zebro

On the figure 4-4, an example is given for coordinate frame assignment. The robot body coordinate frame is assumed to be located at the center of Zebro. The contact point coordinate frame of leg i is assumed to be located vertically below the hip coordinate frame of leg i. Related to the coordinate frames, the dimension of Zebro is also explained by figure 4-4 along with the numbering of every leg. In the figure 4-4, every distance of the contact point frames to the robot body coordinate frame is considered to be symmetric. Then, using the information from figure 4-4, the Jacobian matrices of all legs are defined as

$$J_1 = \begin{bmatrix} 0 & l \\ 2r & b \\ 0 & 1 \end{bmatrix} \quad J_2 = \begin{bmatrix} 0 & l \\ 2r & -b \\ 0 & 1 \end{bmatrix} \tag{4-26}$$

$$J_3 = \begin{bmatrix} 0 & 0 \\ 2r & b \\ 0 & 1 \end{bmatrix} \quad J_4 = \begin{bmatrix} 0 & 0 \\ 2r & -b \\ 0 & 1 \end{bmatrix} \tag{4-27}$$

$$J_5 = \begin{bmatrix} 0 & -l \\ 2r & b \\ 0 & 1 \end{bmatrix} \quad J_6 = \begin{bmatrix} 0 & -l \\ 2r & -b \\ 0 & 1 \end{bmatrix} \tag{4-28}$$

The Jacobian matrices are then used to generate the sensed forward solution. In the sensed forward solution for Zebro, w_{ix} is the sensed and actuated variable and w_{iz} is the not-sensed and not-actuated variable. Based on the active legs, the Jacobian matrices of the active legs are divided as in 4-18. Next, equation 4-19 is formed with variable w_{iz} of active legs on the left side and variable w_{ix} of active legs on the right side. The least square solution of equation4-21 then can be calculated. In this research, only the first three rows of the results are considered. The rotational slip of the legs are not part of this research. With SMPL framework, the least square solution in equation 4-21 can be modified. The modification is possible as the real stance time of the legs can be calculated using the SMPL framework. Multiplying both sides of equation 4-21 with the time results in forward displacement solution, written as

$$\begin{bmatrix} {}^{\bar{R}}\Delta x_{\bar{R}} \\ {}^{\bar{R}}\Delta y_{\bar{R}} \\ {}^{\bar{R}}\Delta \alpha_{\bar{R}} \\ \Delta\alpha_{1z} \\ . \\ . \\ \Delta\alpha_{Nz} \end{bmatrix} = (A_n^T A_n)^{-1} A_n^T B_s \begin{bmatrix} \Delta\theta_{1x} \\ \Delta\theta_{2x} \\ . \\ . \\ \Delta_{Nx} \end{bmatrix} \tag{4-29}$$

In equation 4-29, ${}^{\bar{R}}\Delta x_{\bar{R}}$, ${}^{\bar{R}}\Delta y_{\bar{R}}$, and ${}^{\bar{R}}\Delta z_{\bar{R}}$ are the displacement of Zebro body coordinate frame in lateral, longitudinal, and vertical direction respectively. ${}^{\bar{R}}\Delta\alpha_{\bar{R}}$ is the yaw angle change of Zebro body coordinate frame. The change of motor angle of the active legs are used as the inputs here, with $\Delta\theta_{ix}$ as the change of motor angle of leg i. The change of motor angle $\Delta\theta_{ix}$ can be obtained by using equation 3-107. The displacement of Zebro in equation 4-29 is independent to the world frame. To configure the displacement of Zebro relative to the world frame, another transformation is needed. The transformation of Zebro relative to the world frame is defined as

$$\begin{bmatrix} {}^{F}\Delta x_R \\ {}^{F}\Delta y_R \\ {}^{F}\Delta \alpha_R \end{bmatrix} = \begin{bmatrix} \cos{}^{F}\psi_R & -\sin{}^{F}\psi_R & 0 \\ \sin{}^{F}\psi_R & \cos{}^{F}\psi_R & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{\bar{R}}\Delta x_{\bar{R}} \\ {}^{\bar{R}}\Delta y_{\bar{R}} \\ {}^{\bar{R}}\Delta \alpha_{\bar{R}} \end{bmatrix} \tag{4-30}$$

However, the result of equation 4-30 cannot be obtained in a straightforward way. The dimension of matrices and vectors in equation 4-29 are not constant. The dimension of

matrices and vectors in equation 4-29 depends on the active legs during a certain duration of time. A function is presented below to apply equation 4-29 based on the set of active legs

Function 1 Sensed forward solution for active legs

Require: J_{i_s}, J_{i_n} of all legs

1: **for all** $i \in \mathcal{A}$ **do**

2: $I_i \leftarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

3: $I_A \leftarrow \mathsf{Vectorize}(I_i)$

4: $A_n \leftarrow \mathsf{Diagonalize}(-J_{i_n})$

5: $B_s \leftarrow \mathsf{Diagonalize}(J_{i_s})$

6: **end for**

7: $A_n \leftarrow \begin{bmatrix} I_A & A_n \end{bmatrix}$

8: $\Delta p_n = (A_n^T A_n)^{-1} A_n^T B_s \Theta$

9: **return** $\begin{bmatrix} {}^{\bar{R}}\Delta x_{\bar{R}} & {}^{\bar{R}}\Delta y_{\bar{R}} & {}^{\bar{R}}\Delta \alpha_{\bar{R}} \end{bmatrix}^T \leftarrow \begin{bmatrix} \Delta p_n(1) & \Delta p_n(2) & \Delta p_n(3) \end{bmatrix}^T$ The first three elements of Δp_n

The function 1 will adapt the size of matrices in equation 4-29 based on the number of active legs. As the inputs, current active legs $\mathcal{A}$ and the change of motor angle of active legs Θ are used. The jacobian matrices of all legs are also needed in the function. Every jacobian matrices also must be separated, to J_{i_s} and J_{i_n} for the jacobian matrix i. To start the function, in line 1, $\mathcal{A}$ is defined as the working set of the function. Then, for every element of $\mathcal{A}$, an identity matrix of a leg i is defined in line 2. In the line 3, Vectorize means constructing a new matrix by adding down a identity matrix for every leg in$\mathcal{A}$ vertically. Diagonalize, in line 4 and 5, means constructing a new matrix by adding down a $-J_{i_n}$ and a J_{i_s} for every legs in $\mathcal{A}$ diagonally. The function 1 results in the displacement of and the yaw angle change of the Zebro body coordinate frame. The function 1 is then used in the algorithm below

In the algorithm 2, first the initial position and heading angle is prepared from line 2 to 4. Initial max-plus states x_{initial} are also prepared in line 1. As explained in line 9, the iteration is done from k equals to zero until the final iteration k_{final}. The value of k_{final} can be regarded as the prediction horizon of switching kinematic algorithm. Then, using the system matrix $G^{\star}$, the next two states are generated in line 10 and 11. Knowing the next two states, the maximum touchdown time instant of both states are used as the time boundary in detecting the active legs.

Algorithm 2 Switching kinematic algorithm

1: $x(0) \leftarrow x_{\text{initial}}$: Initialize $x(0)$ with the initial states
2: ${}^{F}x_R \leftarrow {}^{F}x_R$: Initialize initial x-position
3: ${}^{F}y_R \leftarrow {}^{F}y_R$: Initialize initial y-position
4: ${}^{F}\alpha_R \leftarrow {}^{F}\alpha_R$: Initialize heading angle
5: ${}^{\bar{R}}\Delta x_{R_p} \leftarrow 0$
6: ${}^{\bar{R}}\Delta y_{R_p} \leftarrow 0$
7: $\mathcal{A} \leftarrow \{\}$
8: $\Theta \leftarrow \{\}$
9: **for** $k = \{0, 1, ..., k_{\text{final}}\}$ **do**
10: $x(k+1) \leftarrow G^{\star} \otimes x(k)$
11: $x(k+2) \leftarrow G^{\star} \otimes x(k+1)$
12: $t_{\min} \leftarrow \max\, x(k)$
13: $t_{\max} \leftarrow \max\, x(k+1)$
14: **for** $\tau = \{t_{\min} + 0.01, t_{\min} + 0.02\}, ..., t_{\max}$ **do**
15: **for** $i = \{1, 2, ..., 6\}$ **do**
16: **if** $t_i(k) \leq \tau < l_i(k+1)$ **then**
17: $\mathcal{A} \leftarrow \mathcal{A} \cup i$
18: $\Delta\theta_{ix} \leftarrow \theta_{\text{ref}}(t_i(k), l_i(k+1), \tau) - \theta_{\text{ref}}(t_i(k), l_i(k+1), \tau - 0.01)$: see equation 3-107
19: $\Theta \leftarrow \Theta \cup \Delta\theta_{ix}$
20: **else if** $t_i(k+1) \leq \tau < l_i(k+2)$ **then**
21: $\mathcal{A} \leftarrow \mathcal{A} \cup i$
22: $\Delta\theta_{ix} \leftarrow \theta_{\text{ref}}(t_i(k+1), l_i(k+2), \tau) - \theta_{\text{ref}}(t_i(k+1), l_i(k+2), \tau - 0.01)$: see equation 3-107
23: $\Theta \leftarrow \Theta \cup \Delta\theta_{ix}$
24: **end if**
25: **end for**
26: $[{}^{\bar{R}}\Delta x_R, {}^{\bar{R}}\Delta y_R, {}^{\bar{R}}\Delta\alpha_R] \leftarrow \mathsf{SensedForward}(\mathcal{A}, \Theta)$
27: ${}^{F}x_R \leftarrow {}^{F}x_R + \cos(\alpha)\frac{{}^{\bar{R}}\Delta x_R + {}^{\bar{R}}\Delta x_{R_p}}{2} - \sin(\alpha)\frac{{}^{\bar{R}}\Delta y_R + {}^{\bar{R}}\Delta y_{R_p}}{2}$
28: ${}^{F}y_R \leftarrow {}^{F}y_R + \sin(\alpha)\frac{{}^{\bar{R}}\Delta x_R + {}^{\bar{R}}\Delta x_{R_p}}{2} + \cos(\alpha)\frac{{}^{\bar{R}}\Delta y_R + {}^{\bar{R}}\Delta y_{R_p}}{2}$
29: ${}^{F}\alpha_R \leftarrow {}^{F}\alpha_R + {}^{\bar{R}}\Delta\alpha_R$
30: ${}^{\bar{R}}\Delta x_{R_p} \leftarrow {}^{\bar{R}}\Delta x_R$
31: ${}^{\bar{R}}\Delta y_{R_p} \leftarrow {}^{\bar{R}}\Delta y_R$
32: $\mathcal{A} \leftarrow \{\}$
33: $\Theta \leftarrow \{\}$
34: **end for**
35: **end for**

The detection of the active legs is done from line 14 to line 25, for every τ. From line 16 to line 19, for a particular τ, it is checked if a leg i is on the stance period at τ second. The checking is done between state k and $k+1$. If the leg i is on the stance period, then the value of i is included in the set $\mathcal{A}$. The motor angle change of the leg i is also calculated, for 0.01 seconds. The calculation is done with regards to the equation 3-107. The motor angle change of the leg i is then included in the set Θ. These steps are repeated for all legs. The

same procedures are also done from line 20 to 24, now for τ between $k+1$ and $k+2$.

The set $\mathcal{A}$ is defined as the set of active legs. This set will have a certain pattern, depending on the chosen gait and direction. As an example, the set $\mathcal{A}$ for a Zebro with left as the intended direction and $\{1,4,5\},\{2,3,6\}$ as the gait is

$$\{2,3,6,1,5\},\{2,3,6,1,4,5\},\{1,4,5\},\{1,4,5,3\},\{1,4,5,2,3,6\},\{2,3,6\}$$

Both $\mathcal{A}$ and Θ are then used to generate the change of position and yaw angle of the Zebro relative to instantaneous coincident frame of the Zebro, by using the function 1. From line 27 to 29, the position and yaw angle of the Zebro relative to the world frame is calculated. The calculation is done based on the equation in [5]. In the calculation, ${}^{\bar{R}}\Delta x_{R_p}$ and ${}^{\bar{R}}\Delta y_{R_p}$ are the previous position and yaw angle change. If the value of ${}^{F}x_R$ and ${}^{F}y_R$ are stored, the trajectory prediction of the Zebro is obtained by using the switching kinematic algorithm.

4-5 Summary

The foundation of the kinematic modeling of a Zebro is presented in this chapter. The kinematic modeling of the Zebro is based on the kinematic modeling of the wheeled mobile robot, presented in [5]. The kinematic model in [5] is constructed based on the Sheth-Uicker convention, which decided where coordinate frames should be placed. Then, Jacobian matrices of every leg can be derived from the relation between coordinate frames. The sensed forward solution is finally used to predict the motion of Zebro by combining the Jacobian matrices. However, as the legs which are in the stance period are always switching, the kinematic model of Zebro also changes periodically. The switching kinematic algorithm is proposed to predict the kinematic model of Zebro at a particular time. The corresponding Jacobian matrices are used to generate the suitable sensed forward solution.

Chapter 5

Simulation and Implementation

5-1 Introduction

In this chapter, the simulation and implementation results of Zebro with trajectory-following SMPL systems are presented. The simulation results are the trajectory generated by computer-aided design (CAD) model of Zebro, called as virtual Zebro, during the 2D planar simulation in V-REP. The implementation results are the trajectory generated by physical Zebro during the actual trial. Before the results are presented, the properties of both simulation and implementation are explained in 5-2. The properties of the simulation are explained in 5-2-1. The explained properties for the simulation are including the descriptions of the used software and the setting of the simulation. Next, the properties of implementation are explained in 5-2-2. The specifications of Zebro and the setting of the implementation are explained in 5-2-2.

Based on section 5-2, both simulation and implementation are done. The purposes of both simulation and implementation are to know the turning characteristics of the trajectory-following SMPL system and to verify the performance of switching kinematic. The turning characteristics are including the radius of turning and the displacement of turning Zebro. These characteristics are influenced by the gait and the parameters of the trajectory-following SMPL system. Then, in both simulation and implementation, the parameters of the trajectory-following SMPL system are divided into constant parameters and changing parameters. The changing parameters are the parameters which influence the turning characteristics directly, whereas the constant parameters are not. All considered gaits are also tried in both simulation and implementation. The turning characteristics and the generated trajectory are both considered as the results, which are presented in section 5-3. The results of simulation are presented in 5-3-1 and the result of implementation in 5-3-2. The trajectory generated by switching kinematic for a particular gait and set of parameters is compared with the trajectory generated in both simulation and implementation, for the same gait and parameters. In doing both simulation and implementation, there are two hypotheses proposed :

Hypotheses :

1. For larger τ_p, the radius of turning is smaller
2. The switching kinematic model can predict the motion of virtual Zebro

The hypotheses are related to the research questions in this report.

5-2 Simulation and Implementation Properties

5-2-1 Simulation Properties

In designing gait reference generator for a legged robot, simulation can be used to test the performance of the gait reference generator. With simulation, gait reference generator can be tested in a quick, economical and efficient way. The considered simulation here is a 2D planar simulation, as this research only considers motion on flat terrain. In general, there are two platforms which are needed for 2D planar simulation

1. Algorithm Programming Environment
   In this platform, the gait reference generator will be developed, debugged and tested. Usually, numerical computational software is used for this purpose. There are several common options such as Matlab, Scilab, GNU Octave, and Phyton. Those software are open-sourced, except for Matlab.
2. Physical Simulator
   Physical Simulator will be used to generate motion of a legged robot based on the chosen gait reference generator. The motion is generated by the physical simulator based on the kinematic and/or dynamic of the CAD model. The physical simulator also shows the resulting motion virtually, in 3D perspective. There are several software for this purpose, such as V-Rep, Gazebo, and Webots.

In our research, Matlab is used as the algorithm programming environment. The main reason is that Matlab is mathematically considered to be very reliable, with wide variations of functions. Furthermore, the individuals involved in this research are more familiar with Matlab than the other numerical computational software. Consequently, the chosen physical simulator has to be compatible with Matlab.

In this research, V-REP is chosen as the physical simulator. V-REP is chosen as it can be connected with Matlab easily, and is already used for simulation of legged robots. V-REP itself is created by Coppelia Robotics to provide a versatile and scalable simulation framework. There are four kinds of available license for V-REP, Player(open source), Pro Edu(open source for educational), Pro Eval (open source, except for commercial use) and Pro(commercial use). V-REP enables users to utilize various relatively independent functionalities. Those functionalities can be enabled and disabled by the users. Therefore, V-REP does not have main functionality.

The functionality of V-REP is related to two elements in V-REP, scene objects, and calculation modules. Scene objects are elemental objects that define how the 3D simulation will be done. Scene objects operate with other scene objects to form the 3D simulation. Some examples of

scene objects are joints, shapes, sensors, paths, cameras, and lights. The scene objects can be modeled directly in V-REP, with click and drag approach. Calculation modules define the purpose of the simulation, where each purpose can be done independently. There are five calculation modules, which are kinematic module, dynamics module, collision detection module, mesh-mesh distance calculation module, and path/motion planning module. These modules can directly operate on one or several scene objects. The dynamics module uses a physic engine to do a hybrid simulation that combines kinematic and dynamics. There are several physic engines which can be used in V-REP

In this research, V-REP Pro Edu is used. All the calculation modules are activated, with Vortex as the physics engine for the dynamics module. Vortex is chosen as the physics engine as it offers real-world parameters for many physical properties. Thus, Vortex is both realistic and precise. The scene objects in this research are only the floor and the virtual Zebro. The floor is entirely flat, based on the assumption in this research. Most parts of the control structure are programmed in Matlab, except for the continuous time scheduler. The continuous time scheduler is programmed directly in the V-REP while the SMPL systems and the max-plus states are calculated in Matlab. The max-plus states are then sent to V-REP using remote API interface. The remote API is also used to obtain the position and orientation of the virtual Zebro.

In the simulation, the constant parameters are including θ_t, θ_l, the Jacobian matrices of Zebro, τ_Δ, and τ_f. These constant parameters are defined with value :

$$\theta_t = 0.6\text{rad} \quad \theta_l = -0.6\text{rad}$$
$$\tau_\Delta = 0.1 \quad \tau_f = 0.5$$

Parameters θ_t and θ_l are set within the continuous time scheduler, programmed directly in V-REP. The value of θ_t and θ_l are assigned as above to allow enough variation of τ_g and τ_p while still complying the maximum angular velocity of the leg. The combination between a large range of θ and a small value of τ_g might cause the leg rotate back and forth during the stance period. It happens because the leg needs larger angular velocity than the maximum one to rotate from θ_t to θ_l in τ_g seconds. Here, τ_Δ and τ_f are chosen to be constant as they are presumed to have minimal effect to the turning radius. Therefore, τ_Δ and τ_f are assigned with constant small value.

The changing parameters are including the remaining max-plus parameters, τ_g and τ_p, and the gaits. Both τ_g and τ_p are varied to observe the effect of stance period duration to the turning characteristics. Furthermore, the gaits are also varied to observe the effect of chosen gait to the turning characteristics. The stance time τ_g is varied between 0.4, 0.5, 0.6, 0.7 and 0.8 seconds, while the difference time τ_p is varied between 0.1, 0.2, 0.3, and 0.4 seconds. Regarding the gaits, the tested gaits in the simulation are including

- Tripod 1 {1,4,5},{2,3,6}
- Tripod 2 {2,3,6},{1,4,5}
- Quadruped 1 {1,4},{3,6},{2,5}

- Quadruped 2 {1,6},{2,3},{4,5}

The chosen gaits are a kind of tripod and quadruped gait and their exact opposite gaits. These gaits are chosen so the tripod and quadruped gait can be compared with each other, and with the opposite gaits. The pentapod gait is not considered as it is presumed to be ineffective for turning. The simulation is done in several steps. First, a certain gait, τ_g, and τ_p are chosen. Then, the virtual Zebro is tested with those chosen parameters. The trajectory of the virtual Zebro is saved in the V-REP as a .csv file. The .csv file will be transferred into Matlab, and the corresponding figure is generated. These steps are done for every tested gait, and every kind of τ_g and τ_p. The initial position of the virtual Zebro in the simulation is always (0,0), with initial yaw angle 0 rad.

The results of the simulation are compared to the trajectory predicted by switching kinematic algorithm. To use the switching kinematic algorithm, jacobian matrices are needed. The jacobian matrices of the virtual Zebro are defined as

$$J_1 = \begin{bmatrix} 0 & 0.2 \\ 0.3 & 0.14 \\ 0 & 1 \end{bmatrix} \quad J_2 = \begin{bmatrix} 0 & 0.2 \\ 0.3 & -0.14 \\ 0 & 1 \end{bmatrix} \quad J_3 = \begin{bmatrix} 0 & -0.06 \\ 0.3 & 0.18 \\ 0 & 1 \end{bmatrix}$$

$$J_4 = \begin{bmatrix} 0 & -0.06 \\ 0.3 & -0.18 \\ 0 & 1 \end{bmatrix} \quad J_5 = \begin{bmatrix} 0 & -0.25 \\ 0.3 & 0.14 \\ 0 & 1 \end{bmatrix} \quad J_6 = \begin{bmatrix} 0 & -0.25 \\ 0.3 & -0.14 \\ 0 & 1 \end{bmatrix}$$

5-2-2 Implementation Properties

In addition to the simulation, an implementation is also done with the Zebro. Currently, there are three kinds of Zebro developed in TU Delft. They are PicoZebro, DeciZebro, and KiloZebro. PicoZebro is the smallest one from them, which size is approximately similar with a matchbox. PicoZebro consists entirely of 6 printed circuit boards. DeciZebro is designed to be the mainstream Zebro[26]. It is fully modular and intended for mass production. KiloZebro is the largest one between them. It is designed to be operated on all terrain. Therefore, it is equipped with powerful motors. KiloZebro is also the heaviest one, weighted 15 kilograms (kg). KiloZebro is intended to be the charging station for another kind of Zebro.

During the implementation, DeciZebro is used. DeciZebro is used for the implementation as it has the suitable size, and it also has the suitable system for the trajectory-following SMPL system[26]. The length of DeciZebro is 267.96 millimeters (mm), and its width is 196 mm. It also weighs only 1.771 kg. Compared to the KiloZebro, it is relatively easier to operate DeciZebro during the implementation. In DeciZebro, the motor modules are equipped with a locomotion controller. A Raspberry Pi Zero is used as the locomotion controller. Then, the trajectory-following SMPL system can be programmed in the centralized locomotion controller. The trajectory-following SMPL system is written in C++ programming language, using DevC++ as the programming environment. The trajectory-following SMPL system cannot be programmed in the PicoZebro as it uses an embedded integrated circuit as the centralized locomotion controller.

The implementation is done on a carpeted-floor. The carpeted-floor is used as it provides surfaces which are not slippery. During the implementation, constant τ_g, τ_f, and τ_Δ are used.

The touchdown and lift-off angle of every leg is also set to be constant. The values of them are

$$\theta_t = 1.05\text{rad} \quad \theta_l = -0.7\text{rad}$$
$$\tau_\Delta = 0.1 \quad \tau_f = 0.3 \quad \tau_g = 0.4$$

The value of τ_p is varied between 0.03, 0.06, 0.09, and 0.12 seconds. The gait is also varied between tripod 1, tripod 2, quadruped 1, and quadruped 2. During the implementation, the DeciZebro is first placed at a certain initial position. Then, the DeciZebro is operated with a certain gait and a certain τ_p. The motion of the DeciZebro is captured in a video and the trajectory of DeciZebro is generated from the video. A cell phone application, Motion Shot, is used to capture the video. This application will take a video for eight seconds, and the movement is rendered into a single still image. The image will show the generated trajectory of the DeciZebro. The generated trajectory is used to study the turning characteristic of the DeciZebro. This process is repeated for all considered gaits and τ_p.

The results of the implementation are also compared to the trajectory predicted by switching kinematic algorithm, like the results of the simulation. The jacobian matrices of the DeciZebro are defined as

$$J_1 = \begin{bmatrix} 0 & 0.134 \\ 0.7 & 0.098 \\ 0 & 1 \end{bmatrix} \quad J_2 = \begin{bmatrix} 0 & 0.134 \\ 0.7 & -0.098 \\ 0 & 1 \end{bmatrix} \quad J_3 = \begin{bmatrix} 0 & 0 \\ 0.7 & 0.098 \\ 0 & 1 \end{bmatrix}$$
$$J_4 = \begin{bmatrix} 0 & 0 \\ 0.7 & -0.098 \\ 0 & 1 \end{bmatrix} \quad J_5 = \begin{bmatrix} 0 & -0.134 \\ 0.7 & 0.098 \\ 0 & 1 \end{bmatrix} \quad J_6 = \begin{bmatrix} 0 & -0.134 \\ 0.7 & -0.098 \\ 0 & 1 \end{bmatrix}$$

5-3 Results

5-3-1 Results in Simulation

In this subsection, the results of the simulation are presented and explained. The presented results are chosen to test the hypotheses and to present the importance of assumption A2 in 3. The results of tripod 1 are presented and explained first.

In figure 5-1, the trajectory of virtual Zebro which violated assumption A2 is shown. It can be observed that the virtual Zebro turns to the right direction, even though the intended direction is left. To find the cause of the anomaly, the max-plus states of the corresponding parameters are observed. With $\tau_g = 0.8$ and $\tau_p = 0.1$, the max-plus states for $k = 4$ are

$$t(4) = \begin{bmatrix} 5.8 & 6.5 & 6.4 & 5.8 & 5.8 & 6.5 \end{bmatrix}^T$$
$$l(4) = \begin{bmatrix} 5.3 & 5.9 & 5.9 & 5.2 & 5.3 & 5.9 \end{bmatrix}^T$$

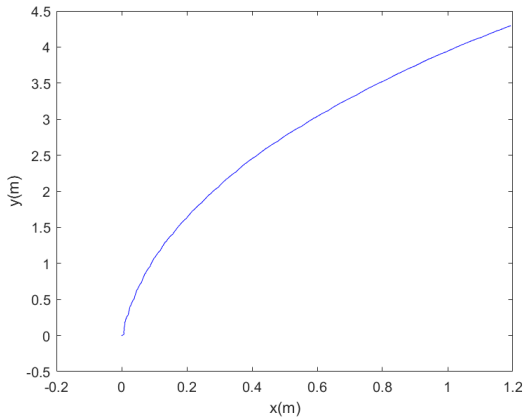


Figure 5-1: The trajectory of virtual zebro with tripod 1 as the chosen gait, $\tau_g = 0.8$, τ_p =0.1, and left as intended direction

From the max-plus states above, all legs in ℓ_1 , {1,4,5} liftoff at the same time instant and touchdown at the different time instant. Thus, the legs in ℓ_1 behave in the opposite way than how they are intended to behave. Different from ℓ_1, the legs in ℓ_2, {2,3,6}, behave as intended. Then, it is interesting to know the true stance time of all legs. The true stance time of all legs are

$$\begin{aligned} \bar{\tau}_{g_1} = 0.9 \quad \bar{\tau}_{g_2} = 0.8 \quad \bar{\tau}_{g_3} = 0.9 \\ \bar{\tau}_{g_4} = 0.8 \quad \bar{\tau}_{g_5} = 0.9 \quad \bar{\tau}_{g_6} = 0.8 \end{aligned}$$

From the true stance time of all legs, the true stance time of the right-sided legs is smaller than the left-sided legs. It means that the right-sided legs are still faster than the left-sided legs, as they should be. The simulation also results in the same cycle time for all legs, which is $\tau_c = 1.4$. It implies that the eigenvalue of the corresponding SMPL system is unique. The value of the stance time of all legs and the eigenvalue show that there is nothing wrong with the flight time and true stance time of all legs.

However, it should be noted that the simulation always starts with the straightforward-only SMPL system with tripod gait, coded directly in the V-REP. This initial condition can be viewed as a disturbance to the intended SMPL system. Therefore, it is a possibility that the steady-state condition is not reached after the initial disturbance, even after the coupling time. Using the corresponding parameters, the assumption A2 is checked for this SMPL system. Both $(\tau_f \otimes \tau_p \otimes \tau_\Delta)^m$ and $(\tau_g \otimes \tau_f \otimes \tau_p)$ are calculated as

$$\begin{aligned} (\tau_f \otimes \tau_p \otimes \tau_\Delta)^m &= 1.4 \\ \tau_g \otimes \tau_f \otimes \tau_p &= 1.4 \end{aligned}$$

Then, it is proven that the assumption A2 is violated. With this particular result, the SMPL system of the virtual Zebro does not fulfill the timing convention of the trajectory-following

SMPL system. The behavior of legs in ℓ_1 points out the violation of timing convention.The violation of timing convention is suspected as the cause of the turning anomaly. The switching kinematic algorithm is used with the same parameters to check if this anomaly can be predicted by the switching kinematic algorithm. However, using the max-plus states, the switching kinematic algorithm generates a very different trajectory compared to the trajectory of the virtual Zebro.

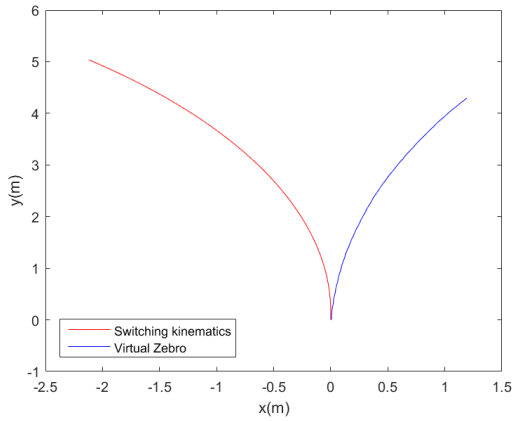


Figure 5-2: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm. Here, tripod 1 is chosen as the gait with left as the intended direction and $\tau_g = 0.8$, τ_p =0.1

Figure 5-2 shows that the switching kinematic algorithm generates trajectory with direction as intended. The difference between the generated trajectory of switching kinematic algorithm and the trajectory of virtual zebro indicates that the anomaly in the turning direction is not related to the kinematic of Zebro. However, this does not imply that the turning anomaly is caused by the violation of timing convention. To further analyze the turning anomaly, the right direction variant of figure 5-2 is presented

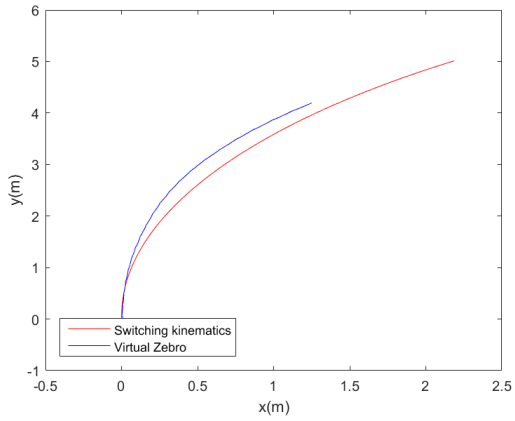
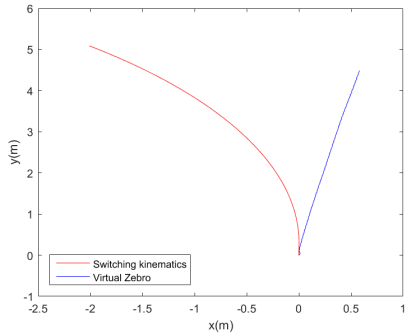
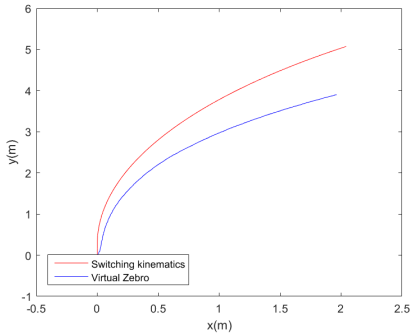


Figure 5-3: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm. Here, tripod 1 is chosen as the gait with right as the intended direction and $\tau_g = 0.4$, τ_p =0.1

Different than in figure 5-2, figure 5-3 does not signify any turning anomaly. The SMPL system of virtual Zebro in 5-3 also violates the timing convention. The violation of timing convention then cannot be concluded as the reason for the turning anomaly. To further find the cause of the turning anomaly, the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm with the same τ_p and varied τ_g are shown below

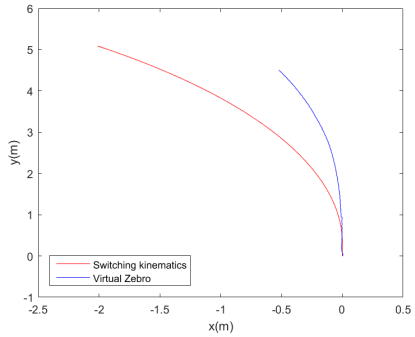


(a) Left direction

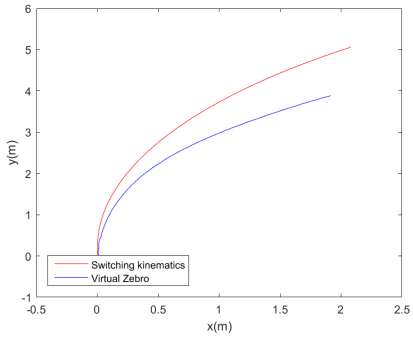


(b) Right direction

Figure 5-4: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, tripod 1 is chosen as the gait with τ_g = 0.5 and τ_p =0.1



(a) Left direction

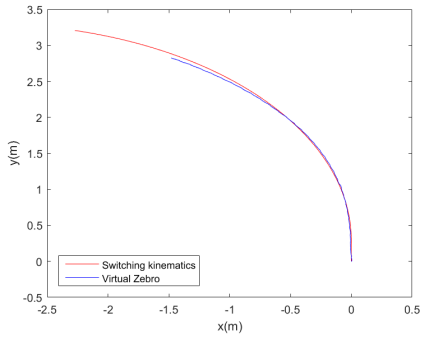


(b) Right direction

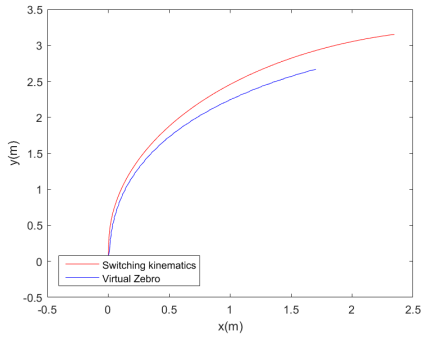
Figure 5-5: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, tripod 1 is chosen as the gait with τ_g = 0.7 and τ_p =0.1

Figure 5-4a to 5-5b shows that turning anomaly happens when $\tau_p = 0.1$ and with left as the intended direction. The switching kinematic function also does not predict the turning motion accurately for those conditions. For that reason, it is interesting to observe the trajectories of virtual Zebro and the generated trajectories of switching kinematic algorithm for τ_p larger than 0.1. Those trajectories can be observed from the figures below

In the figures above, the virtual Zebro turns with the same direction as the intended direction. The generated trajectories of switching kinematic algorithms are also approximately similar with the trajectories of the virtual Zebro. The similarities infer that for the conditions in those figures, switching kinematic algorithms works well. From these observations, it is presumed that there is a minimum value of τ_p so the virtual Zebro can turn with its intended direction.

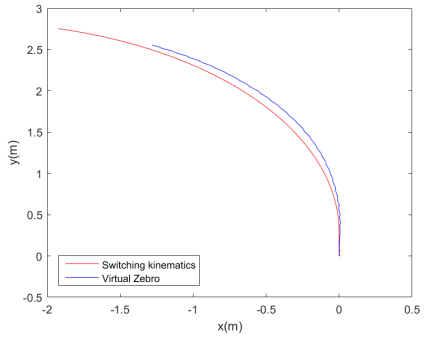


(a) Left direction

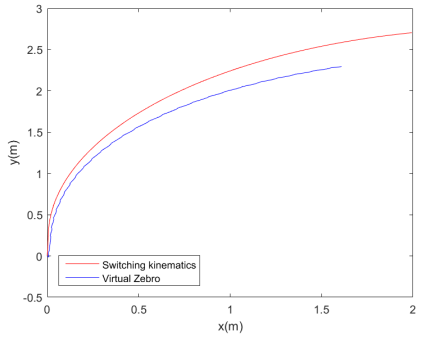


(b) Right direction

Figure 5-6: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, tripod 1 is chosen as the gait with $\tau_g = 0.8$ and τ_p =0.3

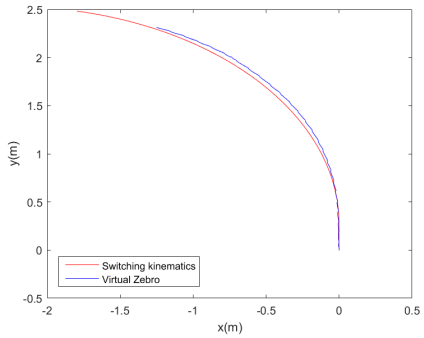


(a) Left direction

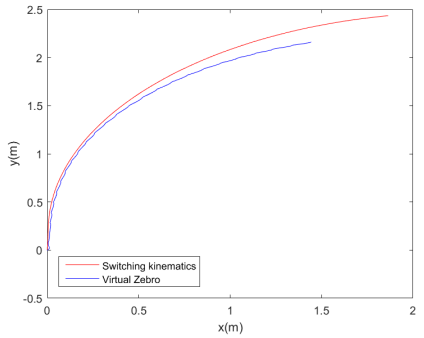


(b) Right direction

Figure 5-7: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, tripod 1 is chosen as the gait with $\tau_g = 0.8$ and τ_p =0.4



(a) Left direction

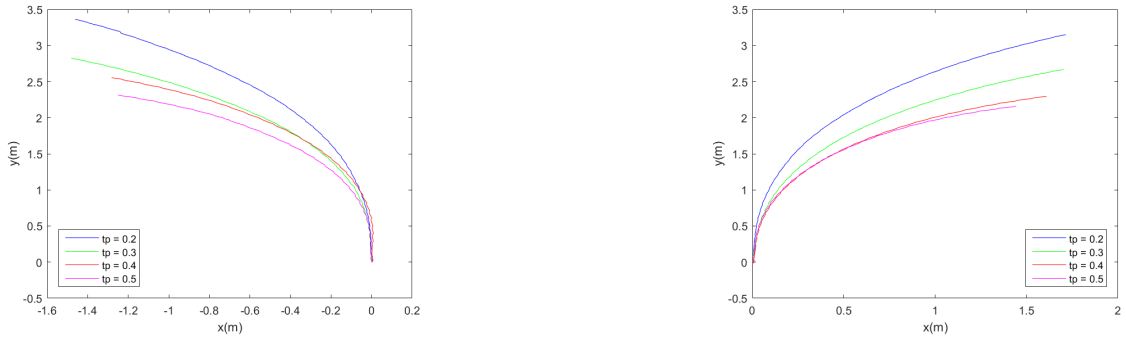


(b) Right direction

Figure 5-8: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, tripod 1 is chosen as the gait with $\tau_g = 0.8$ and τ_p =0.5

This minimum value also indicates the boundary where the switching kinematic algorithm might work.

Furthermore, the virtual Zebro trajectories in figure 5-6a to 5-8b are also compared to understand the effect of τ_p to the turning motion. In figure 5-6a to 5-8b, τ_g is set to be 0.8. Hence, τ_p is the only varied parameter here.



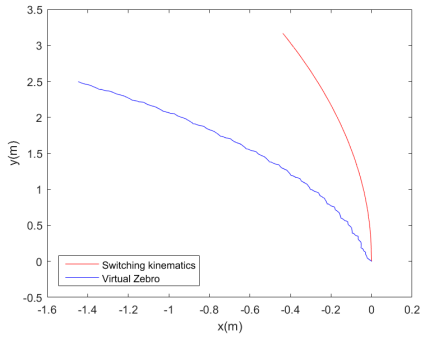
(a) Left direction **(b)** Right direction

Figure 5-9: The comparison between the trajectory of virtual Zebro with $\tau_g = 0.8$, tripod 1 as gait and varied τ_p, for left and right directions

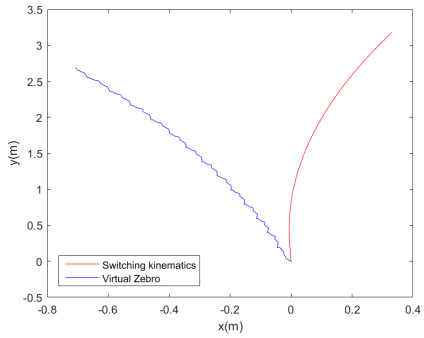
In both figure 5-9a and 5-9b, the increase of τ_p results in smaller turning radius. The increase of τ_p means that the true stance time difference between the outer and the inner side of Zebro is increasing. It is equivalent with increasing the angular velocity difference between the outer and the inner side of differential-drive wheeled robots. In differential-drive wheeled robots, the larger angular velocity difference between outer and inner side also cause smaller turning radius. The results of the simulation, with tripod 2 as the gait, are similar with the results above. Thus, the results are not shown here. The similarities are expected, as those gaits will result in the same sequences of active legs. The only difference is the active legs at the start and the end of the motion. The results of the simulation, with tripod 2 as the gait, can be observed in the Appendix.

Therefore, from the simulation results of virtual Zebro with tripod 1 as the chosen gait, there are three important things that can be implied. First, the value of τ_p has to be over a certain minimum value so the switching kinematic algorithm can predict the turning motion. The minimum value is assumed to be related to the dynamics of the virtual Zebro. Further research is needed to prove this assumption. Second, if the minimum value of τ_p is fulfilled, turning radius can be decreased by increasing the τ_p and keeping the other parameters constant. Then, in designing the supervisory controller, τ_p can be used as the controlled variable to follow a certain trajectory. Third, the simulation shows that the violation of timing convention does not seem to affect the turning motion. It can be observed in figure 5-3 that the virtual Zebro still turns as intended even with violation of timing convention. However, it is important to note that the simulation is on flat terrain. The violation of timing convention might affect the turning motion on the non-flat terrain.

With the results from tripod gait, it is interesting also to check if virtual Zebro with quadruped gait satisfies the hypotheses. The simulation results of virtual Zebro with quadruped 1 as the chosen gait are then presented below

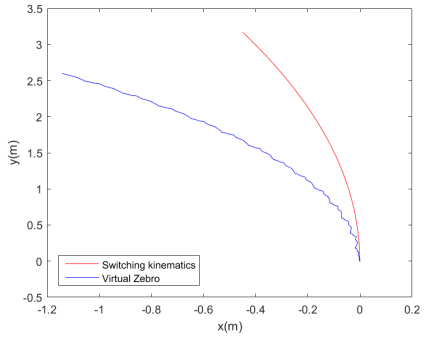


(a) Left direction

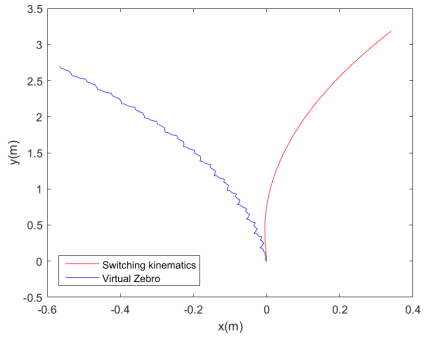


(b) Right direction

Figure 5-10: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, quadruped 1 is chosen as the gait with $\tau_g = 0.5$ and τ_p =0.1

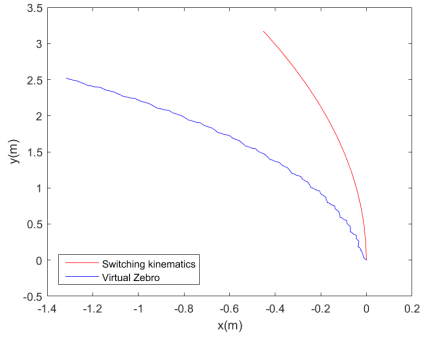


(a) Left direction

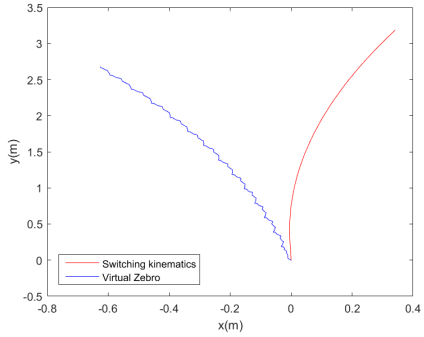


(b) Right direction

Figure 5-11: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, quadruped 1 is chosen as the gait with $\tau_g = 0.6$ and τ_p =0.1



(a) Left direction

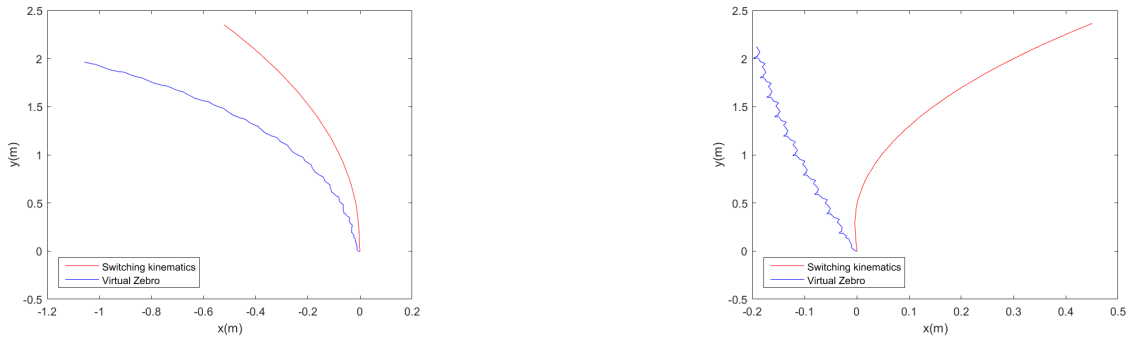


(b) Right direction

Figure 5-12: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, quadruped 1 is chosen as the gait with $\tau_g = 0.7$ and τ_p =0.1

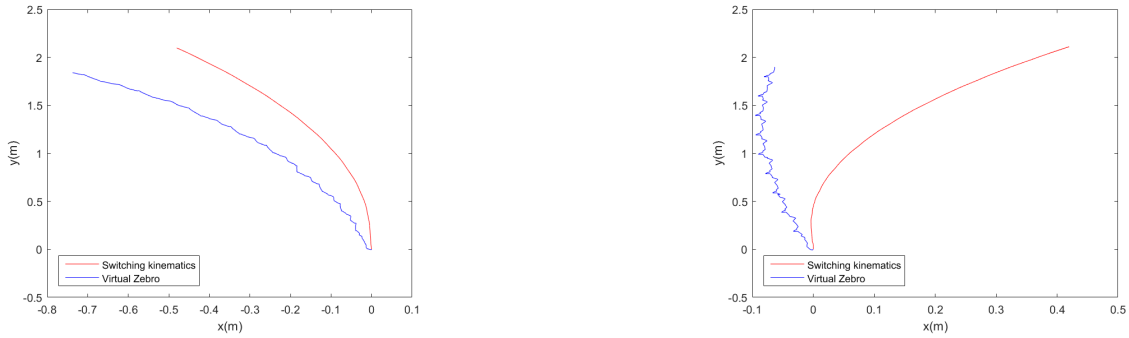
Figure 5-10a to 5-12b are presented as the direct comparison of figure 5-4a to 5-5b. As in figure 5-4a to 5-5b, the parameter τ_p is set as 0.1, with varied τ_g. Figure 5-10a to 5-12b show that the virtual Zebro does not turn to the right when right is chosen as the intended direction. It is the opposite of what occurred in figure 5-4a to 5-5b, where the virtual Zebro does not turn to left as the intended direction. When the left is chosen as the intended direction, virtual Zebro turns to the left. Nevertheless, with left as the intended direction, the trajectories generated by the switching kinematic algorithm do not track the trajectories of virtual Zebro. The radius of generated trajectories is considerably larger than the radius of the trajectories of virtual Zebro.

Then, it is interesting to observe the results of simulation for τ_p larger than 0.1. With tripod 1 as the chosen gait, virtual Zebro turns to the intended direction for τ_p larger than 0.1. The switching kinematic algorithm also predicts the turning motion reliably for that particular condition.



(a) Left direction **(b)** Right direction

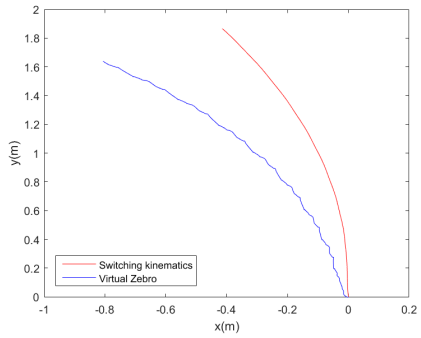
Figure 5-13: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, quadruped 1 is chosen as the gait with $\tau_g = 0.8$ and τ_p =0.3



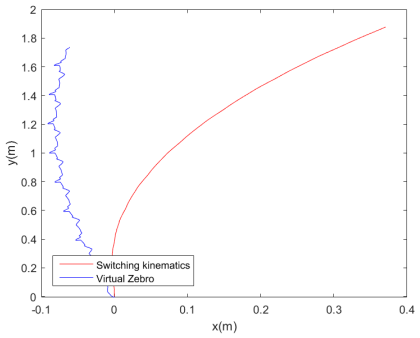
(a) Left direction **(b)** Right direction

Figure 5-14: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, quadruped 1 is chosen as the gait with $\tau_g = 0.8$ and τ_p =0.4

The results of simulation for τ_p larger than 0.1, and τ_g equals to 0.8 are presented in figure 5-13a to 5-15b. From observation of those figures, the turning motion of virtual Zebro is not



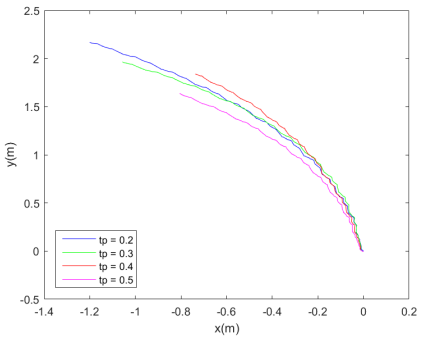
(a) Left direction



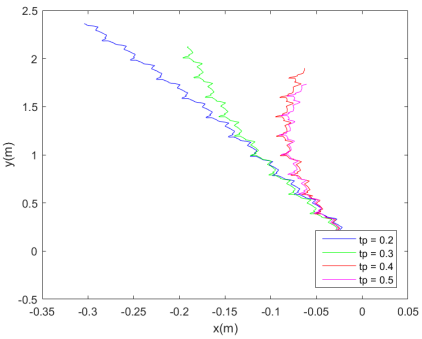
(b) Right direction

Figure 5-15: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, quadruped 1 is chosen as the gait with $\tau_g = 0.8$ and τ_p =0.5

smooth if quadruped 1 is chosen as the gait. The virtual Zebro always meanders while it is turning to the intended direction. The meandering is even more obvious when virtual Zebro is turning to the right direction. Similar with the results in figure 5-10a to 5-12b, when right is the intended direction, virtual Zebro somehow does not turn to the right instantly. If left is the intended direction, virtual Zebro can turn to the left immediately. Still, the switching kinematic algorithm does not predict the turning motion well for both directions. Visually, the prediction error of switching kinematic algorithm is larger when quadruped 1 is utilized.



(a) Left direction



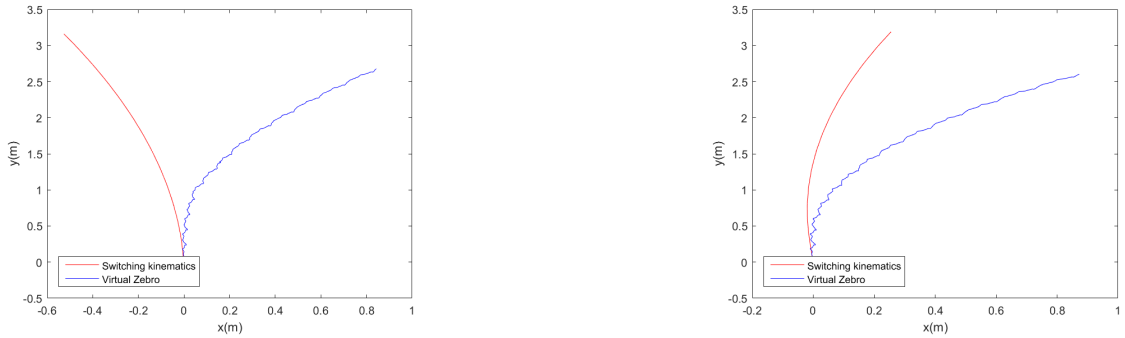
(b) Right direction

Figure 5-16: The comparison between the trajectory of virtual Zebro with $\tau_g = 0.8$, quadruped 1 as the gait, and varied τ_p, for left and right directions

With the results in figure 5-10a to 5-15b, the second hypotheses is not accomplished. To look whether the first hypotheses is accomplished, figure 5-16a and 5-16b are shown above. In general, similar as when virtual Zebro uses tripod 1 as the gait, the increase of τ_p results in smaller turning radius of virtual Zebro. The only exception occurs for τ_p equals to 0.4 with left as the turning direction, as can be seen in figure 5-16a. Then, using quadruped 1 as the gait, τ_p still can be used as the control variable so virtual Zebro can turn with a certain radius.

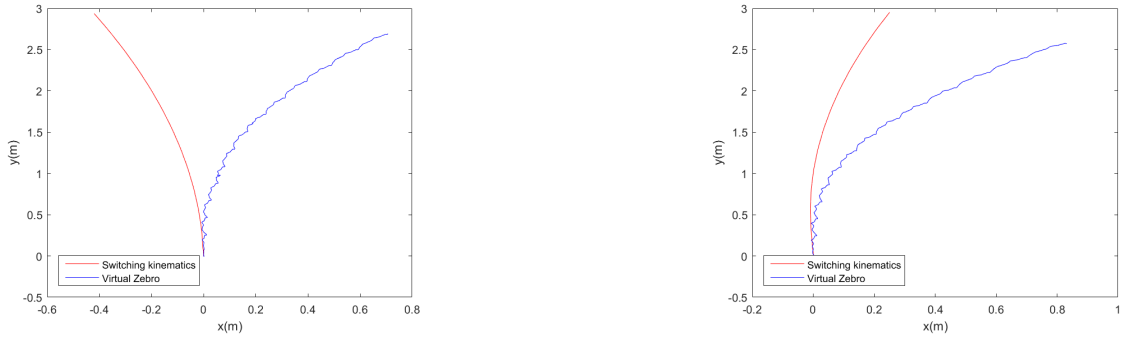
The results of the simulation with quadruped 1 as the gait are compared with the results of

the simulation with quadruped 2 as the gait. Quadruped 2 is chosen as the sequence of the legs in the stance period is the exact opposite of quadruped 1. Then, quadruped 2 can be regarded as the mirror gait of quadruped 1. The simulation results of quadruped 2 as the chosen gait, with τ_p equals to 0.1, are presented first.



(a) Left direction **(b)** Right direction

Figure 5-17: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, quadruped 2 is chosen as the gait with τ_g = 0.5 and τ_p =0.1

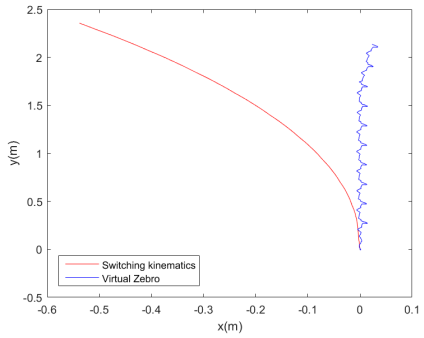


(a) Left direction **(b)** Right direction

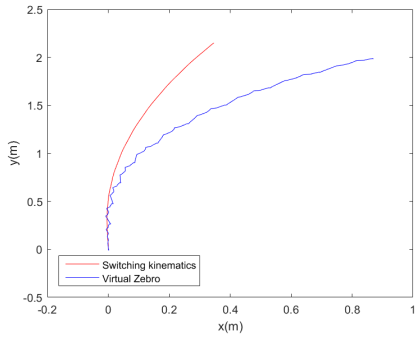
Figure 5-18: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, quadruped 2 is chosen as the gait with τ_g = 0.7 and τ_p =0.1

The result in the figure 5-17a to 5-18b show that the Zebro is also meandering when it is using quadruped 2 as the gait. The main difference is, when quadruped 2 is used, virtual Zebro does not turn to the left when it is the intended direction. It is the opposite of what happen for the simulation with quadruped 1 as the chosen gait. When right is chosen as the intended direction, the virtual Zebro turns to the right. However, still, the trajectory generated by the switching kinematic algorithm is not similar to the trajectory of virtual Zebro for any direction. When the virtual Zebro turns to the right as the intended direction, the radius of generated trajectories is also considerably larger than the radius of the trajectories of virtual Zebro.

For τ_p larger than 0.1 and τ_g equals to 0.8, the results of simulation are presented in figure 5-19a to 5-21b. The results reiterate the difficulty of virtual Zebro to turn left if quadruped 2 is

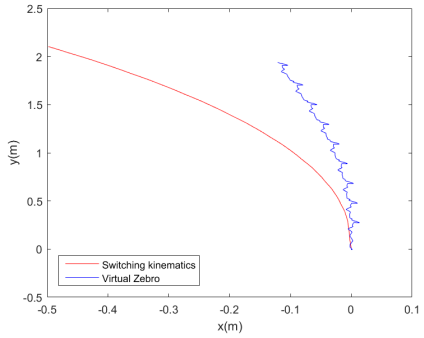


(a) Left direction

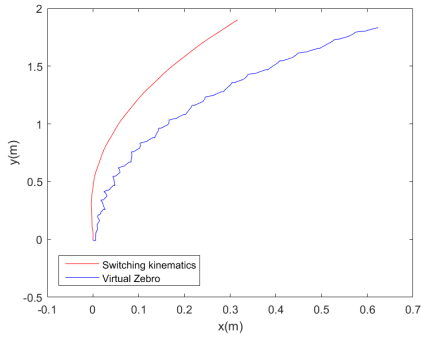


(b) Right direction

Figure 5-19: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, quadruped 2 is chosen as the gait with $\tau_g = 0.8$ and τ_p =0.3

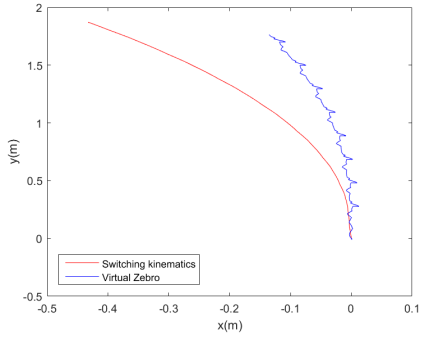


(a) Left direction

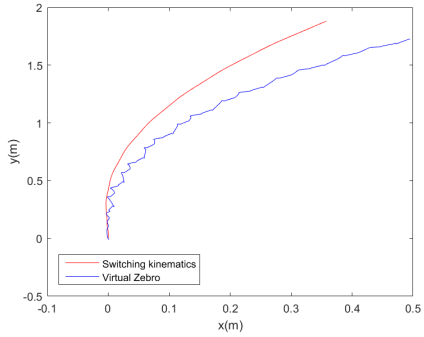


(b) Right direction

Figure 5-20: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, quadruped 2 is chosen as the gait with $\tau_g = 0.8$ and τ_p =0.4



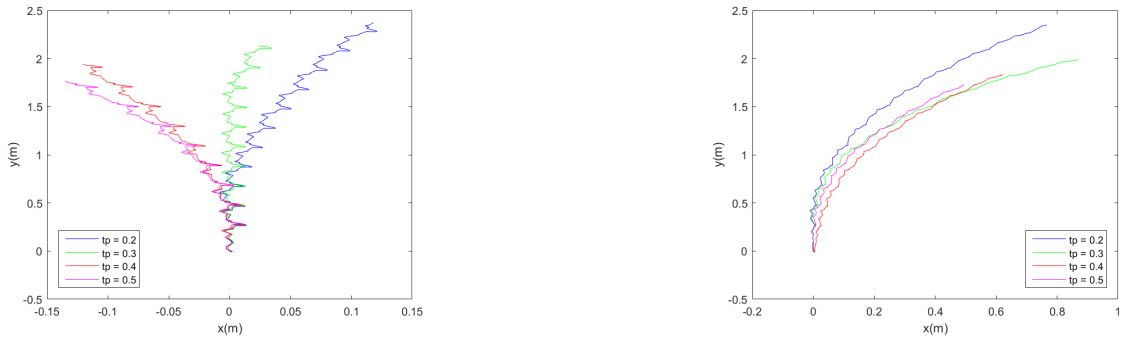
(a) Left direction



(b) Right direction

Figure 5-21: The comparison between the trajectory of virtual Zebro and the generated trajectory of switching kinematic algorithm for left and right directions. Here, quadruped 2 is chosen as the gait with $\tau_g = 0.8$ and τ_p =0.5

used. As the τ_p goes larger, the virtual Zebro turns left when it is the intended direction. The meandering is also still obvious in figure 5-19a to 5-21b. The switching kinematic algorithm does not predict the motion accurately as well. Then, the second hypothesis is also not fulfilled for quadruped 2.

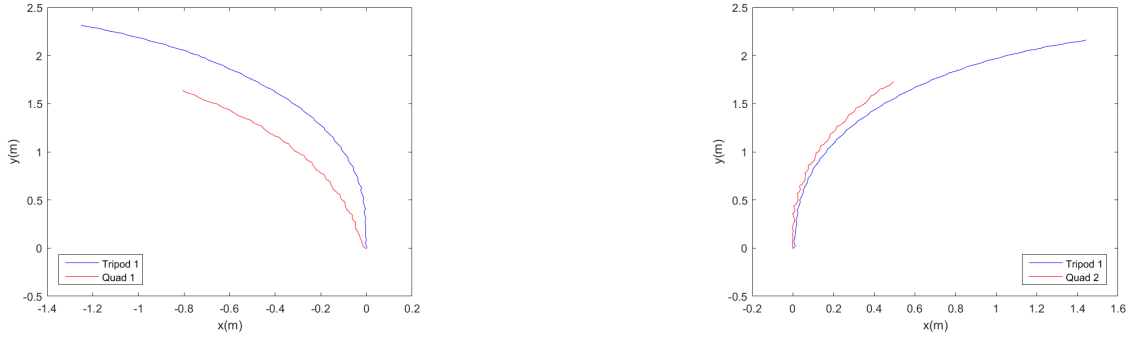


(a) Left direction **(b)** Right direction

Figure 5-22: The comparison between the trajectory of virtual Zebro with $\tau_g = 0.8$, quadruped 2 as the gait, and varied τ_p, for left and right directions

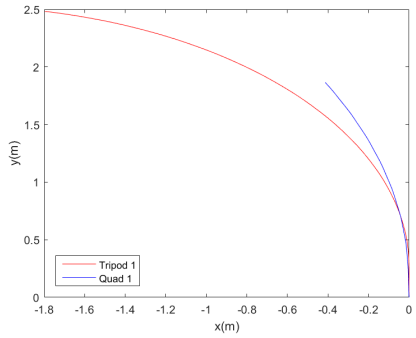
The results in figure 5-19a to 5-21b are compiled into figure 5-22a and 5-22b, to observe the effect of τ_p. In the figure 5-22a, it can be clearly observed that the virtual Zebro turns left with smaller radius as the τ_p goes larger. However, the same observation cannot be seen in the figure 5-22b. The virtual Zebro produces the largest radius when τ_p equals to 0.2, but the trajectories of virtual Zebro with τ_p equals to 0.3, 0.4, and 0.5 have similar radius. Therefore, the second hypotheses is not fully satisfied in this case.

Finally, the results of the simulation of each kind of gait are compared. The comparison is done for τ_g equals to 0.8 and τ_p equals to 0.5. For the left direction, the results of the simulation with tripod 1 and quadruped 1 as the gait are compared. For the right direction, the results of the simulation with tripod 1 and quadruped 2 as the gait are compared. The comparison is done in this way as the virtual Zebro tends to turn to a specific direction if quadruped is used as the gait. The results are

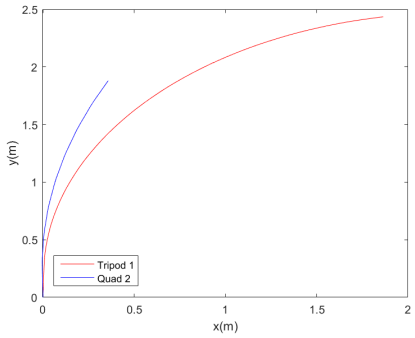


(a) Left direction **(b)** Right direction

Figure 5-23: The comparison between the trajectory of virtual Zebro with tripod and quadruped as the gait, for left and right directions. Here, $\tau_g = 0.8$ and τ_p =0.5



(a) Left direction



(b) Right direction

Figure 5-24: The comparison between the generated trajectory of switching kinematic algorithm with tripod and quadruped as the gait, for left and right directions. Here, τ_g = 0.8 and τ_p =0.5

From the figure 5-23a and 5-23b, it can be observed that the turning radius of virtual Zebro with quadruped gait is smaller than with tripod gait for the left direction. On the other hand, the turning radius of the virtual Zebro with tripod gait is smaller than with quadruped gait for the right direction. Therefore, with these results, it is interesting to observe the corresponding generated trajectory of the figure 5-23a and 5-23b. The results are shown in figure 5-24a and 5-24b. Figure 5-24a and 5-24b show that the turning radius of virtual Zebro with tripod gait should always be smaller than with quadruped gait. Thus, the results in figure 5-23a and 5-23b are not as expected.

There is a presumed reason why the switching kinematic algorithm does not predict the results without an error. The reason is related to the assumption which is used in this report. In this report, the slipping between the legs and the ground is not considered. However, the slipping might occurs during the simulation as the dynamic module is turned on in the V-REP. Therefore, the dynamics between the legs and the ground is considered in the simulation as well. The skidding of the legs might also be considered in the V-REP as the consequence. As the dynamics affect the motion of the virtual Zebro, kinematics alone does not predict the complete motion of the virtual Zebro. The slipping and skidding of the virtual Zebro are not incorporated in the switching kinematic algorithm. Thus, the switching kinematic algorithm cannot predict the displacement caused by either slipping or skidding.

When the virtual Zebro does not turn as it is intended, it is presumed to be caused by the tilting of virtual Zebro. The tilting is caused by the position of the legs which are on the stance period. When the quadruped gait is utilized, the virtual Zebro tilts routinely. In every kind of quadruped gait, there are moments when the front leg of a particular side is on the stance period while the two others are. As the middle and the back leg move on the ground, the front leg does not hold the virtual Zebro body. Then, the virtual Zebro tilts. As an example, the sequences of the active legs for quadruped 1 with left as the direction are

$$\{2,5,3,6,1\},\{2,5,3,6,1,4\},\{3,6,1,4\}$$
$$\{3,6,1,4,5\},\{3,6,1,4,2,5\},\{1,4,2,5\}$$
$$\{1,4,2,5,3\},\{1,4,2,5,3,6\},\{2,5,3,6\}$$

From the sequence of the active legs above, it can be observed that the tilting will happen on the left side of the virtual Zebro. The tilting will happen when the active legs are $\{2, 5, 3, 6\}$. For right as the intended direction, it can be presumed that the virtual Zebro does not always turn right because of the tilting on the left side of the virtual Zebro. The tilting on the left side might cause the decrease of linear velocity on that side. The torque of both legs are also transformed into vertical force, instead of only horizontal force. Therefore, the linear velocity of the left side of virtual Zebro might not be larger than the linear velocity of the right side when the active legs are $\{2, 5, 3, 6\}$. The meandering then happens because of that. The tilting on the left side does not affect much the left turning of virtual Zebro as the linear velocity of the right side is still larger than the left side. Still, the tilting causes meandering of virtual Zebro when the virtual Zebro is trying to turn left. The same principle can answer why the virtual Zebro with quadruped 2 as the gait does not always turn left, for left as the intended direction. The tilting might also answer why the results in figure 5-23a and 5-23b are different with figure 5-24a and 5-24b. With the tilting, a certain lower boundary of τ_p has to be exceeded to create linear velocity difference between each side. From the observation of the results above, the boundary is τ_p equals to 0.4 for virtual Zebro with quadruped gait.

The tilting might also happen when tripod 1 is utilized as the gait, but obviously the effect is not as worse as when quadruped is utilized as the gait. The tilting is less possible to happen when tripod is used as the gait, as the active legs of each side are either the front and back leg, only the middle leg, or all legs. The tilting is more possible to happen the active legs of a side are the middle and the back leg. The reason is that the front part of that side is not held by a leg, while the middle and the back leg give force to the front part. However, the tilting might also answer why a certain lower boundary of τ_p has to be exceeded to create linear velocity difference between each side. From the observation of the results above, the boundary is τ_p equals to 0.2 for virtual Zebro with tripod gait.

The tilting illustrates the dynamic nature of Zebro, where force has a significant effect on the motion of Zebro. Because of the switching active legs and its morphology, the force which occurs on Zebro also cause the vertical displacement. The switching kinematic algorithm is not able to predict this, as it does not consider any vertical displacement to happen. Therefore, the switching kinematic algorithm is less accurate if the tilting is more possible to happen.

5-3-2 Results in Implementation

After several trials on the DeciZebro, there is still no conclusion that can be derived as this report is finished. The results, with tripod 1 as the utilized gait, show that the DeciZebro can turn with the trajectory-following SMPL system. However, the results do not show any pattern that can be observed. The results are

From the results, it can be observed that the DeciZebro always turns right even when the left is chosen as the intended direction. The results do not show clearly the relation between τ_p and the turning radius. With the results above, both hypotheses cannot be proven yet. There are several presumed reasons why DeciZebro does not turn as expected. The first presumed reason is that the touchdown and lift-off angle of every leg is not the same. If the touchdown and lift-off angle of every leg is not the same, the trajectory-following SMPL might not generate the intended turning in DeciZebro. It is obvious, as the kinematics of DeciZebro is directly affected by the touchdown and lift-off angle of every leg. After the

(a) Left direction

(b) Right direction

Figure 5-25: The trajectory of DeciZebro for left and right directions. Here, tripod 1 is chosen as the gait with $\tau_g = 0.4$ and τ_p =0.03

(a) Left direction

(b) Right direction

Figure 5-26: The trajectory of DeciZebro for left and right directions. Here, tripod 1 is chosen as the gait with $\tau_g = 0.4$ and τ_p =0.06

(a) Left direction

(b) Right direction

Figure 5-27: The trajectory of DeciZebro for left and right directions. Here, tripod 1 is chosen as the gait with $\tau_g = 0.4$ and τ_p =0.09

results above are obtained, it is found that the angle of the middle left leg is smaller by five degrees compared to the angle of the middle right leg. The second presumed reason is that the interaction between the legs and the carpet causes a undesired effect. However, further

research is needed to explain this effect. The third presumed reason is that the chosen τ_p is not enough to induce turning to the left side. It is already shown in the simulation that the virtual Zebro has a tendency to turn right even when the intended direction is left. During the simulation, the virtual Zebro will turn left if a certain lower boundary of τ_p is exceeded. The same case might happen with the implementation.

Because of the restriction of time, there is no further implementation done as this report is finished. As the results are not sufficient, the results are not compared with the trajectory generated by the switching kinematic algorithm. From the experience of the implementation, there are several acts which have to be done for the next implementation. The first act is designing a more precise calibration of every leg. The calibration is already done in this implementation, but the angle error has to be measured manually. From the results above, it is presumed that the angle of the legs might not be similar during this implementation. For the next implementation, the angle error of every leg should be measured and corrected automatically during the calibration process. The second act is to do the implementation on several kinds of surface. By doing the implementation on several kinds of surface, the results obtained on each surface can be compared with each other. Then, it can be concluded whether the turning really happens because of the trajectory-following SMPL. The third act is to do the implementation with a larger range of τ_p. With larger range of τ_p, the effect of changing τ_p can really be observed. It also can be checked if the DeciZebro can turn to the intended direction if a certain lower boundary of τ_p is exceeded.

5-4 Summary

In this chapter, the results of the simulation and implementation are obtained. The results show that by increasing the value of τ_p, the turning radius can be decreased in general. The results also show that the switching kinematic algorithm can predict the trajectory of virtual Zebro if the value of τ_p if several conditions are fulfilled. However, the switching kinematic algorithm can only predict the trajectory accurately if there is no significant vertical motion happens. It is shown that if quadruped is utilized as the gait, vertical motion happens and the turning might not happen as intended. However, the results of the implementation are not conclusive yet. A further implementation with improvements is needed.

Chapter 6

Conclusion and Future Works

6-1 Conclusion

This report is mainly concerned with the turning motion of Zebro by means of trajectory-following switching max-plus linear (SMPL) system. The discussion in this report is intended to explore how the turning motion of Zebro can be initiated and predicted, with the trajectory-following SMPL system and the kinematics of Zebro as the foundation. With the discussion in this report, the research questions are answered

- **How can a Zebro turn by means of SMPL system?**
  A Zebro can turn by utilizing the trajectory-following SMPL system. With the trajectory-following SMPL system, the Zebro turns in a similar way such as differential-drive mobile robot. The trajectory-following SMPL system induces different touchdown time instants between the right-sided and left -sided legs in the same sequence. On the other hand, the lift-off time instants between all legs in the same sequence are the same. The stance period of one side then is shorter than another side. Therefore, the angular velocity of right-sided legs and left-sided legs during the stance period are different. The linear velocity of the right side and left side of Zebro are consequently different.

- **How can the turning motion of Zebro be predicted?**
  The turning motion of Zebro can be predicted by using the switching kinematics algorithm in Chapter 4. As a consequence of how it moves, the legs of Zebro which are in stance period always change over time. The switching kinematics algorithm detects which legs are in stance period at a particular time, and constructs the kinematics relation of Zebro with the legs which are in stance period. The states of trajectory-following SMPL system are used to detect the legs which are in stance period and to calculate the change of angle in those legs. The kinematics relation is formed based on the method in **??**. Comparing to the results of simulation, the switching kinematic algorithm can predict the motion of Zebro if the tripod gait is utilized with a certain value of τ_p as the lower boundary. The switching kinematic algorithm needs the lower boundary as a

certain linear velocity difference between each side is needed to minimize the effect of tilting.

- **How is the turning performance of the Zebro?** In the simulation, the Zebro is shown to be able to turn by the trajectory-following SMPL. In general, the parameter τ_p affects the radius of the turning. With larger τ_p, the radius of the turning is smaller. However, the Zebro does not always turn to the intended direction. The turning of Zebro is also not smooth if quadruped is utilized as the gait. From the results of the simulation, it can be concluded that the trajectory-following SMPL system can be applied with respect to several factors as the consideration. Those factors are including the choice of max-plus parameters and the choice of the gait. Still, the results of the implementation are not conclusive yet. A further implementation is needed to verify the expected turning motion of using the trajectory-following SMPL in the Zebro.

It can be concluded that all research questions are answered. In answering the research questions, this report brings three original contributions. The first and second contribution are related to the first research question, while the third contribution is related to the second research question. The contributions can be summarized as :

1. **Trajectory-following SMPL system**
   The trajectory-following SMPL system can be constructed as

$$\begin{aligned} x(k) &= \left[\begin{array}{c|c} \mathcal{E} & \tau_f \otimes M \\ \hline R & \mathcal{E} \end{array}\right] \otimes x(k) \oplus \left[\begin{array}{c|c} E & \mathcal{E} \\ \hline (\tau_g \otimes N) \oplus T & E \end{array}\right] \otimes x(k-1) \\ &= G_0 \otimes x(k) \oplus G_1 \otimes x(k-1) \end{aligned}$$

   where the structure of the matrices depend on the chosen gait and the direction of Zebro. The matrices R and T are formed based on the chosen gait and the direction of Zebro, while the matrices N and M are formed based only on the direction of Zebro. The trajectory-following SMPL enables forward motion to all directions. If the Zebro turns, the trajectory-following SMPL system ensures different touchdown time instants between the right-sided and left-sided legs in the same sequence. In turning, the stance period of the inner legs are longer by τ_p compared to the stance period of the outer legs. The swing period of the inner legs are shorter by τ_p compared to the swing period of the outer legs. The introduction of parameter τ_p causes Zebro to behave like differential-drive mobile robot, while keeping the cycle time of all legs the same.

2. **Eigenstructure of trajectory-following SMPL system**
   The eigenvalue and the eigenvector of the trajectory-following SMPL system are defined as

$$\begin{aligned} \lambda &:= (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes m} \\ \forall j \in \{1, ..., m\}; \forall q^i \in l_j^i; \forall q^o \in l_j^o &: \\ [v]_q^i &:= \tau_f \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \\ [v]_q^o &:= (\tau_f \otimes \tau_p) \otimes (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \\ [v]_{q+n}^{i,o} &:= (\tau_f \otimes \tau_p \otimes \tau_\Delta)^{\otimes j-1} \end{aligned}$$

The eigenvalue and the eigenvector of the trajectory-following SMPL system can be only defined as above if assumptions A1 and A2 in Lemma 3-4.1 are fulfilled. The eigenvalue represents the total cycle time for a leg on the Zebro, while the eigenvector represents the steady-state condition of the max-plus states. The chosen gait and the direction of the Zebro can be inferred from the eigenvector.

3. **Switching kinematic model of Zebro**
   The switching kinematic model of Zebro is adapted from the kinematic model of wheeled mobile robot [5]. The switching kinematic model is able to construct the kinematic model based on the legs which are on the stance period. The kinematic model itself is defined as

$$\begin{bmatrix} {}^{\bar{R}}\Delta x_{\bar{R}} \\ {}^{\bar{R}}\Delta y_{\bar{R}} \\ {}^{\bar{R}}\Delta \alpha_{\bar{R}} \\ \Delta\alpha_{1z} \\ . \\ . \\ \Delta\alpha_{Nz} \end{bmatrix} = (A_n^T A_n)^{-1} A_n^T B_s \begin{bmatrix} \Delta\theta_{1x} \\ \Delta\theta_{2x} \\ . \\ . \\ \Delta_{Nx} \end{bmatrix}$$

   The dimension and elements of the matrices and vectors in the kinematic model depend on which legs are on the stance period, and the number of those legs. The inputs of the kinematic model of Zebro, the change of motor angle $\Delta\theta_{ix}$, are obtained from the max-plus states and phase reference generator. The switching kinematic algorithm is limited by the value of τ_p.

6-2 Future Works

Regarding to the research in this report, there are several topics which can be explored. They are

- An implementation with improvements
  As already explained, there are several improvements that should be done for the next implementation. The improvements are including an improved calibration process, a larger range of tested parameters, and variations on the surface used.

- The new legs for Zebro
  In [27], a new design of legs for Zebro is already developed. The new legs guarantee a linear relation between the change of angle and the displacement of each leg. Therefore, the new design might be useful to develop trajectory-tracking Zebro.

- Coupling time of trajectory-following SMPL system
  In this report, the coupling time of trajectory-following SMPL system is expected to be two, with one as the cyclicity. After several numerical simulations, the expectation seems to be true if several assumptions are satisfied. However, this hypothesis has to be proven in the max-plus algebraic sense. With the knowledge of coupling time, the parameters can be set so the steady-state condition can be reached as soon as possible.

- Gait, variable, and direction transition of trajectory-following SMPL system
  From the results in this report, it is clear that different gait has different turning characteristic. The value of the variables also affect the turning characteristic. Therefore, when Zebro is a following a trajectory, the change of gait or variable can be utilized. The change of gait or variables cannot occur instantly, as a transition period will happen. This report has not discussed the transition period yet. It is also important to note that Zebro might need to change direction. The transition period is also exist during the change of direction. Thus, an effective transition is very important to be applied.

- Supervisory control and gait generator of the trajectory-following SMPL system
  Last, a supervisory controller and a gait generator is needed to have a fully autonomous Zebro with the trajectory-following SMPL. The function of the supervisory controller is to decide the linear velocity of the right and left side of the Zebro. The decision is based on the trajectory which needs to be followed, or the target position. The gait generator will translate the desired linear velocity of each side to the right gait and max-plus parameters. The supervisory controller might be equipped with a trajectory planning algorithm, to decide where the Zebro should go.

Appendix A

Appendix:MATLAB code and extra figures

A-1 MATLAB code of the simulation of trajectory-following SMPL system (Developed from the MATLAB code of dr.Gabriel Lopes)

```
%% Initialize VREP Simulator:
clear all;
clc;
InitSimulator;


%% Global Variables for Communication :

global SyncInfo;

SyncInfo = [];
%% Add Required Paths
addpath('MaxPlus');
addpath('Zebro');

%% Simulation Loop:

SimTime = 0;
RMessage='';
JMessage='';
JX=zeros(1,12);

% chg = input('What gait do you want?');
% dir = input('What direction do you want?');
% tg = input('What tg do you want?');
% tp = input('What tp do you want?');
```

```
27
28 count = 1;
29
30 %% Simulation Code
31
32 while (SimTime < 30)
33
34 % Input to SMPL : gait and direction
35 chg =5;
36 dir =2; %2 for right direction, 1 for left direction
37
38 % Input to SMPL : tg and tp
39 tg = 0.8;%0.6
40 tp = 0.5;%0.2
41
42 % Reading States from Zebros:
43 [JPosition JOrientation JState JRotationMatrix] = Zebro_Read(clientID,
       vrep, Zebro_Juliet_Handle,'Zebro_Juliet#0');
44
45 % Run controller for Zebro Julliet
46 [JMode, JDirection, JTurn, Jtheta_T, Jtheta_L, JA, JReference, JMessage]
       = Zebro_Juliet(SimTime, chg, dir, tg, tp ,RMessage);
47
48 % Send information to Zebro
49 % (State x(k) and Rotation Matrix)
50 [JX]=Zebro_Write(clientID, vrep, 'Zebro_Juliet#0',JMode, JDirection,
       JTurn, Jtheta_T, Jtheta_L, JA, JReference);
51
52 % Collecting the states
53 JStates(:,count) = JX;
54
55 % Record the simuation time
56 [SimTime] = vrep.simxGetLastCmdTime(clientID);
57 SimTime = SimTime*1e-3;
58
59 [dTime(count)] = vrep.simxGetLastCmdTime(clientID);
60 dTime(count) =dTime(count)*1e-3;
61
62
63 r_position(count,:) = JPosition;
64 r_orientation(count,:) = JOrientation; %Value in radian! Take the second
       column
65
66 count = count + 1;
67
68 end
69
70 StopSimulator(vrep, clientID);
71
72 %% Calculating Linear and Angular Velocity data
73
74 ro_deg = rad2deg(r_orientation); % orientation in degree
75
```

```
%Calculating the linear and angular velocity
for i = 1:1:count-2
    for j =1:3
r_lv(i,j) = (abs(r_position(i+1,j)-r_position(i,j)));
r_av(i,j) = (abs(r_orientation(i+1,j)-r_orientation(i,j)));
    end
end

for p = 1:1:length(r_lv)
v_linear(p) = sqrt(r_lv(p,1)^2 +r_lv(p,2)^2);
end

% Calculating the mean of linear and angular velocity
lv_average = mean(r_lv); % Take only 1st and 2nd column, x and y
av_average = mean(r_av); % Take only 2nd column, alpha
vlin_average = mean(v_linear);

[radius,a,b] = radiusofcurv(r_position(:,1),r_position(:,2))

%% Saving simulation result
simresult.position = r_position;
simresult.orientation = r_orientation;
simresult.Vc = r_lv;
simresult.AVc = r_av;
simresult.mVc = lv_average;
simreuslt.mAVc = av_average;
simresult.mVlin = vlin_average;
simresult.rads = radius;
simresult.a = a;
simresult.b = b;
simresult.X =JStates;

 save('simresult_quad3_30.mat','-struct','simresult');
```

```
%% Establishing connection between MATLAB and VREP:
addpath('RemoteApi');

disp('Program Started');

vrep = remApi('remoteApi'); % Create an object named vrep for remoteApi

vrep.simxFinish(-1); % Just to make sure that all connections with VREP
    are
% closed

clientID = vrep.simxStart('127.0.0.1',19997,true, true, 5000,5); % Start
    the
% connection with debug and synchronous mode set to true.


% Check for Valid Connection
if (clientID > -1)

```

```
18      disp('Connected to remote API server');
19
20  else
21
22      disp('Failed connecting to remote API server');
23
24      close;
25  end
26
27
28  vrep.simxSynchronous(clientID, false); %Set-up a-synchronous
        communication on
29  % the client side
30
31  vrep.simxStartSimulation(clientID, vrep.simx_opmode_oneshot_wait); %
        Start
32  % the simulation
33
34
35
36                          %%%%%%%%%% Retrieve all Joint Handles:
                               %%%%%%%%%%%%
37
38  %%%%% Zebro-Juliet Joint Handles %%%%%
39
40  [res Zebro_Juliet_Handle(1)] = vrep.simxGetObjectHandle(clientID, '
        Zebro_Juliet#0', vrep.simx_opmode_oneshot_wait);
41  [res Zebro_Juliet_Handle(2)] = vrep.simxGetObjectHandle(clientID, 'Joint1
        #0', vrep.simx_opmode_oneshot_wait);
42  [res Zebro_Juliet_Handle(3)] = vrep.simxGetObjectHandle(clientID, 'Joint2
        #0', vrep.simx_opmode_oneshot_wait);
43  [res Zebro_Juliet_Handle(4)] = vrep.simxGetObjectHandle(clientID, 'Joint3
        #0', vrep.simx_opmode_oneshot_wait);
44  [res Zebro_Juliet_Handle(5)] = vrep.simxGetObjectHandle(clientID, 'Joint4
        #0', vrep.simx_opmode_oneshot_wait);
45  [res Zebro_Juliet_Handle(6)] = vrep.simxGetObjectHandle(clientID, 'Joint5
        #0', vrep.simx_opmode_oneshot_wait);
46  [res Zebro_Juliet_Handle(7)] = vrep.simxGetObjectHandle(clientID, 'Joint6
        #0', vrep.simx_opmode_oneshot_wait);
47
48  %%%%% Zebro-Romeo Joint Handles %%%%%
49
50  [res Zebro_Romeo_Handle(1)] = vrep.simxGetObjectHandle(clientID, '
        Zebro_Romeo', vrep.simx_opmode_oneshot_wait);
51  [res Zebro_Romeo_Handle(2)] = vrep.simxGetObjectHandle(clientID, 'Joint1'
        , vrep.simx_opmode_oneshot_wait);
52  [res Zebro_Romeo_Handle(3)] = vrep.simxGetObjectHandle(clientID, 'Joint2'
        , vrep.simx_opmode_oneshot_wait);
53  [res Zebro_Romeo_Handle(4)] = vrep.simxGetObjectHandle(clientID, 'Joint3'
        , vrep.simx_opmode_oneshot_wait);
54  [res Zebro_Romeo_Handle(5)] = vrep.simxGetObjectHandle(clientID, 'Joint4'
        , vrep.simx_opmode_oneshot_wait);
```

```
55 [res Zebro_Romeo_Handle(6)] = vrep.simxGetObjectHandle(clientID, 'Joint5'
       , vrep.simx_opmode_oneshot_wait);
56 [res Zebro_Romeo_Handle(7)] = vrep.simxGetObjectHandle(clientID, 'Joint6'
       , vrep.simx_opmode_oneshot_wait);
57
58 % Get Beacon Handles
59 [res Beacon_Handle(1)] = vrep.simxGetObjectHandle(clientID, '
       Beacon_Right_pickup', vrep.simx_opmode_oneshot_wait);
60 [res Beacon_Handle(2)] = vrep.simxGetObjectHandle(clientID, '
       Beacon_Left_pickup', vrep.simx_opmode_oneshot_wait);
61 [res Beacon_Handle(3)] = vrep.simxGetObjectHandle(clientID, '
       Beacon_Right_drop', vrep.simx_opmode_oneshot_wait);
62 [res Beacon_Handle(4)] = vrep.simxGetObjectHandle(clientID, '
       Beacon_Left_drop', vrep.simx_opmode_oneshot_wait);
63 [res Beacon_Handle(5)] = vrep.simxGetObjectHandle(clientID, '
       Beacon_Stairs', vrep.simx_opmode_oneshot_wait);
64 [res Beacon_Handle(6)] = vrep.simxGetObjectHandle(clientID, '
       Beacon_Rough_terrain', vrep.simx_opmode_oneshot_wait);
65 [res Beacon_Handle(7)] = vrep.simxGetObjectHandle(clientID, 'Beacon_Ramp'
       , vrep.simx_opmode_oneshot_wait);
66 [res Beacon_Handle(8)] = vrep.simxGetObjectHandle(clientID, '
       Beacon_Final_platform', vrep.simx_opmode_oneshot_wait);
67 [res Beacon_Handle(9)] = vrep.simxGetObjectHandle(clientID, '
       Beacon_Success', vrep.simx_opmode_oneshot_wait);
68
69 % Store the Beacon attributes as a struct
70
71 Beacon{1} = clientID;
72 Beacon{2} = vrep;
73 Beacon{3} = Beacon_Handle;
74
75 % Initiaze variables to save information from VREP:
76
77 for j = 2:7
78
79 [ret Juliet_Leg_Angle(1,j)] = vrep.simxGetJointPosition(clientID,
       Zebro_Juliet_Handle(j), vrep.simx_opmode_streaming); % Joint Position
80 [ret Romeo_Leg_Angle(1,j)] = vrep.simxGetJointPosition(clientID,
       Zebro_Romeo_Handle(j), vrep.simx_opmode_streaming); % Joint Position
81 [ret Juliet_Leg_Angle_Vel(1,j)] = vrep.simxGetObjectFloatParameter(
       clientID, Zebro_Juliet_Handle(j), 2012, vrep.simx_opmode_streaming); %
        Joint Velcoity
82 [ret Romeo_Leg_Angle_Vel(1,j)] = vrep.simxGetObjectFloatParameter(
       clientID, Zebro_Romeo_Handle(j), 2012, vrep.simx_opmode_streaming); %
       Joint Velcoity
83
84 end
85
86 [ret ] = vrep.simxGetObjectPosition(clientID,Zebro_Juliet_Handle(1), -1,
       vrep.simx_opmode_streaming); % Zebro Juliet Position
87 [ret ] = vrep.simxGetObjectPosition(clientID,Zebro_Romeo_Handle(1), -1,
       vrep.simx_opmode_streaming); % Zebro Romeo Position
```

```
88 [ret ] = vrep.simxGetObjectOrientation(clientID, Zebro_Juliet_Handle(1),
       -1,vrep.simx_opmode_streaming);
89 [ret ] = vrep.simxGetObjectOrientation(clientID, Zebro_Romeo_Handle(1),
       -1,vrep.simx_opmode_streaming);
90
91 for j=1:9
92
93     [res Beacon_Position(j,:)] = vrep.simxGetObjectPosition(clientID,
          Beacon_Handle(j),-1, vrep.simx_opmode_streaming); %Beacon Position
94
95 end
```

```
1  %% This is the Zebro_Juliet controller
2  %
3  % Measurements are:
4  %     t: is the simulation time
5  %     Position:  a vector with coordinates x,y,z
6  %     Orientation: Euler angles using the convention x-y-z
7  %     RotationMatrix: the 3x3 rotation matrix of zebro
8  %     State: a 6 dimentional vector with the angles of all legs
9  %     X: is the current x(k) from the max-plus equations
10 %     IncomingMessage: A message sent by the other robot, only if it is
11 %                      within range
12 %     Beacons: the coordinates of the beacons in space folloing the
13 %              ordering described in the miniproject description PDF
14 %
15 % Inputs:
16 %     Mode:  1 - for Max-Plus control. Requires 'Direction', 'Turn', '
       theta_T', 'theta_L' and 'A'
17 %            2 - for direct control of each individual leg. Requires '
       Reference'
18 %
19 %     Direction: This changes the direction of motion of the left and
       right
20 %                set of legs. Use:
21 %                1 - Go forward
22 %                2 - Go backwards
23 %                3 - Turn in place to the left
24 %                4 - Turn in place to the right
25 %
26 %     Turn: This is an offset introduced in the range of the touchdown
       and
27 %           liftoff angles, with the effect of creating smooth variable
28 %           radius turning while moving forward or backwards. The offset
       is
29 %           'theta_T + Turn' and 'theta_L - Turn' for the right legs and
30 %           'theta_T - Turn' and 'theta_L + Turn' for the left legs. This
31 %           means that you should make sure that  'Turn <= |theta_L -
       theta_T|,
32 %           otherwise you get a strange behaviour.
33 %
34 %     theta_T: is the touchdown angle of the legs. For example -0.4 is a
35 %              good number.
```

```
36 %
37 %     theta_L: is the liftoff angle of the legs. For example 0.4 is a
38 %              good number.
39 %
40 %     A: is the max-plus system matrix (see lecture notes)
41 %
42 %     Reference: is a 6 dimensional vector that sends directly the
43 %                reference angles of each leg to the Zebro robot. In v-
       rep
44 %                a PID controller tracks this reference.
45 %
46 %     SendMessage: you can use this variable to send a message to the
       other
47 %                  robot. Message only arrives if robots are within range
       .
48 %                  Otherwise the message is lost.
49 %
50 function [Mode, Direction, Turn, theta_T, theta_L, A, Reference,
       SendMessage] = Zebro_Juliet(t, chg, dir, tg, tp ,IncomingMessage)
51
52 if (IncomingMessage ~= '')
53     % process incoming message from other robot
54 end
55
56 %% Mode, direction are always 1
57 Mode = 1;
58
59 Direction = 1;
60
61 Turn = 0;
62
63 theta_T = -0.6; %default -0.4
64
65 theta_L = 0.6;  %default 0.4
66
67 %% Deciding the sufficient gait and gait parameter
68
69 gait = GenerateGait(chg);
70 td  = 0.1; %0.2
71
72 tf  =  0.5; % DONT FORGET TO CHANGE BACK INTO 0.5 !!!!!!!
73
74
75 if dir ==1 %Turning left
76 tg_left = tg + tp;
77 tf_left = tf;
78
79 tg_right= tg;
80 tf_right= tf + tp;
81
82 elseif dir == 2% Turning right
83 tg_right = tg + tp;
84 tf_right = tf;
```

```
85
86  tg_left= tg;
87  tf_left= tf + tp;
88
89  else
90  tg_right = 0;
91  tf_right = 0;
92
93  tg_left= 0;
94  tf_left= 0;
95  end
96
97
98
99  %% Constructing matrices
100
101 % Define R, M, N, T
102 R    = ComputeR(gait,td,tp,tg_left,tg_right);
103 T    = ComputeT(gait,td,tp,tg_left,tg_right);
104 M    = ComputeM(tp,tg_left,tg_right);
105 N    = ComputeN(tp,tg_left,tg_right);
106
107 siz = size(R); % size(P) == size(Q)
108
109 t_F = tf*eye(siz)  ;
110 t_G = tg*eye(siz) ;
111
112 % Define Identity matrix
113 E = MPIdentityMatrix(siz);
114
115 % Define Zero matrix
116 eNull = MPNullMatrix(siz);
117
118 % Construct matrix A0 and As
119
120
121 A0_11 = eNull;
122 A0_12 = t_F+E+M;
123 A0_21 = R;
124 A0_22 = eNull;
125
126 A0 = [A0_11 A0_12;A0_21 A0_22];
127 As = MPComputeAStar(A0);
128
129 % Construct matrix A1 and A
130 A1_11 = E;
131 A1_12 = eNull;
132 A1_21 = MPPlus(t_G+E+N,T);
133 A1_22 = E;
134
135 A1 = [A1_11 A1_12;A1_21 A1_22];
136
137 A = MPTimes(As,A1);
```

```
138
139 Reference = [1 1 1 1 1 1]*t;
140
141 SendMessage='';
142
143
144 end
```

```
1  function [Position Orientation State RMatrix] = Zebro_Read(clientID, vrep
       , Handle, Object)
2
3  %% Function reads the following from VREP Zebro :
4
5  % 1. X -> Discrete State events
6  % 2. Position -> Position of Zebro in World Frame
7  % 3. Heading -> Heading Direction of the Zebro in World Frame
8  % 4. State -> Individual Joints Angles and Joint Velocities of the Zebro
       Legs
9
10
11     for j = 2:7
12         % Joint Position
13         [ret State(1,j - 1)] = vrep.simxGetJointPosition(clientID, ...
14          Handle(j), vrep.simx_opmode_buffer);
15         % Joint Velcoity
16         [ret State(1,j + 5)] = vrep.simxGetObjectFloatParameter(clientID
               ,...
17          Handle(j), 2012, vrep.simx_opmode_buffer);
18
19     end
20
21
22     % Zebro Position
23     [ret Position(1,:)] = vrep.simxGetObjectPosition(clientID, ...
24     Handle(1), -1,vrep.simx_opmode_buffer);
25
26
27     % Zebro Orientation
28     [ret Orientation(1,:)] = vrep.simxGetObjectOrientation(clientID, ...
29     Handle(1), -1,vrep.simx_opmode_buffer);
30
31
32
33 [res retInts RMatrix retStrings retBuffer] = vrep.simxCallScriptFunction
       ...
34     (clientID,Object,vrep.sim_scripttype_childscript,...
35     'RotationMatrix',[],[],[],[],vrep.simx_opmode_blocking);
36
37
38 RMatrix = reshape(RMatrix,[4,3]);
39 RMatrix = RMatrix';
40 RMatrix = RMatrix(1:3,1:3);
41
```

```
42
43 end

1  function [X] = Zebro_Write(clientID, vrep, Object, Mode, Direction, Turn,
       theta_T, theta_L, A, Reference)
2
3  %% Function Writes the following to VREP Zebro :
4
5  % 1. Object -> Name of the Object to Which the Child Script is associated
6
7  % 2. Mode -> Choice of Mode of Operation
8      % Mode 1 -> Max-Plus Algebra based gait control
9      % Mode 2 -> Individual leg control
10
11 % Output X -> Discrete State events x(k)
12 %         RMatrix -> Rotation Matrix
13
14 Reference=[1 -1 1 -1 1 -1].*Reference;
15
16 X = [];
17
18 [res retInts retFloats retStrings retBuffer]=vrep.simxCallScriptFunction
       ...
19 (clientID,Object,vrep.sim_scripttype_childscript,...
20 'Mode_Select',Mode,[],[],[],vrep.simx_opmode_blocking);
21
22
23
24 if (Mode == 1)
25
26     [res retInts X retStrings retBuffer]=vrep.simxCallScriptFunction...
27     (clientID,Object,vrep.sim_scripttype_childscript,...
28     'Mode1',Direction,[Turn;theta_T;theta_L;reshape(A,[12*12,1])
           ],[],[],...
29     vrep.simx_opmode_blocking);
30
31 elseif(Mode == 2)
32
33     [res retInts retFloats retStrings retBuffer] = vrep.
           simxCallScriptFunction...
34     (clientID,Object,vrep.sim_scripttype_childscript,...
35     'Mode2',[],[Reference],[],[],vrep.simx_opmode_blocking);
36 end
37
38 %  [res retInts RMatrix retStrings retBuffer] = vrep.
       simxCallScriptFunction...
39 %     (clientID,Object,vrep.sim_scripttype_childscript,...
40 %     'RotationMatrix',[],[],[],[],vrep.simx_opmode_blocking);
41 %
42 % RMatrix = reshape(RMatrix,[4,3]);
43 % RMatrix = RMatrix';
44 % RMatrix = RMatrix(1:3,1:3);
45
```

```
46 end
```

```
1  function StopSimulator(vrep, clientID)
2
3  %% Stop simulation and Clean up the connection:
4
5  % Stop the simulation on VREP:
6  vrep.simxStopSimulation(clientID, vrep.simx_opmode_oneshot_wait);
7
8  % Close the connection to VREP:
9  vrep.simxFinish(clientID);
10
11 %Call the destructor and destroy the vrep object:
12 vrep.delete();
13
14 disp('Program Ended');
15 vrep.simxFinish(clientID);
16
17 end
```

```
1  function [M] = ComputeM(tp,tg_left,tg_right)
2
3  M = -inf*ones(6,6);
4
5  for i =1:1:6
6      if tg_left > tg_right %Turning left
7        if mod(i,2) == 0
8            M(i,i)=tp;
9        elseif mod(i,2) == 1
10           M(i,i) = 0;
11       end
12     elseif tg_left < tg_right %Turning right
13       if mod(i,2) == 0
14           M(i,i)= 0;
15       elseif mod(i,2) == 1
16           M(i,i) = tp;
17       end
18     elseif tg_left == tg_right %straight
19         M(i,i) = 0;
20     end
21 end
```

```
1  function [N] = ComputeN(tp,tg_left,tg_right)
2
3  N = -inf*ones(6,6);
4
5  for i =1:1:6
6      if tg_left > tg_right %Turning left
7        if mod(i,2) == 0
8            N(i,i)= 0;
9        elseif mod(i,2) == 1
10           N(i,i) = tp;
11       end
```

```
    elseif tg_left < tg_right %Turning right
      if mod(i,2) == 0
          N(i,i)= tp;
      elseif mod(i,2) == 1
          N(i,i) = 0;
      end
    elseif tg_left == tg_right %straight
        N(i,i) = 0;
    end
end
```

```
function [R] = ComputeR(gait,td,tp,tg_left,tg_right)

[row,m] = size(gait);
[row,n] = size(gait{1,1});

for i = 1:1:m
    for j = 1:1:n
        pq(i,j)= gait{1,i}{1,j};
    end
end

[ep,eq] = size(pq);

R = -inf*ones(6,6);

for ei = 1:(ep-1)
    for ej = 1:eq
         p(ei,ej) = pq(ei+1,ej);
         q(ei,ej) = pq(ei,ej);
    end
end

for ei = 1:(ep-1)
for ej = 1:eq
        if tg_left > tg_right %Turning left
        % q is even for td
        if mod(q(ei,ej),2)== 0
        R(p(ei,:),q(ei,ej))= td;
        elseif mod(q(ei,ej),2)== 1
        R(p(ei,:),q(ei,ej))= tp;
        end
        % q is odd for tp
        elseif tg_left < tg_right %Turning right
        % q is odd for td
        if mod(q(ei,ej),2)== 0
        R(p(ei,:),q(ei,ej))= tp;
        elseif mod(q(ei,ej),2)== 1
        R(p(ei,:),q(ei,ej))= td;
        end
        % q is even for tp
        elseif tg_left == tg_right
            R(p(ei,:),q(ei,:))= td;
```

```
        else
            R =[];
        end
end
end
```

```
function [T] = ComputeT(gait,td,tp,tg_left,tg_right)

[row,m] = size(gait);
[row,n] = size(gait{1,1});

for i = 1:1:m
    for j = 1:1:n
        pq(i,j)= gait{1,i}{1,j};
    end
end

[ep,eq] = size(pq);

T = -inf*ones(6,6);

p = pq(1,:);
q = pq(m,:);


for ej = 1:eq
        if tg_left > tg_right %Turning left
        % q is even for td
        if mod(q(ej),2)== 0
        T(p,q(ej))= td;
        elseif mod(q(ej),2)== 1
        T(p,q(ej))= tp;
        end
        % q is odd for tp
        elseif tg_left < tg_right %Turning right
        % q is odd for td
        if mod(q(ej),2)== 0
        T(p,q(ej))= tp;
        elseif mod(q(ej),2)== 1
        T(p,q(ej))= td;
        end
        % q is even for tp
        elseif tg_left == tg_right
            T(p,q)= td;
        else
            T =[];
        end
end
end
```

```
function [gait] = GenerateGait(chg)

if chg == 1
    gait = {{1,4,5},{2,3,6}};
elseif chg == 2
    gait = {{2,3,6},{1,4,5}};
elseif chg == 3
    gait = {{1,4},{3,6},{2,5}};
elseif chg == 4
    gait = {{1,4},{2,5},{3,6}};
elseif chg == 5
    gait = {{1,6},{2,3},{4,5}};
elseif chg ==6
    gait = {{1,6},{4,5},{2,3}};
else
    gait =[];
end
```

```
function As = MPComputeAStar(A)

siz = size(A,1);
As=MPIdentityMatrix(siz);
As_ = As;
for n=1:siz-1
    As=MPPlus(MPTimes(As,A),A);
end

As=MPPlus(As,As_);
```

```
function I = MPIdentityMatrix(siz)

I = -Inf*ones(siz);

for i=1:siz
    I(i,i)=0;
end
```

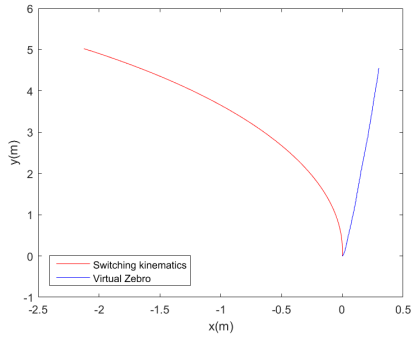
```
function E = MPNullMatrix(siz)

E = -Inf*ones(siz);
```

```
function C = MPPlus(A,B)

C=max(A,B);
```

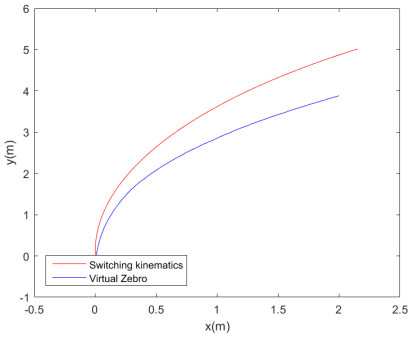
```
function x = MPTimes(a,b)

x=zeros(size(a,1),size(b,2));
for i=1:size(b,2)
    x(:,i) = max(bsxfun(@plus,a',b(:,i)),[],1)';
end
```

A-2 Extra Figures

A-3 Tripod 1

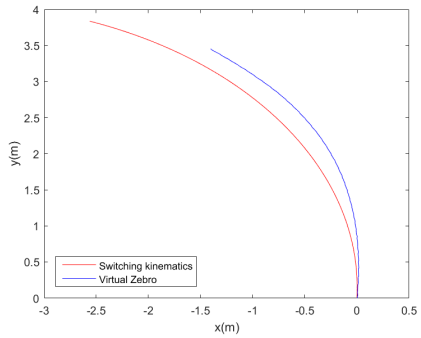


(a) Left direction

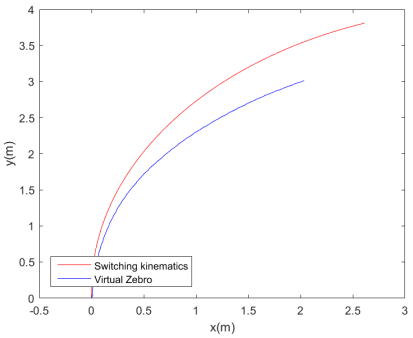


(b) Right direction

Figure A-1: $\tau_g = 0.4$ and $\tau_p = 0.1$

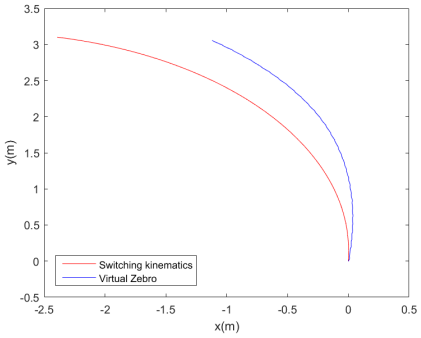


(a) Left direction

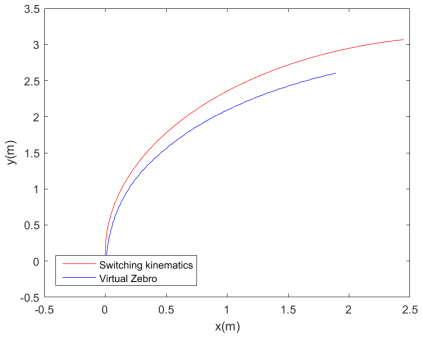


(b) Right direction

Figure A-2: $\tau_g = 0.4$ and $\tau_p = 0.2$

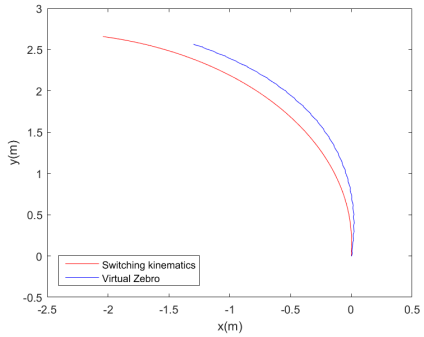


(a) Left direction

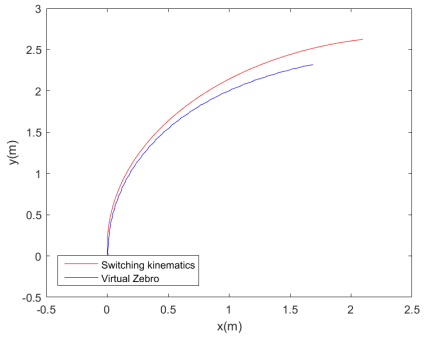


(b) Right direction

Figure A-3: $\tau_g = 0.4$ and $\tau_p = 0.3$

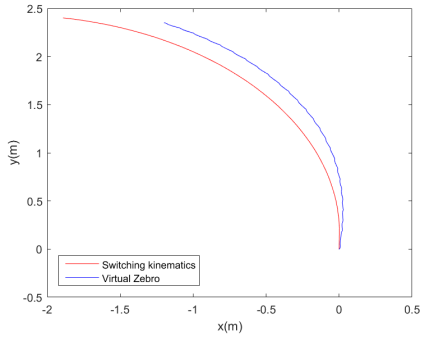


(a) Left direction

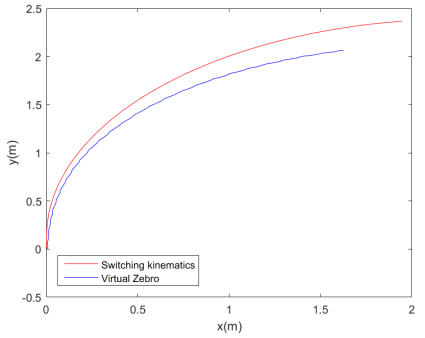


(b) Right direction

Figure A-4: $\tau_g = 0.4$ and τ_p =0.4



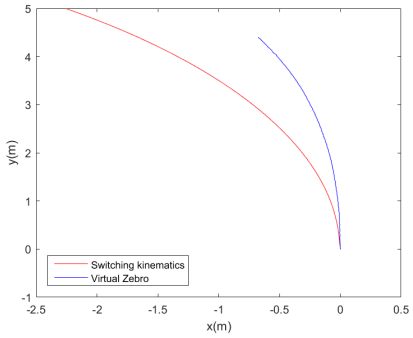
(a) Left direction



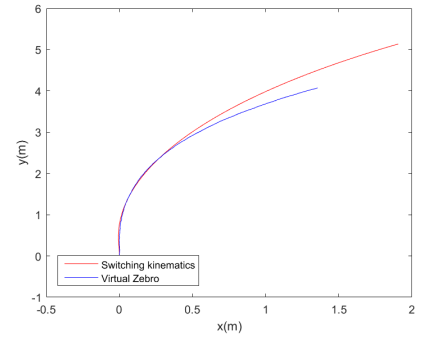
(b) Right direction

Figure A-5: $\tau_g = 0.4$ and τ_p =0.5

A-4 Tripod 2

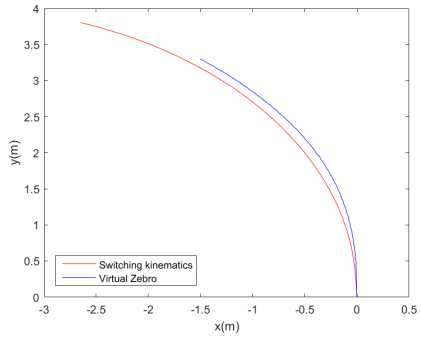


(a) Left direction

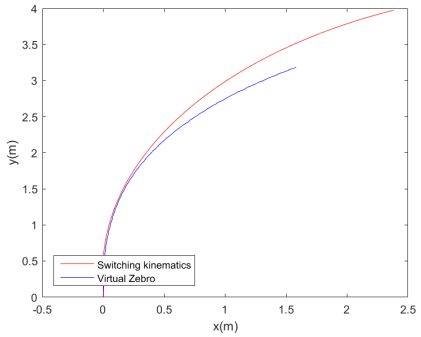


(b) Right direction

Figure A-6: $\tau_g = 0.4$ and τ_p =0.1

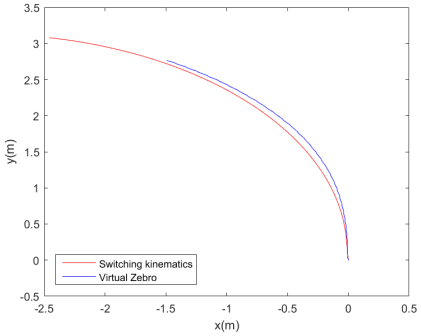


(a) Left direction

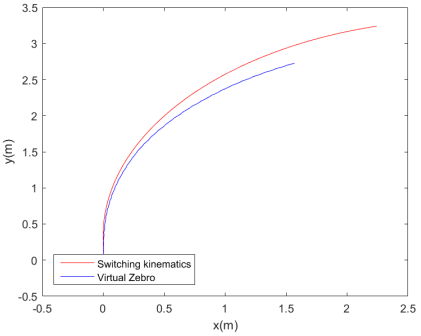


(b) Right direction

Figure A-7: $\tau_g = 0.4$ and τ_p =0.2

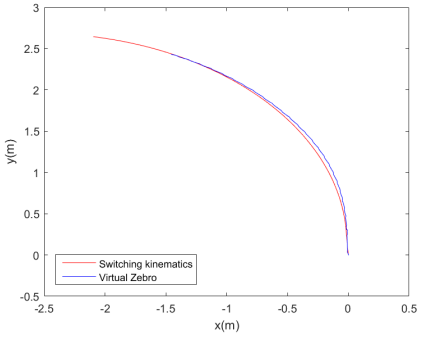


(a) Left direction

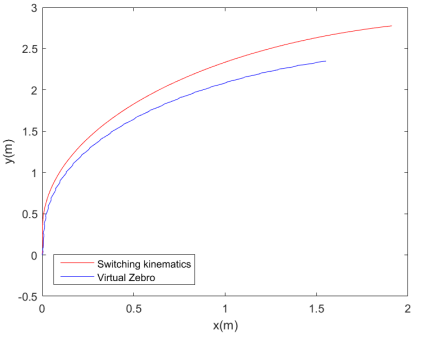


(b) Right direction

Figure A-8: $\tau_g = 0.4$ and τ_p =0.3



(a) Left direction

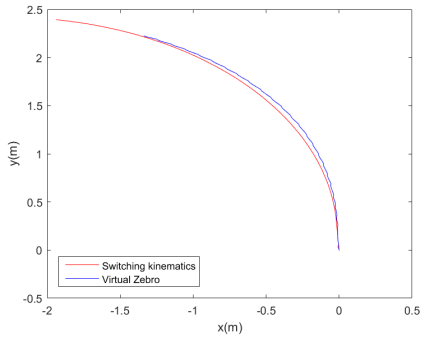


(b) Right direction

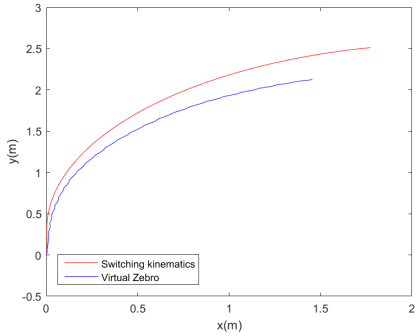
Figure A-9: $\tau_g = 0.4$ and τ_p =0.4

A-5 Quadruped 1

A-6 Quadruped 2

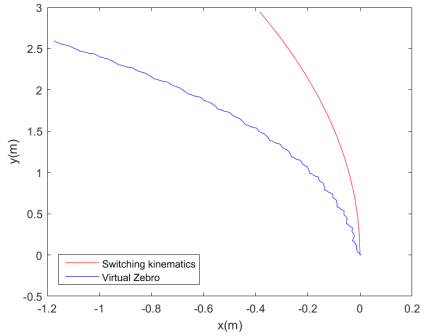


(a) Left direction

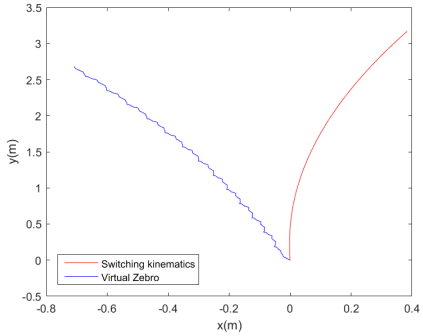


(b) Right direction

Figure A-10: $\tau_g = 0.4$ and $\tau_p = 0.5$

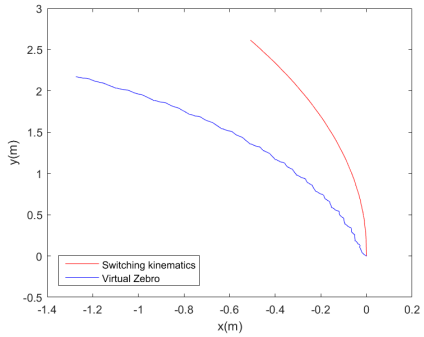


(a) Left direction

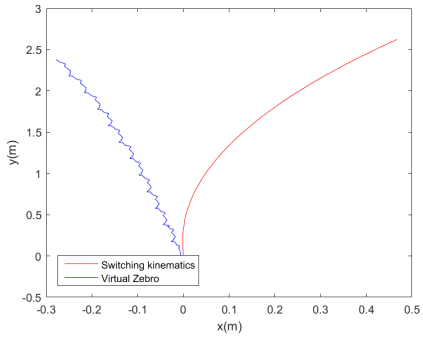


(b) Right direction

Figure A-11: $\tau_g = 0.4$ and $\tau_p = 0.1$

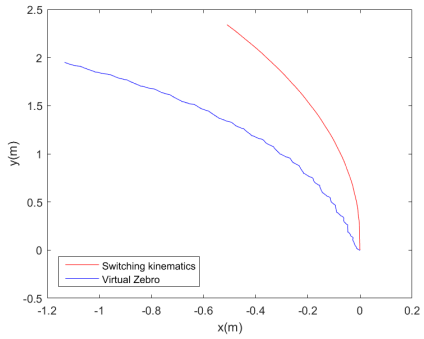


(a) Left direction

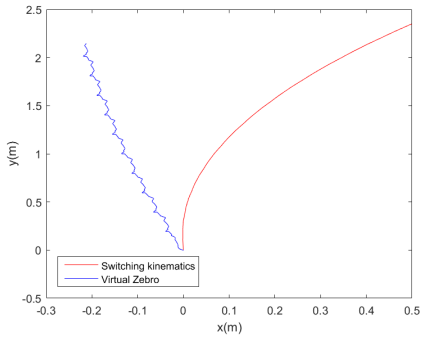


(b) Right direction

Figure A-12: $\tau_g = 0.4$ and $\tau_p = 0.2$

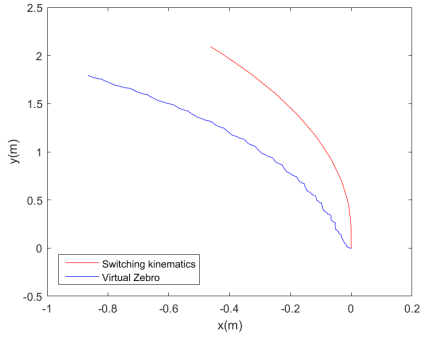


(a) Left direction

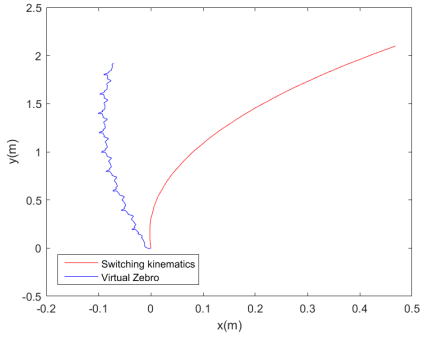


(b) Right direction

Figure A-13: $\tau_g = 0.4$ and $\tau_p = 0.3$

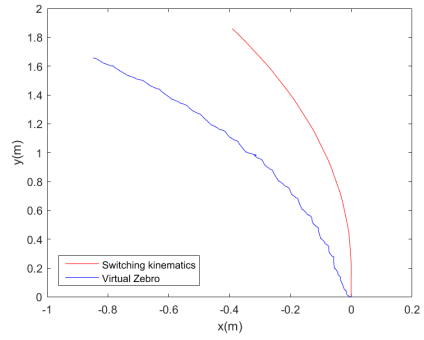


(a) Left direction

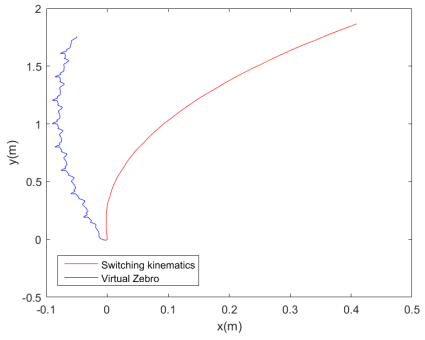


(b) Right direction

Figure A-14: $\tau_g = 0.4$ and $\tau_p = 0.4$

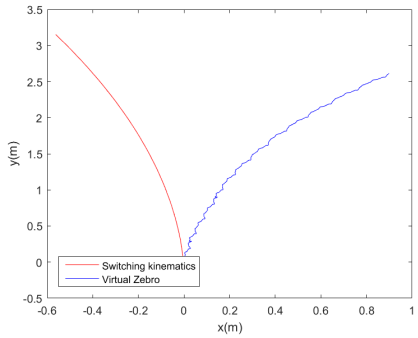


(a) Left direction

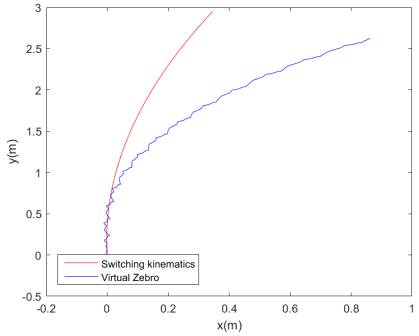


(b) Right direction

Figure A-15: $\tau_g = 0.4$ and $\tau_p = 0.5$

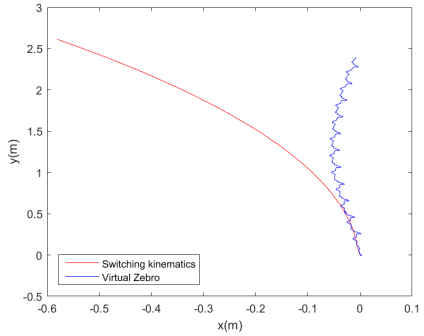


(a) Left direction

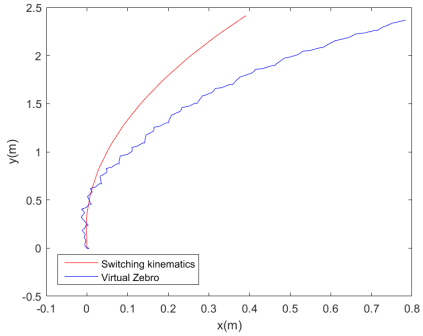


(b) Right direction

Figure A-16: $\tau_g = 0.4$ and $\tau_p = 0.1$

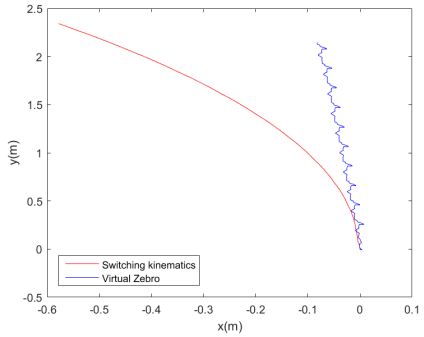


(a) Left direction

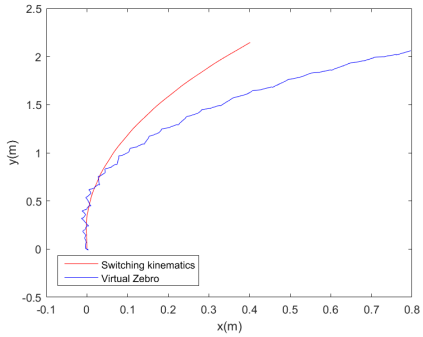


(b) Right direction

Figure A-17: $\tau_g = 0.4$ and $\tau_p = 0.2$

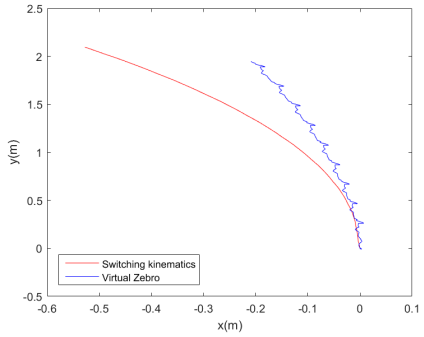


(a) Left direction

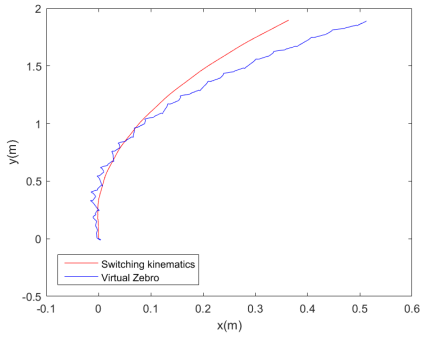


(b) Right direction

Figure A-18: $\tau_g = 0.4$ and $\tau_p = 0.3$

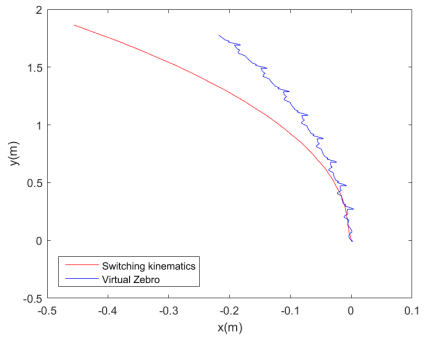


(a) Left direction

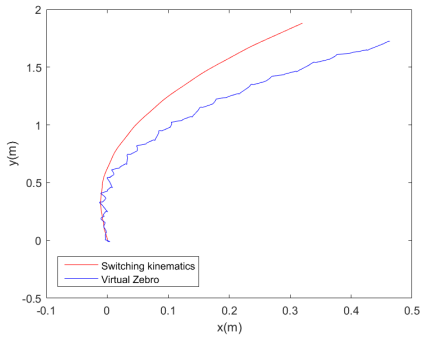


(b) Right direction

Figure A-19: $\tau_g = 0.4$ and τ_p =0.4



(a) Left direction



(b) Right direction

Figure A-20: $\tau_g = 0.4$ and τ_p =0.5

Bibliography

[1] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat, *Synchronization and linearity: an algebra for discrete event systems*. John Wiley & Sons Ltd, 1992.

[2] B. Heidergott, G. J. Olsder, and J. Van Der Woude, *Max Plus at work: modeling and analysis of synchronized systems: a course on Max-Plus algebra and its applications*. Princeton University Press, 2014.

[3] R. A. Cuninghame-Green, *Minimax algebra*, vol. 166. Springer Science & Business Media, 2012.

[4] G. A. D. Lopes, B. Kersbergen, T. Van den Boom, B. De Schutter, and R. Babuska, "Modeling and control of legged locomotion via switching max-plus models," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 652–665, 2014.

[5] P. F. P. Muir and C. C. P. Neuman, "Kinematic modeling of wheeled mobile robots," *Journal of robotic systems*, vol. 4, no. June, pp. 281–340, 1987.

[6] U. Saranli, M. Buehler, and D. E. Koditschek, "RHex: A Simple and Highly Mobile Hexapod Robot," *The International Journal of Robotics Research*, vol. 20, no. July, pp. 616–631, 2001.

[7] K. Waldron and V. Vohnout, "Configuration Design of the Adaptive Suspension Vehicle," *The Intl Journal of Robotics Research*, vol. 3, pp. 37–48, 1984.

[8] A. M. Hoover, S. Burden, X. Y. Fu, S. S. Sastry, and R. S. Fearing, "Bio-inspired design and dynamic maneuverability of a minimally actuated six-legged robot," *2010 3rd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics, BioRob 2010*, pp. 869–876, 2010.

[9] A. M. Hoover, E. Steltz, and R. S. Fearing, "RoACH: An autonomous 2.4 g crawling hexapod robot," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 26–33, 2008.

[10] N. J. Kohut, A. M. Hoover, K. Y. Ma, S. S. Baek, and R. S. Fearing, "MEDIC: A legged millirobot utilizing novel obstacle traversal," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 802–808, 2011.

[11] J. Morrey, B. Lambrecht, a.D. Horchler, R. Ritzmann, and R. Quinn, "Highly mobile and robust small quadruped robots," *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 1, no. October, pp. 0–5, 2003.

[12] P. Birkmeyer, K. Peterson, and R. S. Fearing, "DASH: A dynamic 16g hexapedal robot," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 2683–2689, 2009.

[13] S. Kim, J. E. Clark, and M. R. Cutkosky, "ISprawl: Design and tuning for high-speed autonomous open-loop running," *International Journal of Robotics Research*, vol. 25, no. 9, pp. 903–912, 2006.

[14] G. A. D. Lopes, R. Babuška, B. De Schutter, and A. J. J. Van Den Boom, "Switching max-plus models for legged locomotion," *2009 IEEE International Conference on Robotics and Biomimetics, ROBIO 2009*, vol. 19, pp. 221–226, 2009.

[15] B. De Schutter and T. van den Boom, "Max-plus algebra and max-plus linear discrete event systems: An introduction," *2008 9th International Workshop on Discrete Event Systems*, pp. 36–42, 2008.

[16] T. J. J. van den Boom and B. De Schutter, "Modelling and control of discrete event systems using switching max-plus-linear systems," *Control Engineering Practice*, vol. 14, no. 10, pp. 1199–1211, 2006.

[17] M. Gondran and M. Minoux, *Graphs and algorithms*. Wiley, 1984.

[18] Bart Kersbergen, *Modeling and Control of Switching Max-Plus-Linear Systems: Rescheduling of railway traffic and changing gaits in legged locomotion*. PhD thesis, Delft University of Technology, 2015.

[19] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[20] G. A. D. Lopes, B. Kersbergen, T. Van Den Boom, B. De Schutter, and R. Babuška, "On the eigenstructure of a class of max-plus linear systems," *Proceedings of the IEEE Conference on Decision and Control*, pp. 1823–1828, 2011.

[21] G. A. D. Lopes, T. J. J. Van Den Boom, B. De Schutter, and R. Babuska, "Modeling and Control of Legged Locomotion via Switching Max-Plus Systems," *Proceedings of the 10th International Workshop on Discrete Event Systems (WODES 2010)*, vol. 43, no. 12, pp. 382–387, 2010.

[22] B. Kersbergen, G. A. Lopes, T. J. van den Boom, B. De Schutter, and R. Babuška, "Optimal gait switching for legged locomotion," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2729–2734, IEEE, 2011.

[23] Y. O. Aydin, A. Saranli, Y. Yazicioglu, U. Saranli, and K. Leblebicioglu, "Optimal control of a half-circular compliant legged monopod," 2014.

[24] M. M. Ankarali, E. Sayginer, Y. Yazicioglu, A. Saranli, and U. Saranli, "A Dynamic Model of Running with a Half-Circular Compliant Leg," *Adaptive Mobile Robotics: Proceedings of the 15th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pp. 1–8, 2012.

[25] Chun Sheng Cai and B. Roth, "On the planar motion of rigid bodies with point contact," *Mechanism and Machine Theory*, vol. 21, no. 6, pp. 453–466, 1986.

[26] M. Otten, "DeciZebro: the design of a modular bio-inspired robotic swarming platform," msc thesis, Delft University of Technology, 2017.

[27] T. Hesselmans, A. Steenkamp, L. Tideman, and L. Wouterse, "New legs for the zebro mobile robot," bachelor thesis, Delft University of Technology, January 2017.

Glossary

List of Acronyms

DCSC	Delft Center for Systems and Control
TU Delft	Delft University of Technology