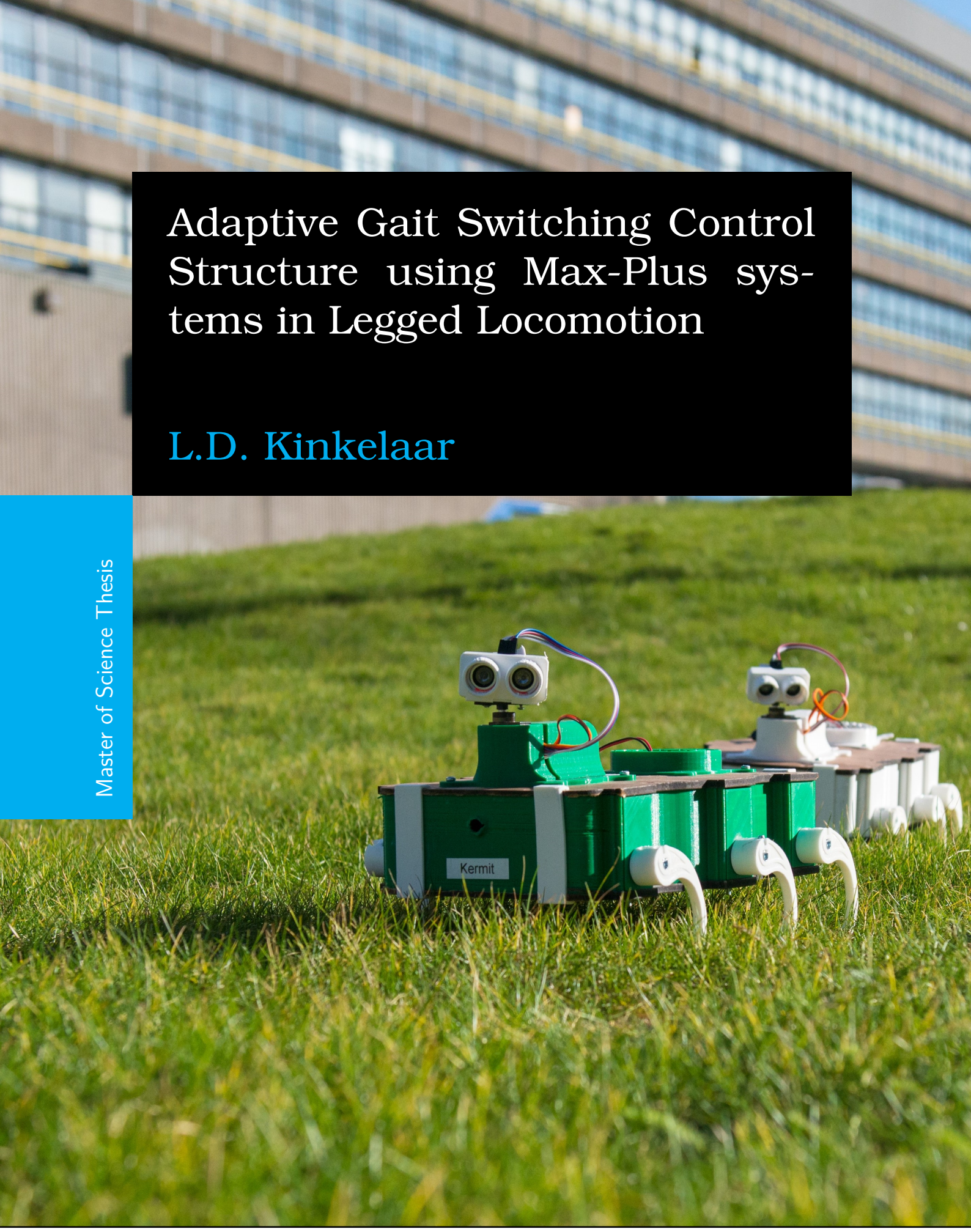
Adaptive Gait Switching Control Structure using Max-Plus systems in Legged Locomotion

L.D. Kinkelaar

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Adaptive Gait Switching Control Structure using Max-Plus systems in Legged Locomotion

Master of Science Thesis

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

L.D. Kinkelaar

May 29, 2018

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

The work in this thesis was supported by the ZeBro-project. Their cooperation is hereby gratefully acknowledged.

Abstract

The ZebRo (Dutch abbreviation: Zesbenige Robot, six-legged robot), is a walking robot designed by the TU Delft [1], with its foundations on the RHex[2]. The current version of the Zebro project, the DeciZebro[3], is made for the research to swarming in robotics, and is about the size of an A4-paper.
The most important feature of legged robots, is the walking. Creating a walking pattern depends on a lot of variables, such as the size and weight of the robot, the amount of legs, the shape and amount of links of the legs, the power of the engines, etc. The locomotion pattern of the RHex is determined by a central pattern generator (CPG) which contains a time-based function for the legs to follow. This method is computationally cheap, but lacks certain aspects. Most importantly, when legs are hindered or delayed, the standard pattern needs to be changed for all legs to counter the delay. Switching between different gaits, as well as the construction of new gaits also requires recalculations of the walking pattern.
To counter these shortcomings, Max-Plus algebra can be used for the timing of the legs of walking robots. This can be done by modeling a walking cycle as two discrete events. These events are the times of the lift-off and touchdown, and can be determined by using a Max-Plus Discrete Event System (DES) framework. The main purpose of these events is to determine two separate periods; the stance period, and the swing or flight period.
Max-Plus algebra is defined as a semi-ring over the union of real numbers and minus infinity. The standard binary operations are taking the maximum ($\oplus$) and addition ($\otimes$), which can be used for scalar and matrix operations.
Using these matrix operations, a Max-Plus linear (MPL) system can be constructed, of the form: $v(k+1) = A \otimes v(k)$.[4] The Max-Plus linear systems contains the gait matrix A, and the vector $v(k)$ containing the lift-off and touchdown times of vector instance k. One of the strengths of this method is the ability to change gaits, by simply changing the A-matrix. This then results in the switching Max-Plus linear (SMPL) system: $v(k+1) = A^k \otimes v(k)$.[4]
This SPML system determined in [4] only contains gaits with successive recirculating leg-groups, where the synchronization of the lift-off of a certain leg-group i relies on the touchdown of the leg-group $i-1$. This allows for gaits like walking for humans, or a slow trot for horses, where a leg-group remains on the ground until the previous leg-group has recirculated. In order to increase the amount of possible gaits, the synchronization of successive leg-groups

is extended to not only contain the previous leg-groups' touchdown times, but also their lift-off times. This extension allows for gaits containing multiple rotation leg-groups at the same time, making gaits like the gallop possible. Because of the desired stable situation of grounded legs, the synchronization is chosen to not extend the timing of the touchdown, as delays should not hinder the return of the legs to their grounded position.
Using the new system and the work of W. Suriana [23], which proposed a novel method of using SMPL-systems for turning gaits, a complete locomotion was introduced in the Zebro. The work of Suriana was previously validated using simulations, but is shown to work in actual robot implementations as well.
The performance of different gaits is tested on different surfaces, to determine the influence of the amount of legs on the ground, and determining whether gait changes could influence the success rate of the system regarding traversing rough terrain. Inertia measurement units (IMU) are used for detecting rough terrain, and tests show that the body movement of a Zebro on both flat and rough terrain are heavily influenced by the gaits used.
Because of several limitations regarding the actuation of the legs, gaits containing moments of static instability could not be tested. However, the framework for implementation regarding the Max-Plus is lain, and tested using simulations.

Keywords: legged locomotion, max-plus, discrete event system, gait switching, Zebro

Contents

Preface and Acknowledgments

This thesis came to be thanks to the help of the ZeBro project and the Delft Center of System and Control. Special thanks to Mattijs Otten, who made the construction of the robot possible, and kick-started the project with its current direction.
Also special thanks to my supervisors, Ton van den Boom, Edwin Hakkenes and Chris Verhoeven, for creating a positive atmosphere of experimentation and research, and guiding me. I also want to thank the rest of the Zebro project for helping me when necessary, and teaching me parts of their disciplines. And of course, I have to thank my family and friends for their support during my whole time at the TU Delft.

Chapter 1

Introduction

1-1 Background

Steadily, the impact of robots on our everyday life is increasing. While some warehouses are completely automated, using either driving robots or railed systems, the adaptations necessary for completely automating processes are quite extensive. Walking robots could be an intermediate step in between humans and wheeled robots, by being able to go anywhere humans can go.
The development of these walking robots is relative complicated. Companies such as Boston Dynamics show great promise with their robots [5], which have either two [6], four [7] or six legs[8]. Other walking robots like the Asimo [9], and TU Delft's Zebro (Zesbenige Robot, Dutch for Six-Legged Robot)[3][10] are also in constant development, and may be able to demonstrate their effectiveness. Some of these robots may be used to assist humans inside difficult terrain, like woods, indoors or rocky areas. While fallen trees and boulders can be major obstacles, indoor places like offices can contain other difficult areas, like stairs and doorsteps, which are much easier conquered by walking robots than their wheeled counterparts.
In order to walk, there needs to be some kind of synchronization between legs, so the robot does not fall down. Two-legged animals like kangaroos and ostriches, but also humans, have a small repertoire of gaits, walking styles. These gaits consist of walking, hopping, skipping and running, where differences lie in the legs used and the order of legs touching down and lifting off. Depending on the situation at hand, there might be a change in what gait is the most efficient to use. [11]
Compared to two-legged animals, the gait options for four- and six-legged animals are much more numerous, as can be seen by the different gaits of horses. A method to capture all these gaits, and implement them in different robots by generating the timing, can be achieved using various methods[12][13]. In this report, the method of using Max-Plus algebra for gait generation[4] is researched and implemented.
Max-Plus algebra changes certain aspects from the standard algebra, by replacing the times-

operator by the plus-operator and the plus-operator by the max-operator. Max-Plus algebra allows timed events to be easily captured in linear systems, which is especially useful for the planning and timing of train schedules, factory processes and gait generation[14].

1-2 Problem Statement

1. How should Max-Plus discrete event systems (DES) be implemented for the locomotion of a walking robot?
2. How can the current Max-Plus model be extended for gaits with aerial phases?
3. What is the performance of the Max-Plus locomotion system?

1-2-1 Current Situation

For the timing of walking robots, widely used methods are the Central Pattern Generator (CPG) and the Buehler clock. A CPG generates reference trajectories for each leg, by solving sets of differential equations [4]. The Buehler clock uses a reference trajectory in every leg, which is (in the case of the RHex) tracked by a PID-controller [2].
In these implementations, delay handling and gait change requires a change in the trajectory generation of all legs, and every change in speed or performance problems require hard-coded gaits and routines.
These (mostly static) gaits are widely used in a lot of walking robots. Robots with running gaits are not that widespread, due to the difficulty of making the control algorithms, but also due to the costs of strong and light actuators, reliable sensors and the requirements of computing power inside the walking system. The implementation of this more complicated, dynamic running behavior of the robots is mostly reserved for more expensive systems.

1-2-2 Improved situation

In order to improve the shortcomings of the current gait-generation, switching Max-Plus linear systems will be implemented and tested. The aim is to make an adaptable, complete locomotion module for the Deci-ZebRo[3]. The main use of the Deci-ZebRo will be to test swarming within a group of robots, much like ants, fish or birds. This testing will be done indoors and outdoors, which makes a walking robot the perfect platform for testing.
Handling rough terrain, stairs, hills and flat ground should be in the repertoire of the Zebro, in order to allow the swarming to function without problem. Nature can be a good source of inspiration in order to find ways to handle these different environments. It can be seen that animals change their gaits when increasing or decreasing their speed[15], or while walking on difficult terrain [16]. It makes sense this will also be incorporated in the walking robot, as the switching Max-Plus linear system allows for fluent and fast changes in the gait, which can be optimized for the situation at hand.
When the robot needs to transverse a big plot of land, the speed requirement will increase. Running gaits can improve the speed, but have difficulties to solve certain parameters0, like stability, body movement and timing issues. Especially in robots with limited computing

power and feedback, the full state of the body is hard to determine, and even harder to control real-time. It will be tried to solve this problem with relative simple control mechanics.

1-2-3 Subproblems

- What are the best performing gaits for the different terrains?
- Using limited sensor input, how can the optimal gait change be determined?
- How can a controller help to improve the performance of running gaits?

1-3 Scope and outline

This report will be about the implementation of a complete Max-Plus discrete event system based locomotion module. The module is responsible for the execution of instructions by a governing top-level controller. It should be capable to provide the timing of the steps of each individual leg, creating and changing gaits appropriate for the task at hand. The module is not responsible for the direct control of the leg-motors, and not responsible for the choice of direction or speed, although some situations can make the locomotion module to limit the speed of the robot.
Furthermore, it should be able to control the robot in such a way that it can make corners, handle delays and solve defects. The locomotion module should essentially translate the directions of above lying decision making to the individual leg controllers so the robot can complete its tasks. A simplified overview of the system can be seen in Figure 1-1.

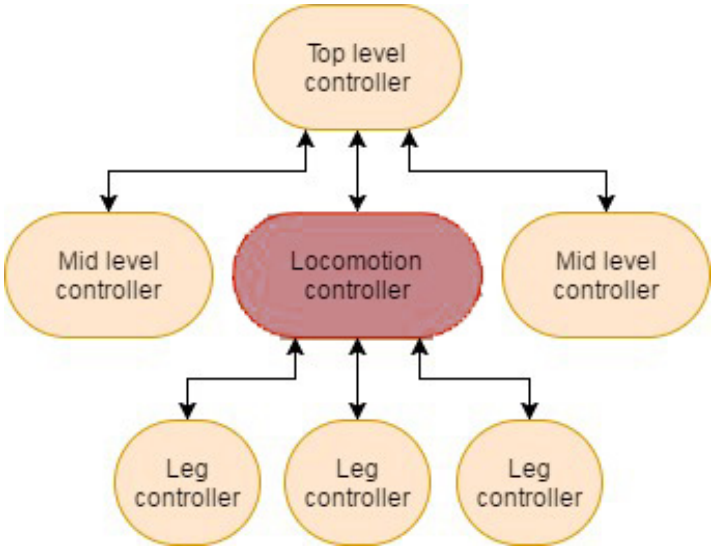


Figure 1-1: A simplified overview of the control system in the Zebro

The top-level controller uses input from communication or vision modules (other mid-level controllers) for the desired walking direction and speed. In order to translate these instructions for the leg controllers, the locomotion controller is used. These translated instructions

are abstracted to contain the desired position and direction of rotation in a certain amount of time. The leg controller will then make a linear interpolation between the current position and the desired position of the leg, and follow that path using a PID-controller.

Next to this, the addition of running gaits in Max-Plus will be researched. For now, the gait generation in Max-Plus for legged locomotion focuses on static gaits. Extending this to having the possibility to construct a lot more different gaits increases the range of performance. Running gaits are defined as gaits with either moments of statical instability or aerial time. This allows for a tool which can incorporate the dynamics of the body of the walking robot, which opens the path for combining the Max-Plus discrete event systems with the dynamics of its system.

Part I

Background and Literature

Chapter 2

Max-Plus Algebra

The purpose of this chapter is to give a general outline to the basics of Max-Plus algebra. In the first section, the operators, algebraic properties and special elements of the Max-Plus algebra will be presented. The second chapter will give the link between the use of Petri nets, a kind of graphs, and Max-Plus algebra, as this sets the basics for using Max-Plus to determine timed events. After this, the calculation and use of matrices and eigenvalues in Max-Plus will be explained. This foundation will be used to define the state space systems necessary for the timing of the locomotion in the fourth section of this chapter.

2-1 Basics

To use Max-Plus algebra in controlling the legs of the robot, the foundation of the Max-Plus algebra needs to be fully understood. Therefore, this section covers the basics of Max-Plus algebra.

2-1-1 Scalar operations

Standard algebra consists of several operations learned in elementary school, addition and multiplication. These operations can be expanded to division, subtraction, powers , etc. Max-Plus algebra differs from most standard algebra, in the way that addition is replaced by taking the maximum of the two elements and multiplication is replaced by addition. These operations are denoted by a plus-symbol and a multiplication symbol surrounded by a circle:

$$x \oplus y = \max(x, y) \tag{2-1}$$
$$x \otimes y = x + y \tag{2-2}$$
$$\text{With: } x, y \in R_\epsilon \tag{2-3}$$
$$\text{Where: } \mathbb{R}_\epsilon \triangleq \mathbb{R} \cup -\infty \tag{2-4}$$

Because of major analogies between $\oplus$ and $\otimes$ on one side and $+$ and $\times$ on the other side, the choice is made to make the signs similar, and to call the $\oplus$ operator the Max-Plus addition, and the $\otimes$ operator the Max-Plus multiplication. Furthermore, the Max-Plus power operation is defined as:

$$x^{\otimes y} = \overbrace{x \otimes x \otimes \ldots x}^{\text{y times}} = y \times x \tag{2-5}$$

Contrary to standard algebra however, there are no inverts of addition and multiplication used in Max-Plus algebra. These are used in regular algebra (minus and division), and this is one of the major differences between Max-Plus and regular algebra. A few examples can be seen in: Appendix A-1 to A-3 and Appendix A-4 to A-6.

2-1-2 Definitions and properties

For standard algebra, we have 0 and 1 as the zero and unit element for respectively addition and multiplication. For Max-Plus algebra, these elements are respectively defined by $\epsilon = -\infty$ and $e = 0$ such that:

$$x \oplus \epsilon = x \tag{2-6}$$
$$x \otimes \epsilon = \epsilon \tag{2-7}$$
$$x \otimes e = x \tag{2-8}$$

It can be seen that the zero-element ϵ shows similarities with zero in regular algebra, where: $x + 0 = x$, $x \times 0 = 0$ and $x \times 1 = x$.

The algebraic properties of Max-Plus that are shared with the properties of standard algebra are:

Commutativity: For $x, y \in R_\epsilon$:

$$x \oplus y = y \oplus x \tag{2-9}$$
$$x \otimes y = y \otimes x \tag{2-10}$$

Associativity: For $x, y, z \in R_\epsilon$:

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z \tag{2-11}$$
$$x \otimes (y \otimes z) = (x \otimes y) \otimes z \tag{2-12}$$

Distributivity: For $x, y, z \in R_\epsilon$:

$$x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z) \tag{2-13}$$

Some examples can be seen in: Appendix A-7 to A-13.

2-1-3 Vector and Matrix calculations

The Max-Plus algebra can be extended to matrix and vector operations. These operations show a lot of similarities with regular algebra. Matrix addition in Max-Plus algebra is defined as:

$$\text{For } A, B, C \in \mathbb{R}_{\epsilon}^{n \times m}\textbf{:} \tag{2-14}$$
$$A \oplus B = C, \text{ with:} \tag{2-15}$$
$$c_{i,j} = a_{i,j} \oplus b_{i,j} \tag{2-16}$$

Similarly to regular algebra matrix addition, this is solely possible if A and B are of the same size.

In order to describe matrix multiplications, it is easiest to first define matrix-vector multiplications:

$$\text{For } A \in \mathbb{R}_{\epsilon}^{n \times m}, B \in \mathbb{R}_{\epsilon}^{m \times 1} \text{ and } C \in \mathbb{R}_{\epsilon}^{n \times 1}\textbf{:} \tag{2-17}$$
$$A \otimes B = C, \text{ where:} \tag{2-18}$$
$$c_i = a_{i,1} \otimes b_1 \oplus a_{i,2} \otimes b_2 .... \oplus a_{i,m} \otimes b_m \tag{2-19}$$

Now, we can now define matrix multiplication as:

$$\text{For } A \in \mathbb{R}_{\epsilon}^{n \times m}, B \in \mathbb{R}_{\epsilon}^{m \times p} \text{ and } C \in \mathbb{R}_{\epsilon}^{n \times p}\textbf{:} \tag{2-20}$$
$$A \otimes B = C, \text{ Where:} \tag{2-21}$$
$$c_{i,j} = a_{i,1} \otimes b_{1,j} \oplus a_{i,2} \otimes b_{2,j} .... \oplus a_{i,m} \otimes b_{m,j} \tag{2-22}$$

Additionally, the zero and identity matrix of regular algebra ($0_{n \times n}$ and $I_{n \times n}$) have Max-Plus algebra counterparts. These are defined by $\mathcal{E}$ and E, where $\mathcal{E}$, the zero matrix, is a matrix consisting only of $\epsilon = -\infty$. E is the identity matrix, which consists of $e = 0$ on the diagonal ($E_{ij} = e$ for $i = j$), and $\epsilon = -\infty$ on all other entries. The identity matrix is strictly square ($E \in \mathbb{R}_{\epsilon}^{n \times n}$).

$$\mathcal{E}_{n\times n} = \overbrace{\begin{bmatrix} \epsilon & \dots & \epsilon \\ \vdots & \ddots & \vdots \\ \epsilon & \dots & \epsilon \end{bmatrix}}^{\text{n}} \Big\}\text{n} \tag{2-23}$$

$$E_{n\times n} = \overbrace{\begin{bmatrix} e & \epsilon & \dots & \epsilon & \epsilon \\ \epsilon & e & \epsilon & \dots & \epsilon \\ \vdots & \ddots & \ddots & \ddots & \epsilon \\ \epsilon & \dots & \epsilon & e & \epsilon \\ \epsilon & \dots & \epsilon & \epsilon & e \end{bmatrix}}^{\text{n}} \Big\}\text{n} \tag{2-24}$$

Now, define matrix $A_{n\times m} \in {\mathbb{R}_\epsilon}^{n\times m}$, the following properties of $\mathcal{E}$ and E are used:

$$A \oplus \mathcal{E}_{m\times m} = A \tag{2-25}$$
$$A \otimes \mathcal{E}_{m\times m} = \mathcal{E}_{n\times m} \tag{2-26}$$
$$A \otimes E_{m\times m} = A \tag{2-27}$$

As can be seen, the matrix multiplication is the same in Max-Plus algebra as it is in regular algebra, with the only difference that the $+$ is replaced by $\oplus$ and the $\times$ is replaced by $\otimes$. Again, some examples can be seen in: Appendix A-14 to A-17.

Eigenvalues and eigenvectors

Like in regular algebra, the eigenvectors and eigenvalues of matrices in Max-Plus algebra posses useful information about the matrix. Let $A \in \mathbb{R}_\epsilon^{n\times n}$, then if there exists a $\lambda \in \mathbb{R}_\epsilon$ and $v \neq \mathcal{E}^{n\times 1} \in \mathbb{R}_\epsilon^n$, such that $A \otimes v = \lambda \otimes v$. In this case, λ is a Max-Plus algebraic eigenvalue of A, with v as its corresponding eigenvector. There are multiple ways to determine the eigenvalues and eigenvectors. One of them is the power algorithm, the method used in this report and in the implementation. An extended explanation of this method can be read in the article 'The Power Algorithm in Max Algebra'[17].

For now, the regular power algorithm is used, which is given by [17]:

1. Take an arbitrary vector $x_0 \neq \mathcal{E}^{n\times 1}$.
2. Multiply this vector with A several times, until $A^{k+m} \otimes x_0 = \lambda^m A^k x_0$.
3. Let $x(l) = A^{k+l}x_0$ with $l = 0, \dots, m$. Then $x(m) = \lambda^m x(0)$.
4. Define the vector $v = \frac{1}{m}[x(0) + x(1) \cdots + x(m-1)]$.

An example can be seen in Appendix A-1-3

2-2 Graphs

The method of visualizing a certain timed event by using graphs increases the understanding of the whole system, which helps the use of Max-Plus algebra in discrete event systems. [14]

The graphs that are of importance to the Max-Plus algebra are directed weighted graphs. Every graph consists nodes and arcs, or more formal: graph $\mathcal{G}$ consists of $(\mathcal{N}, \mathcal{D})$, With $\mathcal{N}$ a (finite) set of nodes, and $\mathcal{D} \subset \mathcal{N} \times \mathcal{N}$, the arcs that connect them.

For directed graphs, there is the possibility that the arc of node i to node j (arc(i,j)) might be different than arc(j,i) or even non-existent. For undirected arcs, this is not the case. To add a value to a certain path in graphs, weights ($w(i,j) \in \mathbb{R}_\epsilon$) are added to each arc.

The use of graphs for events like trains arriving at and leaving from stations, arriving at certain parts of supply lines in factories or the walking of a legged robot can show clear overviews of the events in a certain procedure. An example of a graph and its corresponding Max-Plus matrix can be seen in Figure 2-1. The Max-Plus matrix is a representation of the time or other quantity between two certain phases or events. The path from node i to node j is defined in the ji^{th} element of the matrix. It is not always possible to go from every node to every other node. In this case, the path will have a length of $\epsilon = -\infty$ to signify this.

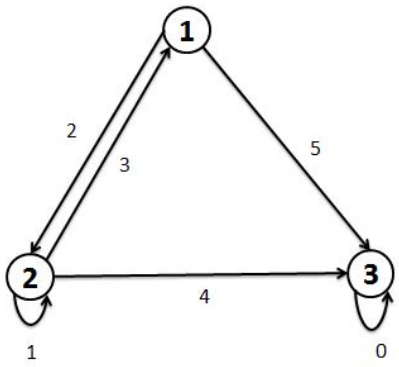


Figure 2-1: Example of a directed weighted graph

The corresponding Max-Plus matrix A for this graph then becomes:

$$A = \begin{pmatrix} \epsilon & 3 & \epsilon \\ 2 & 1 & \epsilon \\ 5 & 4 & 0 \end{pmatrix} \tag{2-28}$$

An example regarding this graph and matrix can be seen in Appendix A-2-1.

For any two nodes i and j, a series of interconnection arcs form a path $p = (i_k, j_k) \in \mathcal{D}(A)$ with $k \in 1, 2, \ldots, m$, $i_1 = i$, $j_k = i_{k+1}$ for $k < m$, and $j_m = j$.

The length of the path is defined by the m, with the following notation: $|p|_l = m$. The weight of a path $|p|_w$ is defined by the sum of the weights of all arcs that the path consists of. Finally, the average weight of a path p is defined as $|p|_{aw} = \frac{|p|_w}{|p|_l}$.

Special types of paths are circuits, which are paths which end where they begin ($i = j$). An elementary circuit is a circuit where all nodes of the circuit have one outgoing and one ingoing path, allowing for only one possible circuit.

The weight of the path from node i to node j can be seen clearly if the path is only one arc long. The maximal weight of the path from node i to node j in a certain amount of steps can also be calculated using Max-Plus algebraic matrix powers:

$$A^{\otimes k} = \overbrace{A \otimes A \otimes \cdots \otimes A}^{k \text{ times}} \tag{2-29}$$

The maximal path $|p_{ij}^k|_w$ from node i to node j in k steps then becomes:

$$|p_{ij}^k|_w = [A^{\otimes k}]_{ji} \tag{2-30}$$

Some examples regarding this can be seen in A-2-2.

This concept can be taken even further, by defining the maximal path from a certain node to a certain node, regardless of the amount of steps.

$$A^+ \triangleq \bigoplus_{k=1}^{\infty} A^{\oplus k} \tag{2-31}$$

This matrix can be simplified if the average weight of the circuits is smaller than e:

$$\text{For:} \quad A \in \mathbb{R}_{\epsilon}^{n \times n} \tag{2-32}$$

$$A^+ \triangleq \bigoplus_{k=1}^{n} A^{\oplus k} \tag{2-33}$$

This relation will later be used in the construction for the gait matrices.

Chapter 3

Max-Plus in Legged Locomotion

Now the foundation of understanding Max-Plus algebra has been lain, the next step is to understand how the Max-Plus can be used for the implementation as central pattern generator (CPG). A CPG is module that generates the order legs of a robot or even an animal should move in. Needless to say, the CPG is crucial in order to make a robot walk. In several researches [12],[13],[18],[2], different methods to define gaits have been used.

Even though the manual implementation of patterns is relatively straightforward, implementing different gaits and gait transitions is harder to achieve. The use of Max-Plus algebra for the central pattern generator allows for more variety in this regard, and makes rapid addition of new gaits possible when the general framework is in place.

3-1 Basics

In order to model a certain gait using Max-Plus, the gait needs to be fitted as a discrete event system (DES). In order to discretize the continuous event of a step, points of interest need to be determined. This is done by defining the touchdown and lift-off times of each separate leg. This information can than be used to make a trajectory for each leg, which can be followed by the leg controller. The use of Max-Plus for the control of the legs has been clearly explained in the research of G. A. D. Lopes, B. Kersbergen, T. J. J. van den Boom, B. De Schutter and R. Babuška[4]. This section is a short recap of this article.

The touchdown and lift-off times are concatenated in a vector $x(k)$ which contains the information of event k, the k^{th} full rotation of each leg[19]. The parameters and variables of interest can be seen in 3-1.

Sign	Parameters and state variables
$x(k)$	Total state vector of the touchdown and lift-off times of the separate legs of event k
$t_i(k)$	Touchdown time of leg i at event k
$l_i(k)$	Lift-off time of leg i at event k
i	Leg index
k	Event counter
θ_l	Lift-off angle of the legs
θ_t	Touchdown angle of the legs
τ	Current time
τ_f	Flight time of the leg
τ_g	Ground time of the leg
τ_δ	Double stance time

Table 3-1: Gait parameters and state variables [19]

The flight, ground and double stance time can be adjusted to influence the velocity of the robot. The flight time is the duration the leg spends in the air to return to the position needed to make the next step. The double stance time is the overlapping time between two subsequent steps, and is used to synchronize the legs before the next step is taken. The ground time of the leg represents the duration the leg spends in the supporting phase.

For a six-legged robot, the total state vector of touchdown and lift-off times $x(k)$ is as follows:

$$x(k) = \begin{bmatrix} t_1(k) \\ t_2(k) \\ t_3(k) \\ t_4(k) \\ t_5(k) \\ t_6(k) \\ l_1(k) \\ l_2(k) \\ l_3(k) \\ l_4(k) \\ l_5(k) \\ l_6(k) \end{bmatrix}$$

The touchdown time can be calculated by adding the flight time to the moment of lift-off. The lift-off time is more difficult to determine, as it depends ground time. The ground time on its turn depends on the gait that is used, and on the flight and double stance time.

This results in two simple equations, which can determine the lift-off and touchdown times[19]:

$$t_i(k+1) = l_i(k+1) \otimes \tau_f \tag{3-1}$$
$$l_i(k+1) = t_i(k) \otimes \tau_g \tag{3-2}$$

In Equations 3-1 and 3-2, it can be seen that the legs do not rely on other legs in order to

move. For almost all gaits, it is strongly preferred that the legs react on delays or inaccuracies of not only itself, but also on the position of other legs.

This can be done by adding terms that define a certain order in which the robot is supposed to move its legs. For example, in the case of the wave gait ($1 \to 2 \to 3 \to 4 \to 5 \to 6$), it is necessary to add extra conditions to the time instants.

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

$$l_i(k+1) = \begin{cases} t_i(k) \otimes \tau_g \oplus t_6(k) \otimes \tau_\delta & \text{for } i = 1 \\ t_i(k) \otimes \tau_g \oplus t_{i-1}(k+1) \otimes \tau_\delta & \text{for } i = 2, ..., 6 \end{cases} \tag{3-4}$$

As can be seen from Equations 3-1 to 3-4, the touchdown times of event $k+1$ depend on the lift-off times of $k+1$ and vice versa. This results in a system where $x(k+1)$ is defined as Equation 3-5, and show dependency of both $x(k)$ and $x(k+1)$.

$$x(k+1) = A_0 x(k+1) \oplus A_1 x(k) \tag{3-5}$$

Now, following Equations 3-1 to 3-2 for all legs, and implementing this in Equation 3-5, results (for a six-legged robot) in the system seen in 3-6. On the diagonal of A_1, $e = 0$ is placed. This is to make sure that the lift-off and touchdown events of iteration $k+1$ can never be earlier than the events of iteration k.

$$x(k+1) = \left(\begin{array}{c|c} \mathcal{E}_{6\times6} & \tau_f \otimes E_{6\times6} \\ \hline \mathcal{E}_{6\times6} & \mathcal{E}_{6\times6} \end{array}\right) \otimes x(k+1) \oplus \left(\begin{array}{c|c} E_{6\times6} & \mathcal{E}_{6\times6} \\ \hline \tau_g \otimes E_{6\times6} & E_{6\times6} \end{array}\right) \otimes x(k) \tag{3-6}$$

Now, as can be seen in the results of adding a certain sequence to the movement of the different legs in Equations 3-3 and 3-4, some additional terms need to be included in the matrices to define the order of the legs. These terms are defined by the matrices P and Q. The method for the generation of these matrices will be explained in Section 3.2. The system is now extended to Equation 3-7:

$$x(k+1) = \left(\begin{array}{c|c} \mathcal{E}_{6\times6} & \tau_f \otimes E_{6\times6} \\ \hline P & \mathcal{E}_{6\times6} \end{array}\right) \otimes x(k+1) \oplus \left(\begin{array}{c|c} E_{6\times6} & \mathcal{E}_{6\times6} \\ \hline \tau_g \otimes E_{6\times6} \oplus Q & E_{6\times6} \end{array}\right) \otimes x(k) \tag{3-7}$$

Due to P being nilpotent in the Max-Plus sense, which is further explained in the research of Lopes et al.[20], it is possible to calculate A_0^*.

$$A_0^+ = \bigoplus_{k=0}^{\infty} A^{\oplus k} \tag{3-8}$$

$$A_0^* = E \oplus A_0^+ \tag{3-9}$$

This allows for the description of $x(k+1)$ in a non-implicit form, which results in Equation 3-12.

$$x(k+1) = A_0 \otimes x(k+1) \oplus A_1 \otimes x(k) \tag{3-10}$$
$$x(k+1) = A_0^* \oplus A_1 \otimes x(k) \tag{3-11}$$
$$x(k+1) = A \otimes x(k) \tag{3-12}$$

This A-matrix contains all the information needed for the implementation of a certain gait in the system.

3-2 Gait Matrices

For the implementation of the gaits, it is important to understand how the matrices P and Q can be defined. It is possible to devise a huge amount of gaits using these two matrices. It has to be noted, that using this method, it is only possible to let each leg rotate once per cycle, or never at all.

First, the order in which the legs make their steps needs to be determined. This is done by making groups of legs which depart after the arrival of the group before them. This overview of groups of legs and the order in which they depart is called the gait, with the following definition:

$$\mathcal{G} = l_1 \prec l_2 \cdots \prec l_m \tag{3-13}$$

Where l_i is defined as a group of legs from the set $1, 2, \ldots, n$ for a n-legged system. Between two groups of legs, there will be no overlap in the included legs. The more formal definition of this is given in [Optimal gait switching for legged locomotion]:

$$\bigcup_{p=1}^{m} l_p = 1, 2, \ldots, n \tag{3-14}$$
$$\forall i \neq j : l_i \cap l_j = \emptyset \tag{3-15}$$

Now, the construction of the matrices P and Q is possible using any gait determined according to Equations 3-13 to 3-15.
This is done following the procedure defined in Equations 3-16 - 3-17.

Let P and Q be square matrices of the size $n \times n$ with n the number of legs.
To define P, for $\forall j = \{1, ..., m-1\}$, with m the amount of leg groups, $\forall p \in l_{j+1}$ and $\forall q \in l_j$:

$$P_{i,j} = \begin{cases} \tau_\delta & \text{for} \quad P_{p,q} \\ \epsilon & \text{for every other} \quad P_{i,j} \end{cases} \tag{3-16}$$

To determine Q, for $\forall v \in l_1$ and $\forall w \in l_m$:

$$Q_{i,j} = \begin{cases} \tau_\delta & \text{for} \quad Q_{v,w} \\ \epsilon & \text{for every other} \quad Q_{i,j} \end{cases} \tag{3-17}$$

The determined P and Q matrices can now be implemented in Equation 3-7 in order to calculate the A_0 and A_1 matrices. For a gait defined by:

$$\mathcal{G}_s = \{1,2\} \prec \{3,4\} \prec \{5,6\} \tag{3-18}$$

This results in the following P and Q matrices:

$$P = \begin{bmatrix} \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \tau_\delta & \tau_\delta & \epsilon & \epsilon & \epsilon & \epsilon \\ \tau_\delta & \tau_\delta & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \tau_\delta & \tau_\delta & \epsilon & \epsilon \\ \epsilon & \epsilon & \tau_\delta & \tau_\delta & \epsilon & \epsilon \end{bmatrix}, \qquad Q = \begin{bmatrix} \epsilon & \epsilon & \epsilon & \epsilon & \tau_\delta & \tau_\delta \\ \epsilon & \epsilon & \epsilon & \epsilon & \tau_\delta & \tau_\delta \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix} \tag{3-19}$$

A fully worked out example can be found in Appendix A-3.

3-3 Gait Switching

The transition between gaits in animals are almost always optimal. For walking robots, the transitions between those gaits need to be defined for stable behaviour. The research of B. Kersbergen et al. [21], accurately describes how gaits can transition into each other. The report will be followed for the gaits of a six-legged robot.

3-3-1 Gait Transitions

To fully understand the transitions, the difference between gaits need to defined accurately. As gaits are defined by the order in which legs move, there are almost infinite possibilities to walk. For a six-legged robot, there already are numerous walking patterns. Let's look at three different tripod 2-grouped gaits, i.e. gaits with 2 groups of 3 legs that revolve:

$$\text{Tripod gait 1:} \quad \mathcal{G}_{t1} = \{1,4,5\} \prec \{2,3,6\} \tag{3-20}$$

$$\text{Tripod gait 2:} \quad \mathcal{G}_{t2} = \{2,3,6\} \prec \{1,4,5\} \tag{3-21}$$

$$\text{Tripod gait 3:} \quad \mathcal{G}_{t3} = \{1,3,5\} \prec \{2,4,6\} \tag{3-22}$$

As can be seen, the gaits are all different. However, gait 1 and gait 2 have the same groups of legs that alternate. Gait 3 has different groups of legs that rotate together.
The behaviour of a robot walking with gait 1 will be identical to the behaviour of the same

robot walking with gait 2, while gait 3 will operate remarkably different.
In the transient behaviour while changing gaits, the difference between gait 1 and gait 2 will show, due to the difference in starting step. The use of Max-Plus algebra makes sure the transitions between gaits are always done with enough legs on the ground, but if the subsequent gaits are not compatible, the time to make a transition will be big. For faster and more elegant movement, this transition time needs to be minimized.
To illustrate the difference in transient behaviour between gaits 1 and 2, the transition to gait $\mathcal{G}_{ts} = \{1,4\} \prec \{3,6\} \prec \{2,5\}$ will be shown in Figure 3-1.

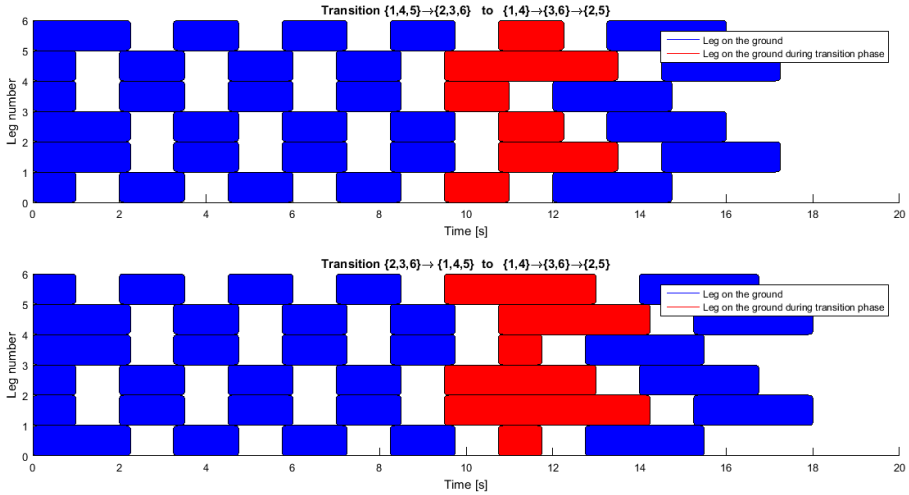


Figure 3-1: Difference in gait transition from $\mathcal{G}_{t1}$ or $\mathcal{G}_{t2}$ to $\mathcal{G}_{ts}$, where coloured blocks represent the leg on the ground ($\tau_f = 1[s], \tau_\delta = 0.25[s], \tau_g = 1[s]$)

It can be seen that, because of the sloppy gait transition, the same amount of steps take approximately 1 second longer. This is due to the way Max-Plus works in calculating the new lift-off times, when it is required for the legs to stay on the ground for τ_g when they just touched down.
The transition from gait 2 shows a moment when all legs are on the ground for an relative long period around 11 seconds, after legs $\{1,4,5\}$ have touched down. Now because legs $\{1,4\}$ need to recirculate immediately, but are also required to stay on the ground for τ_g, this results in a delay. Now, following [21] again, the method to quantify the effectiveness of the gait transition is done by determining how certain legs move to different groups.

When the Zebro switches gaits, this means that one or more legs changes the group it operates in. For instance, one leg changes from group i to group j. Now assume every cycle starts at a new $t = 0$. For a certain leg in group i, the leg lifts off at time instant:

$$t = (\tau_f \otimes \tau_\delta)^{\otimes i} \tag{3-23}$$

Of course, when the leg is in group j, the time of lift-off becomes:

$$t = (\tau_f \otimes \tau_\delta)^{\otimes j} \tag{3-24}$$

Now, the time it takes for the leg to synchronize with the new group depends on how many groups the leg moves. So the larger the difference between i and j, the bigger the ground time variance will be.
For $j > i$, this results in an extra ground time for the switching leg, defined by:

$$(\tau_f \otimes \tau_\delta)^{\otimes j-i)} \tag{3-25}$$

Equivalently, if $i > j$, not the switching leg, but the group the leg switches to will have a postponed lift-off. The duration of this delay will be given by:

$$(\tau_f \otimes \tau_\delta)^{\otimes i-j)} \tag{3-26}$$

So, the difference between i and j determines the amount of delay. For instance, the transition of the gait for the following gaits:

$$\{1\} \prec \{2,3\} \prec \{4,\underbrace{\mathbf{5}\} \prec \{6\} \rightarrow \{1\} \prec \{2,3\} \prec \{4\} \prec \{\mathbf{5}}},6\} \tag{3-27}$$

Will result in a smaller delay than the transition between the gaits:

$$\{1\} \prec \{2,\underbrace{\mathbf{3}\} \prec \{4,5\} \prec \{6\} \rightarrow \{1\} \prec \{2\} \prec \{4,5\} \prec \{\mathbf{3}}},6\} \tag{3-28}$$

To measure the 'quality' of the gait transition, B. Kersbergen et al. suggests to derive a quantification for how well the system performs during the transition by using:

$$\bar{\sigma} = \frac{\sigma(\tau_{g1}, \tau_{g2}, \ldots \tau_{gn})}{\tau_g} \tag{3-29}$$

Where τ_{gi} for $i = 1 \ldots n$ in a n-legged system, is the actual time spent on the ground during the transition. The standard deviation of the ground time during the transition is denoted by σ. For the transitions seen in Equations 3-27 & 3-28, the resulting gait transitions and can be seen in Figure 3-2. For the gait transition in Equation 3-27, the $\bar{\sigma} = 0.51$, for Equation 3-28, $\bar{\sigma} = 1.02$

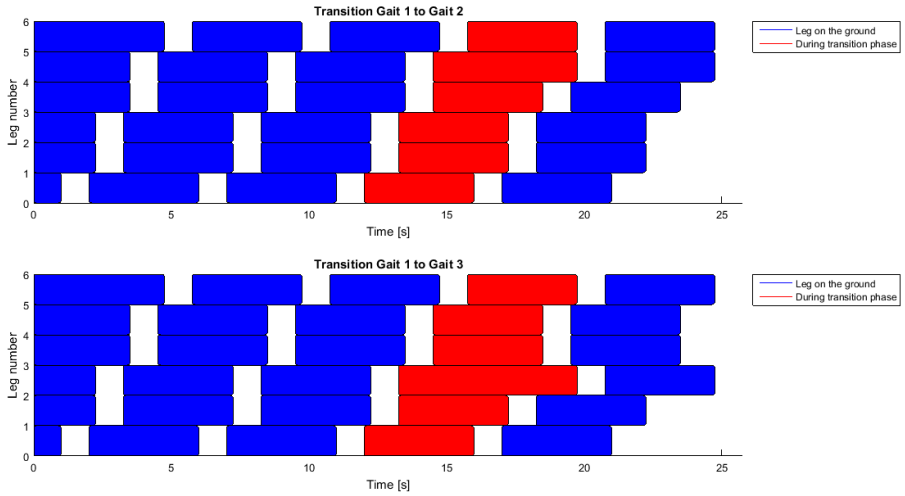


Figure 3-2: Gait transitions given by Equation 3-27 & 3-28.

3-3-2 Transition Parameters

To optimize the transition between gaits, individual flight times of the legs can be changed during the transition[4][21]. The synchronization will then not be hindered by certain legs staying on the ground longer than necessary.
This is done by changing the system slightly to allow for different flight times in one gait. The system changes from Equation 3-30 to Equation 3-31.

$$x(k+1) = \left(\begin{array}{c|c} \mathcal{E}_{6\times6} & \tau_f \otimes E_{6\times6} \\ \hline P & \mathcal{E}_{6\times6} \end{array}\right) \otimes x(k+1) \oplus \left(\begin{array}{c|c} E_{6\times6} & \mathcal{E}_{6\times6} \\ \hline \tau_g \otimes E_{6\times6} \oplus Q & E_{6\times6} \end{array}\right) \otimes x(k) \tag{3-30}$$

$$x(k+1) = \left(\begin{array}{c|c} \mathcal{E}_{6\times6} & R \\ \hline P & \mathcal{E}_{6\times6} \end{array}\right) \otimes x(k+1) \oplus \left(\begin{array}{c|c} E_{6\times6} & \mathcal{E}_{6\times6} \\ \hline \tau_g \otimes E_{6\times6} \oplus Q & E_{6\times6} \end{array}\right) \otimes x(k) \tag{3-31}$$

$$\text{With:} \qquad R = \begin{pmatrix} \tau_{f1} & \epsilon & \dots & \epsilon \\ \epsilon & \tau_{f2} & \epsilon & \dots \\ \vdots & \ddots & \ddots & \vdots \\ \epsilon & \dots & \epsilon & \tau_{f6} \end{pmatrix} \tag{3-32}$$

So, the A-matrix depends on the gait, but also on the variables like the double stance time. A can be defined as a function dependent on these variables:

$$A(\mathcal{G}, \tau_f \otimes E, \tau_g, \tau_\delta) \tag{3-33}$$

To determine the R-matrix, calculating the eigenvectors of the different gait matrices is necessary. Using the method described in Section 2-1-3, the eigenvectors can be determined. For

gaits $\mathcal{G}_1$ and $\mathcal{G}_2$, the corresponding eigenvectors are given by:

$$v_1 = \begin{pmatrix} t_{\mathcal{G}11} \\ \vdots \\ t_{\mathcal{G}16} \\ l_{\mathcal{G}11} \\ \vdots \\ l_{\mathcal{G}16} \end{pmatrix} = \begin{pmatrix} t_{\mathcal{G}1} \\ l_{\mathcal{G}1} \end{pmatrix} \quad \text{and} \quad v_2 = \begin{pmatrix} t_{\mathcal{G}21} \\ \vdots \\ t_{\mathcal{G}26} \\ l_{\mathcal{G}21} \\ \vdots \\ l_{\mathcal{G}26} \end{pmatrix} = \begin{pmatrix} t_{\mathcal{G}2} \\ l_{\mathcal{G}2} \end{pmatrix} \tag{3-34}$$

Following the algorithm proposed in the research of B. Kersbergen et al.[21], the transition τ_f parameters can be determined.
Given gaits $\mathcal{G}_1$ and $\mathcal{G}_2$ and their respective A-matrices:

1) Determine τ_{extra} following Equation 3-35

$$\tau_{extra} = (l_{\mathcal{G}2} - t_{\mathcal{G}1}) - \min(l_{\mathcal{G}2} - t_{\mathcal{G}1}) \tag{3-35}$$

2) If $\tau_{f\mathcal{G}1} \geq max(\tau_{extra})$, a transition vector can be computed using:

$$\tau_{trans1} = \begin{pmatrix} \tau_{f\mathcal{G}1} - \tau_{extra(1)} \\ \vdots \\ \tau_{f\mathcal{G}1} - \tau_{extra}(6) \end{pmatrix} \tag{3-36}$$

R is defined as:

$$R = diag(\tau_{trans1}) \tag{3-37}$$

The sequence of A-matrices during the transition can then be defined as:

$$\begin{aligned} &\vdots \\ &A(\mathcal{G}_1, \tau_{f\mathcal{G}1} \otimes E, \tau_{g\mathcal{G}1}, \tau_{\delta\mathcal{G}1}) \\ &A(\mathcal{G}_1, R(\tau_{trans1}), \tau_{g\mathcal{G}1}, \tau_{\delta\mathcal{G}1}) \\ &A(\mathcal{G}_2, \tau_{f\mathcal{G}2} \otimes E, \tau_{g\mathcal{G}2}, \tau_{\delta\mathcal{G}2}) \\ &\vdots \end{aligned}$$

3) For the case that $\tau_{f\mathcal{G}1} < max(\tau_{extra})$, two transmission matrices are necessary:

$$A(\mathcal{G}_1, R(\tau_{trans1}), \tau_{g\mathcal{G}1}, \tau_{\delta\mathcal{G}1}) \tag{3-38}$$
$$A(\mathcal{G}_2, R(\tau_{trans2}), \tau_{g\mathcal{G}2}, \tau_{\delta\mathcal{G}2}) \tag{3-39}$$

The transition vectors are calculated using:

$$\tau_{trans1}(i) = \max(\min(\tau_{extra}(i), \tau_{f\mathcal{G}1}), \tau_{fmin}) \tag{3-40}$$
$$\tau_{trans2}(i) = \tau_{f\mathcal{G}2} - (\tau_{trans1}(i) - \tau_{extra}(i)) - \min(\tau_{t\mathcal{G}1} - \tau_{extra}) \tag{3-41}$$

Where τ_{fmin} is defined as the minimal flight time possible due to the physics of the robot.
The sequence of the A-matrices then becomes:

$$\vdots$$
$$A(\mathcal{G}_1, \tau_{f\mathcal{G}1} \otimes E, \tau_{g\mathcal{G}1}, \tau_{\delta\mathcal{G}1})$$
$$A(\mathcal{G}_1, R(\tau_{trans1}), \tau_{g\mathcal{G}1}, \tau_{\delta\mathcal{G}1})$$
$$A(\mathcal{G}_2, R(\tau_{trans2}), \tau_{g\mathcal{G}2}, \tau_{\delta\mathcal{G}2})$$
$$A(\mathcal{G}_2, \tau_{f\mathcal{G}2} \otimes E, \tau_{g\mathcal{G}2}, \tau_{\delta\mathcal{G}2})$$
$$\vdots$$

Using these steps, the transition can be improved even further. An example to illustrate the difference between using no optimal gait change, no gait transitional parameters and using gait transition parameters can be seen in Figure 3-3. The gaits are defined as:

$$\mathcal{G}_1 = \{1, 4, 5\} \prec \{2, 3, 6\} \tag{3-42}$$
$$\mathcal{G}_2 = \{3, 6\} \prec \{1, 4\} \prec \{5, 2\} \tag{3-43}$$
$$\mathcal{G}_3 = \{1, 4\} \prec \{2, 5\} \prec \{3, 6\} \tag{3-44}$$

It can be noticed that the transition from gait 1 to gait 3 is the optimal one due to the smallest amount of changes in the leg groups.

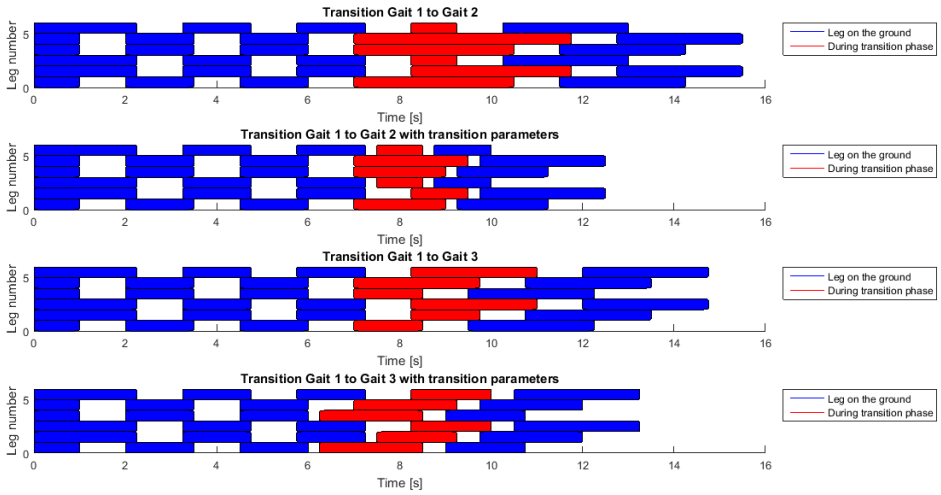


Figure 3-3: Gait transitions given by Equations 3-42 - 3-44. The gait parameters are the same for each gait, and are given by: $\{\tau_f, \tau_g, \tau_\delta\} = \{1, 1, 0.25\}$, and the minimal leg flight time $\tau_{fmin} = 0.5$. After 5 steps, the simulation is stopped.

It is interesting to see the difference in the time it takes to complete the gait transition with or without transition parameters. Furthermore, the time it takes to change the gait for the non-optimal transition shows a slightly faster result. It can however be seen that the walking pattern after the gait transition is not fully stable yet, and that an additional step is required to obtain a regular walking pattern.
The standard deviation measure to determine the quality of the gait change $\bar{\sigma}$ (Equation

3-29) is calculated for these four situations and can be seen in Table 3-2:

Gait transition		
Gaits	Transition parameters	$\bar{\sigma}$
$\mathcal{G}_1 \rightarrow \mathcal{G}_2$	No	1.5309
$\mathcal{G}_1 \rightarrow \mathcal{G}_2$	Yes	0.6275
$\mathcal{G}_1 \rightarrow \mathcal{G}_3$	No	0.6847
$\mathcal{G}_1 \rightarrow \mathcal{G}_3$	Yes	0.2739

Table 3-2: Gait transition quality for the gait transitions given by Equations 3-42 - 3-44. The gait parameters are the same for each gait, and are given by: $\{\tau_f, \tau_g, \tau_\delta\} = \{1, 1, 0.25\}$

The stability of the leg movement is better when the transition parameters are used. Even though the gait change with the sub-optimal gait transition was faster, the walking behaviour of the robot shows less variance when the optimal transition is used. The effectiveness of the gait transition parameters are obvious, and can be seen in both the time as well as the variance of the steps.

3-4 Modeling delays

Fluctuations in the height of the terrain walked on may result in a different time of touchdown or lift-off. In the research of D. van Amstel et al.[22], multiple models of handling these disturbances are suggested.

The main problem remains however, that the predicted event time can only be approximated by the current rotational position of the leg, when the actual event has not happened yet. Gathering information that suggests at which angle the leg will touchdown or lift-off, is not in the scope of this project. For flat ground, this angle can be determined quite easily, for more rough terrain, this requires sensors that are currently not on board of the Zebro.

Therefore, the exact time of interest can only be determined when it has happened, and can only then be updated. One solution for this is implementing more sensors, which can measure the relative distance between the leg and the ground. This will not be done in this project due to the difficulty of implementation, as well as the (projected) relative small effect it will have on the performance of the system.

The three methods of modelling the disturbance on the modelled system which are defined by [22], though slightly changed, will be explained.

- **Additive Disturbance**
  Max-Plus additive disturbance is used to calculate the updated state by using Max-Plus addition:
  $$\hat{x}(k) = x(k) \oplus d(k) \tag{3-45}$$
  Where $d(k)$ are the actual lift-off and touchdown times, $x(k)$ the calculated state and $\hat{x}(k)$ the updated state. It can be seen that the resulting updated state is never earlier

than the calculated state.

- **Multiplicative Disturbance**
  For the multiplicative disturbance, the delays are modelled to be represented by a disturbance matrix, which contains the delays ($d_e(k)$) on the diagonal: $D(k) = \text{diag}(d_e(k))$.

$$\hat{x}(k) = D(k) \otimes x(k) \tag{3-46}$$

  Events that happen ahead of schedule will be propagated in this method, contrary to the case where the additive disturbance is used.

- **Change of holding times**
  The method which is most compatible with the actual physical event is to change the holding times. Because the flight time and ground time actually change, these fluctuations are reflected in the A-matrix. Again, the implementation will guarantee the delays will be processed. However, if the system is ahead of schedule, the system will not be influenced by this.

$$\hat{x}(k) = (A \oplus A_{dis}(k))x(k-1) \tag{3-47}$$

  Where $A_{dis}(k)$ is of the same dimensions as A, and incorporates the actual lift-off and touchdown times.

The implementation of the additive disturbance can be altered to incorporate the events that happen before schedule. The easiest method is to just use the actual lift-off and touchdown times as the updated state. This is the least computational heavy method to deal with the delays, and is implemented in the Zebro[22]. For simulation purposes however, the change of holding times with a random defined small time instant is the most accurate to implement, and reflects the actual situation the best.

Part II

Theoretical Background

Chapter 4

Locomotion Module

The main part of my research is to make a locomotion module capable of processing general walking instructions, in order to instruct the separate legs to walk. For the implementation, the requirements of the locomotion module need to be defined strictly. The input and output of the module should be known, the interaction between separate modules and its place in the whole system should be clear.

4-1 Module Design

When designing the module, two domains need to be combined. The theoretical part of the Max-Plus vector generation, and the mechanical implementation of controlling the leg modules need to do what is required, together. For this implementation and the combination of these two domains, the bigger picture must be understood and the functions of the module clearly defined.

4-1-1 Locomotion Overview and Project Outline

A complete locomotion system should be able to process instructions regarding a desired position and complete the correct navigation from the current location of the robot to the desired location. This process requires a great deal of steps in between the path planning and the actuation of the legs. A complete overview of the steps necessary for this process can be seen in Figure 4-1.

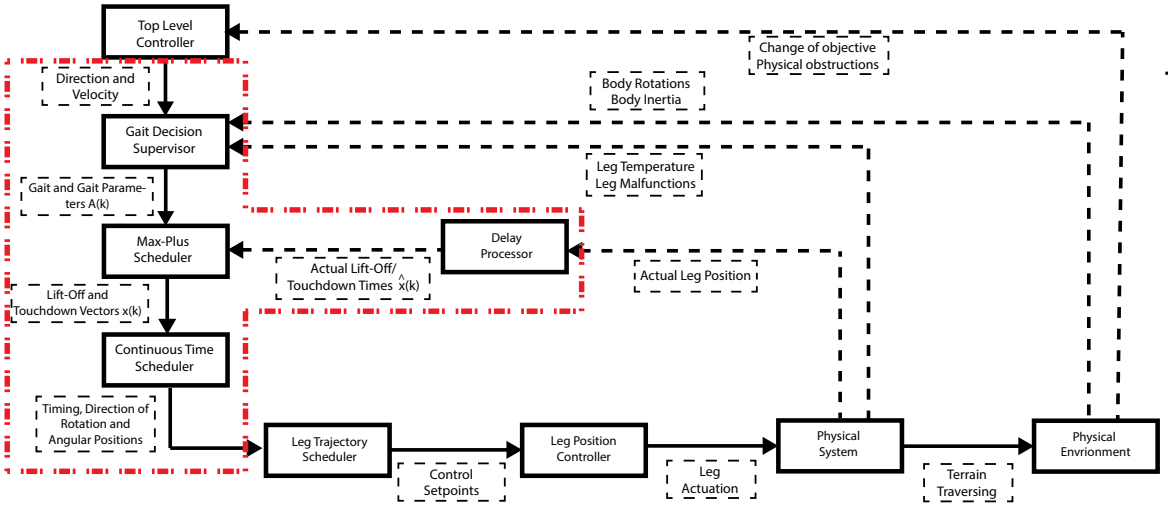


Figure 4-1: Schematic overview of the locomotion control scheme of a walking robot. The feedback loops are denoted in the dashed lines, while the control flow can be seen in the normal lines. The dash-dotted line delimits the scope of this research. This part will be called the locomotion module.

To fully explain the locomotion control scheme, the parts outside the scope of this project (which are not part of the locomotion module), will be described in this chapter. The parts of the locomotion module; the function of the **Gait Decision Supervisor** (Chapter: 4-3), **Max-Plus Scheduler** (Chapter: 4-2-1) and **Continuous Time Scheduler** (4-2-2) will be explained later in this chapter.

Top Level Controller

The locomotion control of a robot is first determined by the path planning needed to reach a certain goal. This goal or objective might also be the motion itself, for example, the following of a certain object. From this motion, or the path planning, a certain speed (or distance in an amount of time) and direction are necessary. These directions can contain turning on the spot, regular turning and walking straight. This report only focuses on the walking straight ahead, but the turning on the spot will be implemented as well. Currently, research is done to turning while walking using Max-Plus algebra. The performance of this regular, car-like turning is promising, and will be implemented later on.

Leg Trajectory Scheduler

The locomotion module supplies the leg trajectory scheduler with a specific location and time the leg needs to be somewhere. In order to process this information and to make a trajectory the legs can follow, the leg trajectory scheduler must generate a desired location for the leg at a certain time, and update this gradually to complete the step. The motor-controller of the legs then knows on which location the leg should be controlled.

Physical System

The physical system of the ZeBRo contains the actual body of the robot, along with its legs and leg motors. Using the desired trajectory supplied by the trajectory scheduler, the motor of the leg is controlled. The legs supply the feedback of the leg position, and if possible more information about the functioning of the leg, for instance the temperature of the motor in case of overheating.

Physical Environment

The physical environment is made up of the direct surroundings of the ZeBRo, most notably the walking surface and direct obstacles. Because of the influence of the ruggedness of the surface, irregularities in the height and the material composition of the surface, the effects of the environment on the body of the ZeBRo can have a big impact. Feedback of this impact can for instance be measured by an inertia measurement unit (IMU) in order to view how the system is impacted by the environment in terms of body rotations and accelerations.

4-1-2 Statement of Requirements

In order to fully determine the requirements of the module, specifications and tasks need to be determined. The locomotion module should be capable of executing these tasks sufficiently, while also being generally robust and able to recognize and solve errors where possible.
The locomotion module should be able to:

- Communicate with a higher-level instructor, either man or machine, which instructs on a direction and speed. Additionally, extra requirements regarding stability or special situations can be added.
- Use the supplied desired speed and direction to instruct the legs on how to move, using Max-Plus calculations for the timing of the locomotion. Besides walking straight ahead, the locomotion module must include turning.
- Make sure the walking is done safe and secure, and is executed without harming the robot or its surroundings.
- Use available sensors to determine the limits of locomotion, and adapt the walking style of the robot to optimize for speed or stability in an array of situations.

4-2 Locomotion Program

In order to explain how the required functions of the locomotion module will be fulfilled, the different parts of the program will be explained in this chapter.
The main locomotion program can be divided in certain functions. The main decisions will be made by the Switch Decision function, which will decide what A-matrix will be used in the further calculations. The processing of this Max-Plus matrix and the Max-Plus calculations will be handled in Section 4-2-1.

For the processing of the Max-Plus implementation to information suitable for the motors, the Continuous Time Scheduler function will be used. This will be further explained in chapter 4-2-2.
The locomotion program is programmed in C++ language, because of the limited available processing power, and the possibility for embedded implementation.

4-2-1 Max-Plus implementation

The central part of the Locomotion program is the Max-Plus implementation of the locomotion. Essentially, Chapter 3 needs to be implemented as an environment capable of updating the lift-off/touchdown vector when needed, while keeping the processing power as low as possible.

Max-Plus Gait Matrix Calculation

In order to use the gaits in the ZeBRo, the P and Q-matrices need to be generated through an algorithm. This does however require the entry of leg orders which can be used. While the amount of different gaits relies on the input of these gaits to work with, the flight, double stance and ground times can all be changed to whatever value required, which changes the speed of the ZeBRo

In order to encompass all calculations necessary for Max-Plus algebra, a toolbox for $C++$ needs to be made, which contains functions that perform Max-Plus operations. This toolbox contains the following functions (among others):

- Standard Max-Plus operations (with $a, b \in \mathbb{R}_{-\infty}$)
  - Addition ($a \oplus b$, or standard $\max(a, b)$-function)
  - Multiplication ($a \otimes b$ or standard addition $a + b$ function)
  - Power ($a^{\otimes b}$ or standard multiplication function $a \times b$)
- Vector and Matrix operations (with $\mathbf{x} \in \mathbb{R}_{-\infty}^{n\times 1}$, and $\mathbf{v}, \mathbf{w} \in \mathbb{R}_{-\infty}^{n\times n}$)
  - Matrix addition ($\mathbf{v} \oplus \mathbf{w}$)
  - Matrix multiplication ($\mathbf{v} \otimes \mathbf{x}$ or $\mathbf{v} \otimes \mathbf{w}$)
  - Max-Plus Eigenvector and Eigenvalue calculator (Eigen($\mathbf{v}$))
  - Kleene Star operation ($A_0^* = \bigoplus_{k=0}^{\infty} A^{\oplus k}$)
- Legged Locomotion Max-Plus implementation
  - Gait Decider function (Used by the Gait Decision Supervisor)
  - P- and Q-matrix builder (Used by the Gait Decision Supervisor)
  - A_0- and A_1-matrix builders (Following 3-1)
  - τ_f, τ_δ and τ_g decider (Used by the Gait Decision Supervisor)

– Transition Matrix Calculator (Following 3-3-2)

These functions are needed for the building of the Max-Plus gait matrices, while also allowing for the more difficult operations. The functions are needed as well for the Kleene Star operation, but also for calculating the Eigenvalues en Eigenvectors needed for the calculation of the transition gait matrix.
The relative heavy calculations of the changing gait matrices and transition gait matrices need to be done when other (time depended) processes are not executed.

4-2-2 Continuous Time Scheduler

When the gait matrices are determined, the lift-off/touchdown vectors can be calculated, by starting with initial lift-off/touchdown vector $\mathbf{v}_0 = \{0\}_{12\times 1}$. The next lift-off/touchdown vector can than be calculated by:

$$\mathbf{v}_{i+1} = A^{(i)} \otimes \mathbf{v}_i \qquad \text{for } i \in \mathbb{N} \tag{4-1}$$

With $A^{(i)}$ the gait matrix on instance i.
For the processing of the Max-Plus framework in order to reach continuous time leg control, multiple steps need to be taken. These are described in parts.

Max-Plus Vector Processing

In order to instruct the legs at any given time, more than one lift-off/touchdown vector needs to be in memory. For instance, two succeeding lift-off/touchdown vectors can be seen in Equations 4-2, where the first six instances represent the touchdown times of the six legs, and the last six instances represent the lift-off times.

$$\mathbf{v}_1 = \begin{bmatrix} 9.6 \\ 10 \\ 10 \\ 9.6 \\ 9.6 \\ 10 \\ 9.3 \\ 9.7 \\ 9.7 \\ 9.3 \\ 9.3 \\ 9.7 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} 10.4 \\ 10.8 \\ 10.8 \\ 10.4 \\ 10.4 \\ 10.8 \\ 10.1 \\ 10.5 \\ 10.5 \\ 10.1 \\ 10.1 \\ 10.5 \end{bmatrix} \tag{4-2}$$

As can be seen, the vectors represent a tripod gait ($\{1,4,5\} \prec \{2,3,6\}$), with $\tau_f = 0.3$ and $\tau_\delta = 0.1$. At time instant $t = 9.9$, the next event for legs 2, 3 and 6 is the touchdown at time instant $t = 10$. If the current time instant $t > max(v_1)$, the new vector v_3 will be calculated following Equation 4-1. However, if the path for the other 3 legs needs to be made in order to get them at the right place, information of the next vector is required.

Event List at $t = 9.9$			
Leg Number	Current Event Time	Next Event Time	Current Event Type
1	10.1	10.4	Lift-Off
2	10	10.5	Touchdown
3	10	10.5	Touchdown
4	10.1	10.4	Lift-Off
5	10.1	10.4	Lift-Off
6	10	10.5	Touchdown

Table 4-1: Event list of the ZeBRo continuous time scheduler at $t = 9.9$.

Therefore, there are always two vectors active within the program, and the most relevant time instances for each leg are stored within an event-list. An example of the contents of the event-list at time $t = 9.9$ can be seen in Table 4-1.

Now, when the time will go on, and $t > 10$, there are two actions to be taken: the first one is to change the instructions the event-list for leg group $\{2, 3, 6\}$. This implies updating the current event time to become the next event time, while also changing the current event type. The addition of the next event time is to increase the response time of the system. Instead of first updating the event list and calculating the new current event time, sending the new leg instructions afterwards, the information about the next event time can be send directly, with the program catching up after sending instructions.
Because of the limited computational power of the system, and the time dependence of the leg actuation, this implementation is more solid in situations requiring more calculations. For instance, gait changing, which requires a great deal of computational power, can take up precious time which can influence the execution of faster gaits.

Event-List Processing

For the sending of information to the leg modules, the decision needs to be made between sending an desired angle which needs to be reached in a desired time period, or sending a sequence of angles on which the leg consecutively sets its control set-point.
This depends for a large part on the ease of communication between the locomotion module and the separate leg modules. On the other side, the calculation power of the leg module itself is important for this decision as well. This trade off needs to be made for the optimal result, and will be made in deliberation with the leg module designers.
For the leg positioning itself, there are two angles of importance, coupled to the event type. The lift-off angle defines the desired position of the leg on the lift-off event, while the touchdown angle does the same for the touchdown event. In Section 4-3-3, the make-up of these angles is explained, and their impact on the performance of the system analysed.
If the choice is made to continuously update the control angle of each leg, a simple linear interpolation will be used between the lift-off and touchdown angles. This will ensure optimal use of the maximal speed of the leg motors in the aerial phase. Likewise, the actuation of the legs in the ground phase will keep the system in an approximately constant speed. The continuously updating of the leg control angles must be done at a high enough rate in order

to keep the leg moving smoothly.
On the other hand, when the leg decides its own path, the desired leg angle needs to be send right when the event-list needs updating. The system becomes less flexible and relies more on the leg module, but the decrease of traffic decreases the possibility of communicational problems.

4-2-3 Delay Processing

For performance evaluation, but also for safety considerations, the leg locations need to be monitored. The monitoring makes delay-handling possible, and can prevent situations resulting in falling down due to the lack of legs on the ground. Following the logic of Section 3-4, and combining the low processing power restriction with this theory, the choice is made to implement the additive disturbance.
When the event list needs updating, the leg check can be performed. Some of the legs might be in between certain events, and their location is not of the biggest importance to check (regarding time delay!). The other legs, which need to be at their destination at the current time instant, require monitoring in order to give them their next instructions.
If the positions are correct, the lift-off/touchdown vectors need no updating. When one of the legs has a small delay, the new instructions for the legs will not be send. While waiting for the delayed leg, the system will add the delay to the lift-off/touchdown vector, pauses for a short amount of time, after which the leg angle will be measured again, until the leg is at its correct place. If some delay is present, the calculation of the next vector need to be redone as well, in order to incorporate the delay throughout the system. This calculation can be seen in Equation 4-4.
As an example, for $t = 9.6$ using the lift-off/touchdown vectors of Equation 4-2, the leg angles are measured. However, leg number one has not arrived at its touchdown location. The system waits a $t_{delay} = \alpha < 0.1$ amount of time, after which the system updates the vector $\mathbf{v}_1$ with the delay vector $\mathbf{d}$ to $\hat{\mathbf{v}}_1$ (Eq:4-3). Note that the delay vector contains the actual lift-off and touchdown times, or the current time instance when the lift-off and touchdown times are in the future.

$$\hat{\mathbf{v}}_1 = \mathbf{v}_1 \oplus \mathbf{d}_1 = \begin{bmatrix} 9.6 \\ 10 \\ 10 \\ 9.6 \\ 9.6 \\ 10 \\ 9.3 \\ 9.7 \\ 9.7 \\ 9.3 \\ 9.3 \\ 9.7 \end{bmatrix} \oplus \begin{bmatrix} 9.6 \\ 10 \otimes \alpha \\ 10 \\ 9.6 \\ 9.6 \\ 10 \\ 9.3 \\ 9.7 \\ 9.7 \\ 9.3 \\ 9.3 \\ 9.7 \end{bmatrix} = \begin{bmatrix} 9.6 \\ 10 \otimes \alpha \\ 10 \\ 9.6 \\ 9.6 \\ 10 \\ 9.3 \\ 9.7 \\ 9.7 \\ 9.3 \\ 9.3 \\ 9.7 \end{bmatrix} \tag{4-3}$$

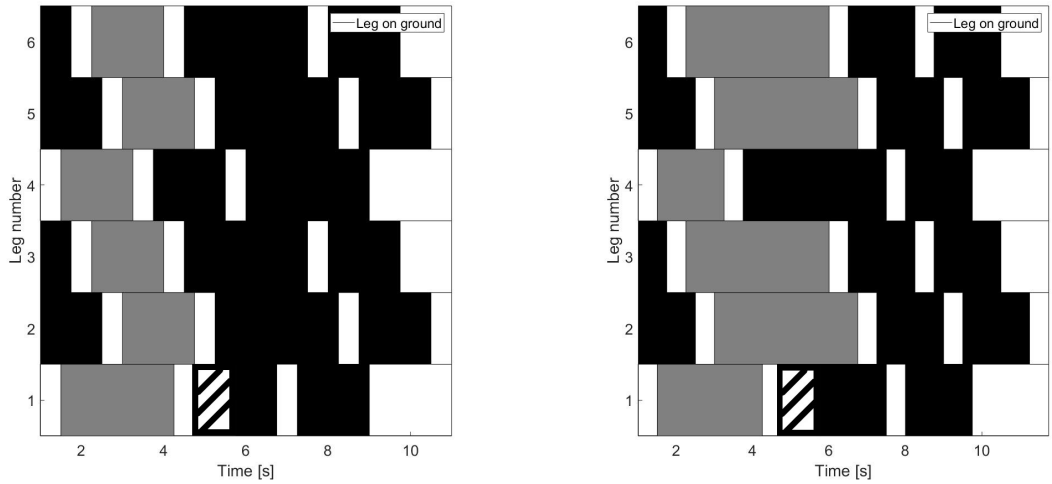
The next lift-off/touchdown vector can than be calculated using vector $\hat{\mathbf{v}}_1$ and the current gait matrix A.

$$\mathbf{v}_2 = A \otimes \hat{\mathbf{v}}_1 \tag{4-4}$$

This system does however only work on the propagation of delays onto the next state vector. There are situations possible in which the system needs faster delay handling. This requires delaying the step of the next leg-group for a certain amount of time. When a delay in the legs is present, the system can predict the minimal delay time, and can use this to calculate its effect on the planning of the other legs. This requires the usage of the A_0^*-matrix, which keeps the relations between the timing of the legs in place after the delay of other legs. The matrix A_0^* can be calculated using Equation 3-8.

$$\hat{\mathbf{v}}_1 = A_0^* \otimes \hat{\mathbf{v}}_1 \tag{4-5}$$

An example of the system response of the Max-Plus gait generation system due to delay with and without updating during the vector can be seen in 4-2. The updating of the vector happens following Equation 4-5.



(a) Gait schedule of the system without intermediate updating

(b) Gait schedule of the system with intermediate updating

Figure 4-2: System response of the Max-Plus gait generation with lift-off delay of one second, and a touchdown delay of 1.5 second on Leg 1 (the striped part). Gait: Tetrapod ($[1,4] \prec [3,6] \prec [2,5]$), $\tau_f = 0.5$, $\tau_g = 1$ and $\tau_d = 0.25$

.

The resulting gait schedule shows the delay of Leg 1, and the resulting delay of the lift-off of all other legs. As can be seen in Figure 4-2a, the gait schedule only updates the next state vector (increasing ground times after the delay), while the current state vector has not responded to the delay of the first leg. Figure 4-2b shows an increased ground time during the delay period, reacting faster and without lifting the legs before the delayed leg touches down.

For the safety of the robot, but mainly for the safety of bystanders, several layers of security need to be added to the system as a whole. The leg-module has an own safety stop, measuring the energy consumption is a method to guarantee the legs do not try to do things it is not supposed to do, destroying parts, hurting people, etc. Adding to this, when the delay of one the legs is at a certain level, some assumptions can be made about the (lack of) performance. The robot should be able to work in a safe environment, which it should not make (any) more dangerous.

4-3 Gait Decision Supervisor

The changing conditions the ZeBRo has to face influences the way the ZeBRo will walk. This will be done by changing the Max-Plus calculations, but also by changing certain parameters of the legs. The Max-Plus part of the changing gait can be denoted by a changing A-matrix. The calculation of every new lift-off/touchdown vector is than described by:

$$x(k+1) = A^{(k)}(\mu(k)) \otimes x(k) \tag{4-6}$$

With $\mu(k)$ the switching function. The A-matrix depends on the inputs of the top-level controller, but also on the input of the sensors, which will together be processed by the switching function. Alongside with the changing of the A-matrix, which encompasses the type of gait, but also the time between the steps, the lift-off and touchdown angles can also be altered. For some situations, smaller steps might be more beneficial, but for other situations, the step size can be increased, or even both angles can be changed.

4-3-1 Gait Stability

Examples of gaits for six-legged creatures or robots are the tripod and the tetrapod gaits. The leg orders of these gaits are given by:

$$\mathcal{G}_{tripod} = \{1,4,5\} \prec \{2,3,6\} \tag{4-7}$$
$$\mathcal{G}_{tetrapod} = \{1,4\} \prec \{3,6\} \prec \{2,5\} \tag{4-8}$$

These leg orders can be switched around, but the general idea about the number of legs on the ground, and make-up of the leg groups remains the same. To achieve static equilibrium after each step, at least 3 legs must be in contact with the ground. For the tetrapod gait, this means that one of these legs is redundant. However, for extreme uneven terrain, this extra leg can compensate for the loss of ground contact of one of the other legs. There remains one problem for the stability, and that is the location of legs that make ground contact. In order to keep the system stable, the centre of mass needs to lie within the triangle spanned by the supporting legs. An example of a stable system and an unstable system can be seen in Figure 4-3. For the ease of calculations , the centre of mass is placed in the middle of

the robot. In the actual ZeBRo, this centre of mass will be very near the middle, due to the double symmetry inside the robot.

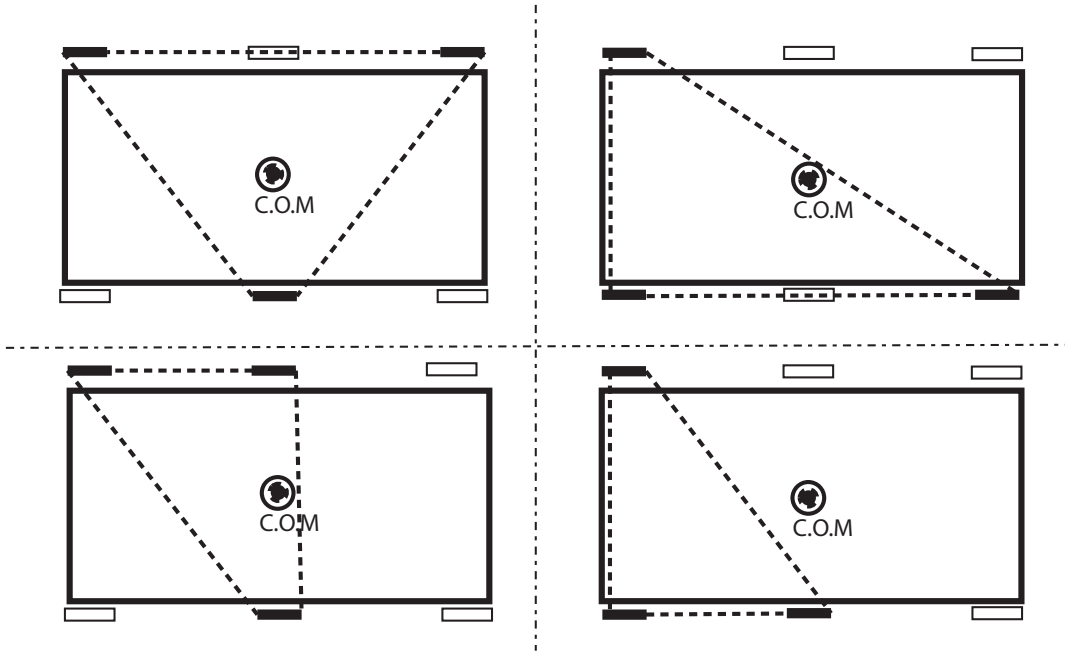


Figure 4-3: Top view of a simplified walking robot with the center of mass (C.O.M.) in the middle. The black squares are grounded legs, the white squares are legs in the air. The top-left picture shows the standard tripod stance and its support triangle, while the bottom-left shows the case that the left front leg is in the air, while the left middle leg is grounded. The right pictures depicts two scenarios, one with the right front leg on the ground and the right middle leg in the air, and vice versa. It can be seen that this last situation (bottom-right) does not encompass the center of mass in its support triangle.

So, it can be seen that in order to keep the system stable, at least one leg should be on each side. Additionally, there must be at least one leg be grounded at the back, and one at the front in order to keep the support triangle encompassing the center of mass. For the case that two of the three supporting legs are the middle legs, the six-legged robot is unstable; a small deviation in the pitch will result in a tilting and falling down of the robot. So, for maximal stability, the pentapod gait (example of a possible leg order can be seen in Equation 4-9) might be the best choice.

$$\mathcal{G}_{Pentapod} = \{1\} \prec \{2\} \prec \{3\} \prec \{4\} \prec \{5\} \prec \{6\} \tag{4-9}$$

However, the rotation speed of the legs are limited and therefore, there have to be some considerations for the maximal allowed body rotations, while still maintaining a certain speed. For some situations, the allowed body rotations might be smaller due to a special payload, while other situations just require fast traversing of terrain.
Therefore, two different situations are proposed to be tested: one situation which focuses on speed, and one situation focusing on keeping the platform as stable as possible. To quantify

these situations, and judge their performance, criteria need to be composed.
The most important judge of performance is the amount of body rotation at a certain moving speed. This can than be compared to the results of different gaits. Other criteria are the maximal speed possible using the gait, and the minimal body roll of each gait. The rotational and translational accelerations also play an important roll in the determination of the performance, in order to make sure the robot does not suffer damage.

4-3-2 Turning Gaits

For the implementation of turning, there are three methods which can be used:

1. **Turning on the spot.** By changing the direction of rotation of the legs on either the left or right side, the robot will turn on the spot to respectively left and right.

2. Creating **step size differences** between right and left legs will result in a difference in traveled distance on the left and right side.

3. Utilizing speed difference between left and right side of the robot by **implementing the research of W. Suriana**[23]. This implementation uses changes in the timing of the actuation of the legs by changing the Max-Plus matrix A to create a difference in distance covered by the legs on the two sides.

The use of turning on the spot is not only required for sharp turning in corners, but also for quickly reversing the walking direction. The other two methods are used for gentle turning. Slightly increasing and decreasing the step size on the left and right side of the robot results in slight turning in the direction where the step size is decreased.
Implementing the researched method of W. Suriana should also result in slight turning, without the negative effects of tilting due to the step size difference (Figure 4-4). The implementation of the Max-Plus method is however one with a slight delay, as the changing of leg angles can be added during walking, while the next lift-off/touchdown vector needs to wait on the end of the current.

4-3-3 Changing Leg Angles

In order to improve the walking, not only the gait, but also the further processing of the gait is important to consider. The shape of the leg, the type of movement, the time in between steps, all have influence on the walking pattern. Because the shape of the leg (C-shaped) and the type of the movement (a 1-link rotating leg) are already determined, adapting the leg angles is the next part of increasing the stability of the system.
The height of the leg attachment point should be the same for the lift-off and touchdown-position. An illustration of this situation can be seen in Figure 4-4.

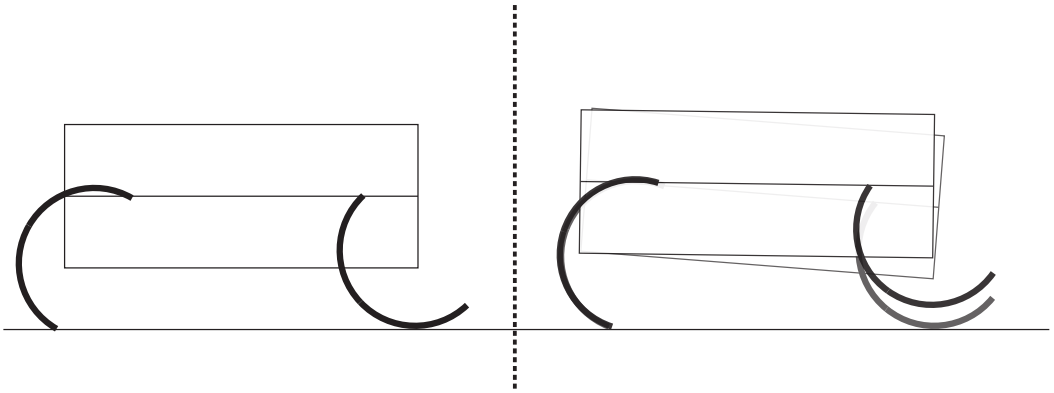


Figure 4-4: Sideview of a simplified walking robot. Due to the height difference of the legs in the right part of the picture, the robot will bounce up and down with every step, resulting in unwanted body movement

There can be varied with the height difference in the step. Increasing the angles increases the step size (and accordingly, the speed), but also the height difference within each step. The resulting consequence is an increase in required motor torque.
With an increasing height difference, the amount of vertical movement increases, which may be detrimental for a payload or even the robot itself. Limiting this in order to limit body accelerations could be required.
When the double stance time is made bigger than zero, the leg angles should change accordingly in order to keep the robot as leveled as possible.

4-3-4 V-Rep Testing

In order to make the Gait Decision Supervisor as effective as possible, there need to be tests which measure the performance of gaits, and the influence of parameters on these gaits. Testing the robot in different scenarios can give important information regarding the limits of performance and the changes in behavior along different situations.

V-Rep

For the simulations, V-Rep is used as a test environment. V-Rep is a robot simulator program, which allows for simulations containing the dynamics of robots, and allowing for using different kind of programs for the control of these simulated robots. Alongside simulating the ZeBRo itself, different environmental settings (empty planes, bumpy patches, staircases) can be implemented to see how the physics are influenced. More information about the V-Rep simulation tool can be seen on the website of Coppelia Robotics [24].

From the TU Delft, there is already a model of the ZeBRo available, which can be used for the simulations. The model, and all code in V-Rep and Matlab which makes the simulation run is adapted from the code of dr. Gabriel Lopez. There is, however a difference between the dimensions of the virtual ZeBRo and the real-life ZeBRo. Because of these differences, the

results cannot be copied 1:1 to the actual ZeBRo. Unfortunately, changing certain parameters results in the loss of performance of the simulated ZeBRo, therefore, the choice is made to not change the model. As a result, the performance of the simulated ZeBRo can not be compared 1:1 to the real ZeBRo. The tests will show qualitative results about the behavior of the system as a function of gaits, gait parameters and environment.

Parameter	V-REP ZeBRo	Real-life ZeBRo
Body Length [mm]	440	260
Body Width [mm]	240	160
Body Height [mm]	67,5	64,5
Leg Height [mm]	145	74
Mass [kg]	7	2

Table 4-2: Comparison between the simulated V-Rep ZeBRo, and the current installation of the real-life ZeBRo

As can be seen in Table 4-2, the size of the simulated ZeBRo exceeds the size of the real-life version. The increased height of the center of mass (half of the body height plus the leg height) of the simulated ZeBRo can result in the system being more susceptible to losing balance. For both ZeBRos, the center of mass is considered to be in the absolute center of the (body of the) robot.

Gaits

In order to research the effectiveness of the gaits in each situation, 5 different kind of gaits to research are proposed. To limit the amount of gaits researched, gaits with the same order, but different starting groups (e.g. $\{1,4,5\} \prec \{2,3,6\}$ and $\{2,3,6\} \prec \{1,4,5\}$) are considered the same.
The researched gaits are as follows:

1. Tripod gait: $\{1,4,5\} \prec \{2,3,6\}$
2. Tetrapod gait: $\{1,4\} \prec \{3,6\} \prec \{2,5\}$
3. Metapod gait: $\{1\} \prec \{4,5\} \prec \{2,3\} \prec \{6\}$
4. Pentapod gait: $\{1\} \prec \{6\} \prec \{2\} \prec \{5\} \prec \{3\} \prec \{4\}$
5. Climb gait 1: $\{1,2\} \prec \{3,4\} \prec \{5,6\}$
6. Climb gait 2: $\{5,6\} \prec \{3,4\} \prec \{1,2\}$

For the tripod, tetrapod and pentapod gaits, the reason of implementation is quite simple; varying the amount of legs on the ground while maintaining an leg order which varies the actuated leg in a logical order (maintaining the support triangle). The climb gait is added in order to validate its behavior on stairs. To investigate which climbing gait has better results, the order in which the legs recirculate is slightly altered.
The metapod (name is chosen because it is between (Greek: Metá) the tetrapod and pentapod gait) gait however, has a different amount of legs in the different leg groups. This is done to

keep the system from having two legs of the front and back rows in the recirculation. This might improve the stability with regards to the tetrapod gait, while it may be faster than the pentapod gait.

Parameters

Two types of parameters need to be changed: the Max-Plus implementation flight, ground and double stance time, and the leg angles. For the leg angles, following Section 4-3-3, is a matter of increasing the distance covered in the ground phase by increasing the angle of liftoff and touch-down with respect to the normal of the ground. For the Max-Plus parameters, it is chosen to research six different sets of flight, ground and double stance times. These can be seen in Table 4-3. It is chosen to set the ground time the same as the flight time, so the gaits with no double stance time actually show the desired behavior. This also applies to the gaits with double stance time, where the actual double stance time is determined by the ground time if this exceeds a gait determined limit.

Time Parameter	τ_f	τ_g	τ_δ
Fast Set 1	0.5	0.5	0.0
Fast Set 2	0.5	0.5	0.2
Normal Set 1	0.8	0.8	0.0
Normal Set 2	0.8	0.8	0.3
Slow Set 1	1.1	1.1	0.0
Slow Set 2	1.1	1.1	0.5

Table 4-3: Overview of the different Max-Plus parameter sets

For the leg angles, steps of 10 degrees are proposed, ranging from 10 degrees to 30 degrees. The angles tested can be seen in Table 4-4.

Leg Angles	Touchdown α_{td}	Lift-off α_{lo}
Angle Set 1	10°	-10°
Angle Set 2	20°	-20°
Angle Set 3	30°	-30°
Angle Set 4	40°	-40°

Table 4-4: Overview of the different Max-Plus parameter sets

Terrains

The tests will be performed on 4 different type of terrains:

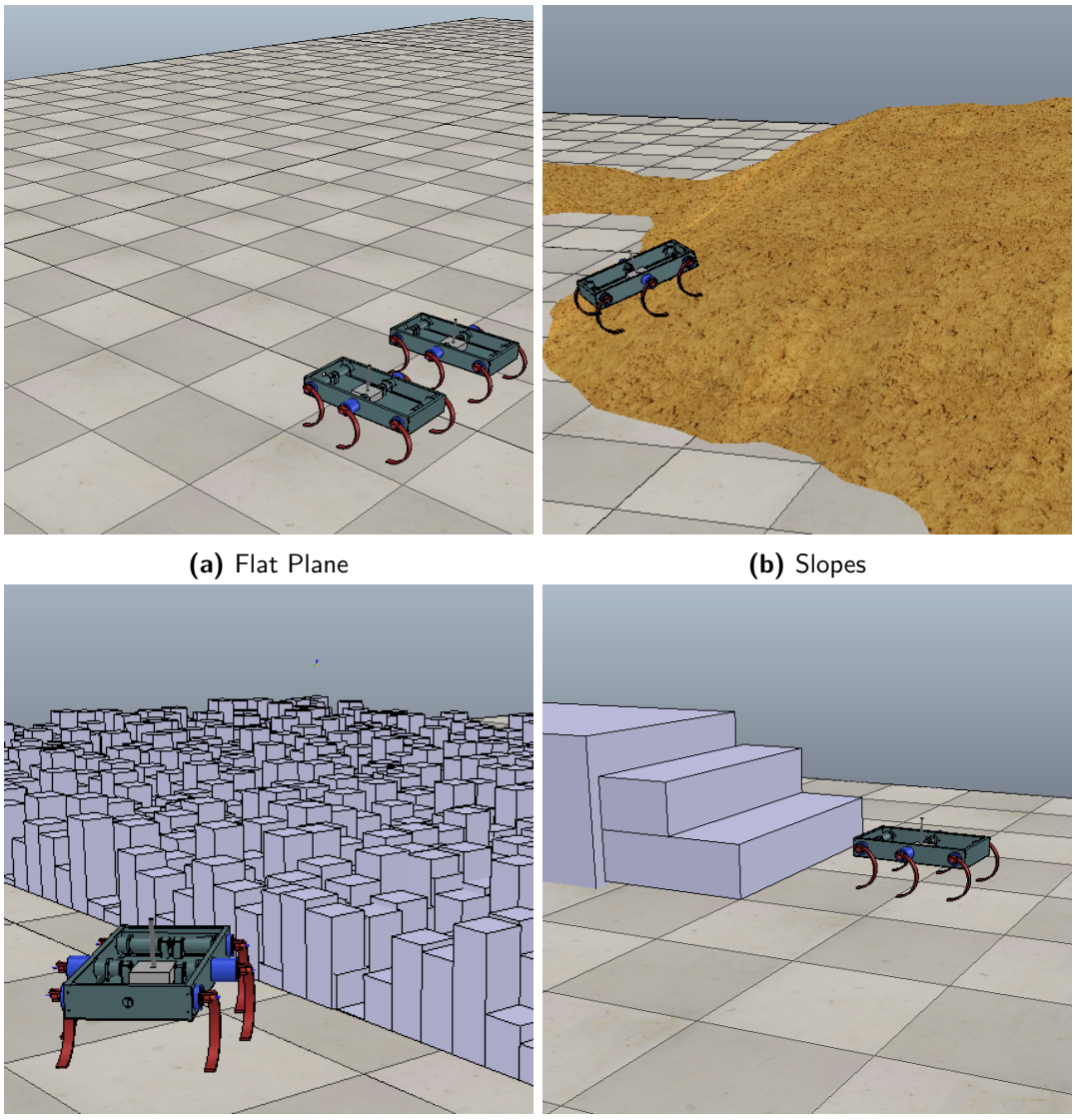
Flat Plane A leveled surface with a flat ground

Slopes Surface with sloping hills (max. 30° steep)

Spikes Spiked surface with a very uneven ground (ranging from 1 cm to 25 cm high spikes, random distribution)

Stairs Regular stairs with steps appropriately sized for the legs of the ZeBRo ($\sim$ 15 cm in height)

The surfaces can be seen in Figure 4-5.



(a) Flat Plane **(b)** Slopes

(c) Spikes **(d)** Stairs

Figure 4-5: The different surfaces on which the V-REP ZeBRo will be tested.

4-4 Results V-REP Tests

In order to classify the results from all the tests, there are three major criteria the system will be judged on. First and foremost, the speed of traversing the terrain will be researched. Body movement is also recorded, in order to determine the vertical and sideways movement of the body during walking. The body rotations are tracked to measure the stability of the

system. Every test is performed 5 times for 20 seconds.

4-4-1 Gait Performance On Flat Terrain

The most interesting part of the tests is the comparison of different gaits and how they perform on each of the surfaces. To start the comparison, first the gaits are tested on a flat plane. The Max-Plus τ_f and τ_δ and the leg angles are kept constant at: $\tau_f = 0.5$, $\tau_\delta = 0.1$, angle of lift-off $\alpha_{lo} = -30°$, angle of touchdown $\alpha_{td} = 30°$.
The difference in distance traveled by the robot on flat terrain using different gaits can be seen in Figure 4-6. The results of all 5 tests are very similar to the results of the test portrayed.

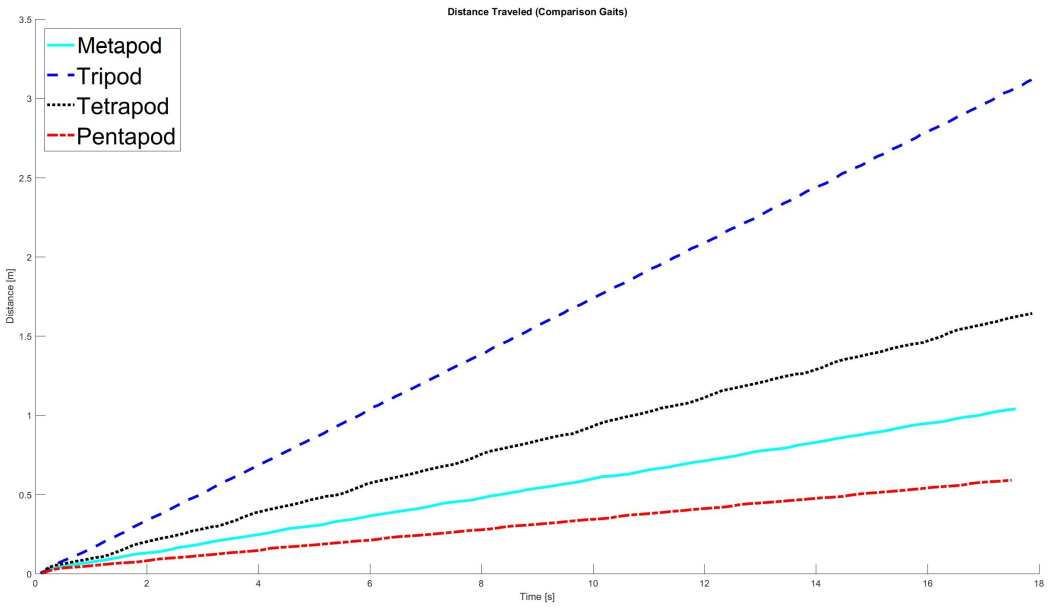
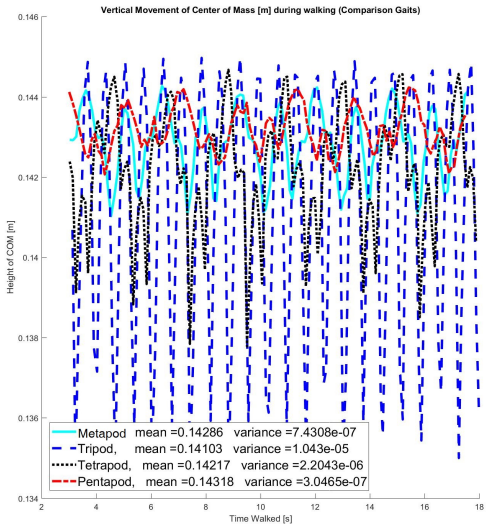
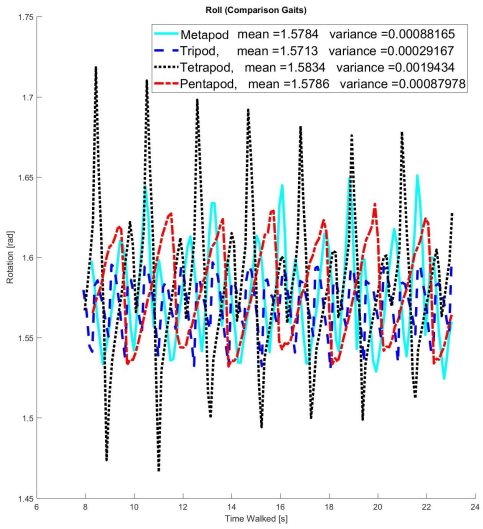


Figure 4-6: Comparison of distance traveled with different gaits on flat terrain

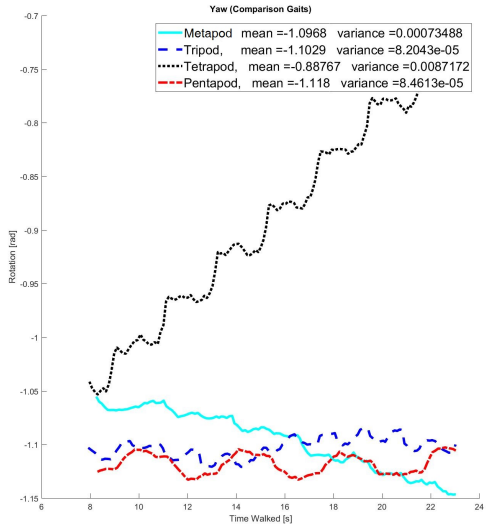
As can be seen, the movement of the ZeBRo is a lot faster using the tripod gait than using any other gait, which could be expected. Assuming every step size is independent of the gait, the ratio of the seperate gaits traveled distance can be compared using the eigenvectors of the A-matrices of the gaits. However, the actuation of the legs when different groups are on the ground results in a smaller distance traveled, due to partly overlapping paths.
The vertical movement and body rotations of the ZeBRo on flat terrain can be seen in Figure 4-7a.



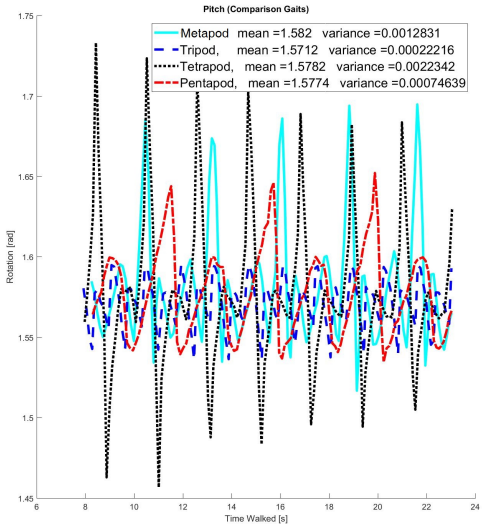
(a) Vertical movement



(b) Body Roll



(c) Body Yaw



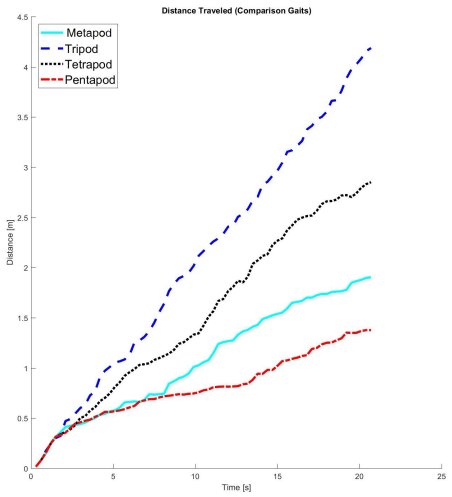
(d) Body Pitch

Figure 4-7: Comparison of body movement and rotations using different gaits ($\tau_f = 0.5$, $\tau_\delta = 0.1$, $\alpha_{lo} = -30°$, $\alpha_{td} = 30°$) on a flat surface

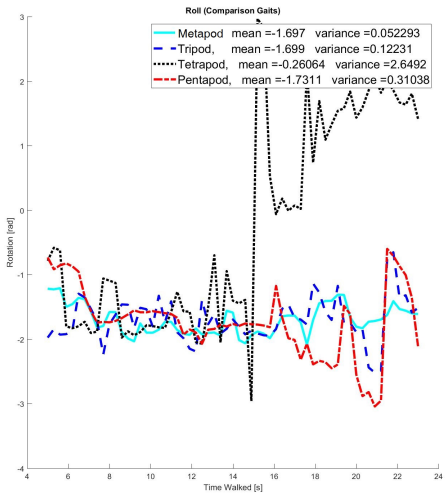
As can be expected, the bigger amount of leg groups, the more constant the vertical height of the ZeBRo. Comparing the body rotations however, shows that the tripod gait scores the best on rotational stability. Furthermore, interesting behaviour can be seen when watching the yaw progression of the tetrapod gait, which seems to be slightly turning. The rotational stability of the tetrapod and metapod gait show some difficulties on the leveled terrain.

4-4-2 Gait Performance On Difficult Terrain

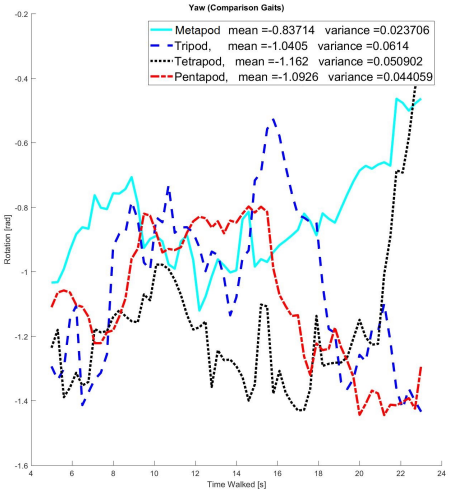
Comparing the same gaits on the bumpy terrain shows about the same results, with comparable body rotations and distance ratio between the gaits. More interesting results to show, are the ones on the spiked floor, which were not completed successfully by all gaits. An example of the distance traveled and the body rotations can be seen in Figure 4-8. The vertical body movement is discarded due to the alternating height of the floor.
An interesting moment can be noticed: the sudden body roll and pitch change of the robot using the tetrapod gait (which indicates the robot falling over).



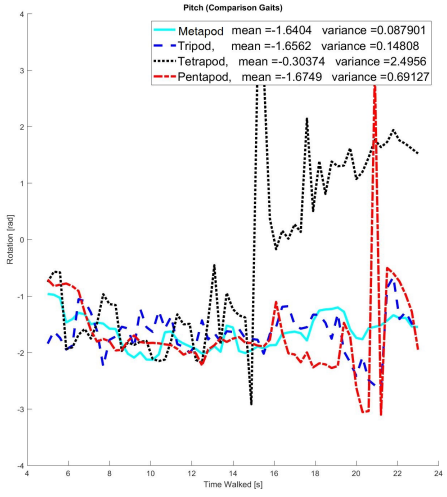
(a) Distance travelled



(b) Body Roll



(c) Body Yaw



(d) Body Pitch

Figure 4-8: Comparison of body movement and rotations using different gaits ($\tau_f = 0.5$, $\tau_\delta = 0.1$, $\alpha_{lo} = -30°$, $\alpha_{td} = 30°$) on a spiked surface

The spiked surface had some interesting results regarding the performance of the robot, as several simulations resulted in flipping over, which qualifies as a failed test. When the results of all tests are filtered, the amount of failed tests regarding the gaits, step size, double stance time and overall speed can be reviewed. It has to be noted that the distance covered has a big impact on the test results, as the amount of spikes encountered is simply higher than the spikes encountered while walking with a lower speed. Even while keeping this in mind, some conclusions can be made about the performance of the system.
The results of these tests can be seen in Tables 4-5 to 4-8.

Failed Test Percentage	Test 1	Test 2	Test 3	Test 4	Test 5	All Tests	Average Distance
Pentapod	21%	50%	33%	25%	25%	31±12%	0.84 [m]
Metapod	29%	42%	29%	38%	38%	35±6%	1.30 [m]
Tetrapod	54%	67%	63%	67%	50%	60±8%	1.53 [m]
Tripod	67%	71%	50%	46%	54%	58±11%	2.60 [m]

Table 4-5: Failed test percentage on the spiked surface divided per gait, with the average distance of the **successful** runs.

Failed Test Percentage	Test 1	Test 2	Test 3	Test 4	Test 5	All Tests	Average Distance
$\tau_\delta \neq 0$	38%	52%	44%	25%	38%	39±10%	1.28 [m]
$\tau_\delta = 0$	48%	63%	44%	63%	46%	53±9%	1.67 [m]

Table 4-6: Failed test percentage on the spiked surface divided over walking with and without double stance time, with the average distance of the **successful** runs.

Failed Test Percentage	Test 1	Test 2	Test 3	Test 4	Test 5	All Tests	Average Distance
Angle Set 1	21%	54%	38%	38%	29%	36±11%	0.84 [m]
Angle Set 2	46%	63%	46%	38%	50%	48±8%	1.40 [m]
Angle Set 3	38%	54%	29%	46%	38%	41±9%	1.40 [m]
Angle Set 4	67%	58%	63%	54%	50%	58±7%	1.67 [m]

Table 4-7: Failed test percentage divided on the spiked surface over the different lift-off/touchdown angle sets, with the average distance of the **successful** runs.

Failed Test Percentage	Test 1	Test 2	Test 3	Test 4	Test 5	All Tests	Average Distance
$\tau_f = \tau_g = 1.1$	31%	47%	28%	22%	34%	33±9%	1.13 [m]
$\tau_f = \tau_g = 0.8$	28%	59%	47%	47%	28%	42±12%	1.53 [m]
$\tau_f = \tau_g = 0.5$	69%	66%	56%	63%	63%	63±5%	1.93 [m]

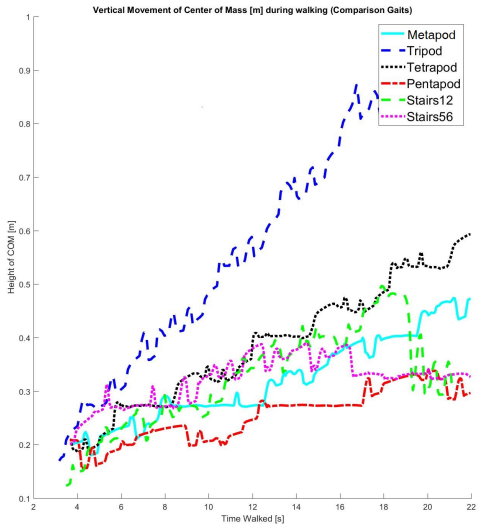
Table 4-8: Failed test percentage on the spiked surface divided for different values of $\tau_f = \tau_g$, with the average distance of the **successful** runs.

The resulting percentages show a clear overview of the success rate of the different settings on the spiked environment. It has to be noted that this type of environment, and especially the dimensions of the spikes, were extremely demanding for the robot to complete. On comparison, the unsuccessful runs on the bumpy environment did not exceed the 5%.

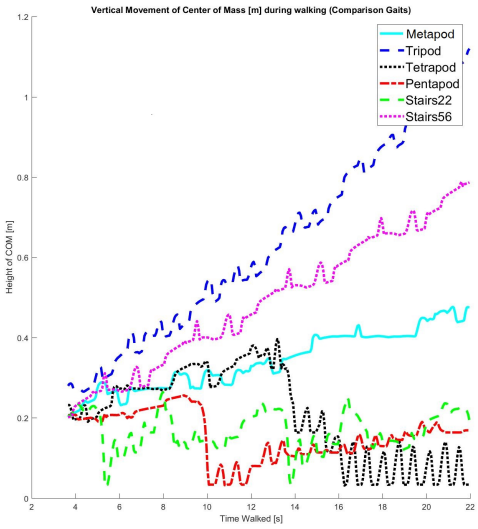
Considerable differences with respect to the flat surface can be seen on multiple points, for example on the distance travelled using the different angle sets, but also on the performance of the different gaits.
For the stairs climbing, the ZeBRo encountered a terrain which required heavy-duty performance. Like the spiked surface, the stairs were supposed to be unforgiving in nature, and might have been somewhat difficult for the ZeBRo to walk on. However, due to the rough selection, the performance of the gaits can be reviewed well.

4-4-3 Gait Performance on Stairs

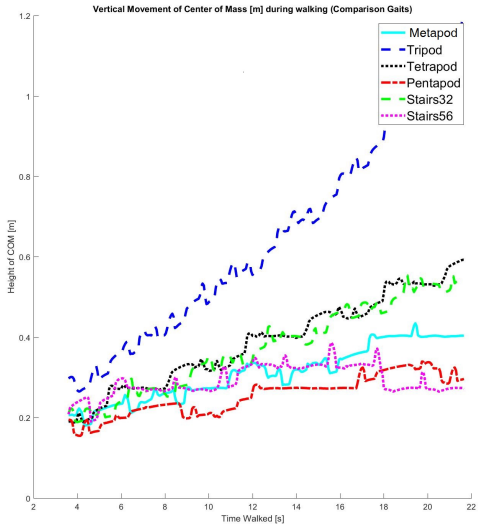
For the same gait parameters used in Figure 4-7, the climbing of the stairs in 4 test can be seen. Note that the stairs did not change, but the starting coordinates did (slightly). The body rotations are not portrayed, as they are not clear to read due to all the different movements during the climbing of the stairs.



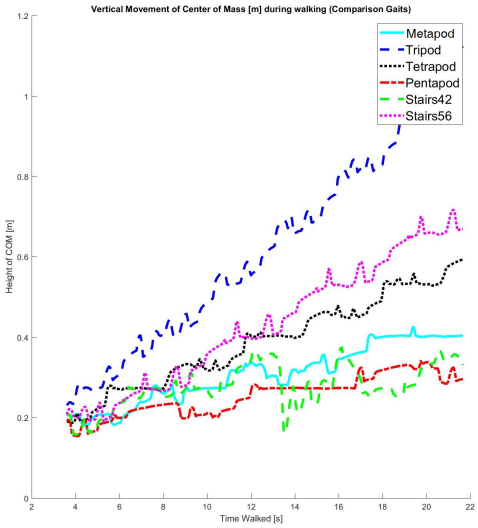
(a) Vertical movement of the ZeBRo, Stairs test 1



(b) Vertical movement of the ZeBRo, Stairs test 2



(c) Vertical movement of the ZeBRo, Stairs test 3



(d) Vertical movement of the ZeBRo, Stairs test 4

Figure 4-9: Comparison of the vertical movement of the ZeBRo using different gaits ($\tau_f = 0.5$, $\tau_\delta = 0.1$, $\alpha_{lo} = -30°$, $\alpha_{td} = 30°$) on the stairs

To show the difference in performance, not just between the gaits, but also the performance of the gaits in different tests, the results of 4 different stair climbing test can be seen in Figure 4-9. From these four tests, it can be seen the performance of the stair climbing gaits is not great. Again, it can be seen that several times during test 2, the ZeBRo manages to fall of the stairs. This could be due to a big jaw change, due to the slightly altered starting conditions.

4-4-4 Gait and Parameter Selection

Using the data of the V-Rep tests, the selection for situation-specific gaits can be determined. Separating the different situations and watching the performance on the environments show that the tripod gait is the most versatile, as well as the fastest gait. Changing of the gait is most important when encountering extreme uneven terrain. The gait which can be changed to could be the pentapod or the metapod gait.
Other interesting findings are the performance of slower gaits on the difficult terrain, the increase of the success rate with addition of double stance time, and the success rate of the Angle Set 3, which shows good results on every terrain.
The moment of gait change can be found in Figures 4-7 and 4-8, as the tripod gait shows very low body roll on even terrain. On the more uneven terrain however, the body rotations can increase drastically. It makes sense to incorporate these results, and allow the gait to change in the situation of difficult terrain. Classifying when the terrain changes, requires measuring when body rotations exceeds certain values.
Deciding for what values the gaits needs to be changed can be done by determining the body roll on the easier planes, and resetting the gait back to the tripod after a certain time of limited body roll.
A pseudo-code example of this implementation can be seen in Algorithm 1.

```
Data: Gait Change Algorithm
Result: Allows for changing the gait from tripod to a slower gait when the body
        rotations exceed certain values

initialization ZeBRo-processes ;
initialization Tripod A-Matrix ;
start Walking Modus ;
while In Walking Modus do
    Calculate Max-Plus lift-off/touchdown vector using A-matrix;
    Readout Body Rotations;
    if Body Rotations > Max-Allowable Rotation then
        A-Matrix = Metapod/tetrapod Gait;
    else if Body Rotations(Last n Seconds <Max-Allowable Rotation) then
        A-Matrix = Tripod Gait;
end
```

Algorithm 1: Gait change pseudo-code

Implementing this code in the simulation requires determining two variables: the *Max-Allowable Rotation*, and the value for n. n is defined as the variable which decides how long after receiving the last time instance exceeding the $MaxAllowable\ Rotation$, the robot resets to the tripod gait.

4-4-5 Adaptive Gait Implementation

The choice for these variables is somewhat arbitrarily, as the system will not gain a massive stability increase when n is changed from 1.1 second to 1 or vice versa. More important is the choice for the allowed body rotation before the system shifts to a more conservative gait. For now, this is chosen to be the metapod gait. Testing with the real system might prove which gait (pentapod or metapod) is more capable.
The body rotation variable needs to be well outside the range of normal, straight floor use. Ideally, slightly rugged terrain is also still conquered using the tripod gait. Therefore, the delta angle of the Max-Allowable rotation will be slightly larger than the angles the ZeBRo needs to face on the bumpy terrain, which is about 30°. However, the determination of this angle should be seen in qualitative context, as different robots undergo different movements due to the surface they walk on.

Chapter 5

Gaits With Aerial Phases

To define the difference between static walking and dynamic walking, the easiest explanation is to look at us, humans. While walking and running have the same order of legs to take off and touch down, there are some major differences in speed and energy consumption. Regular walking has moments with both legs on the ground, and moments with one leg on the ground. When both legs are on the ground, the stability of your body increases, especially in rugged terrain. Running however, has moments where both legs are in the air, so the body is not in contact with the ground. While this is not preferred on difficult terrain, the maximal speed of running is a lot higher than the speed of walking.

Besides running and walking, which both feature the leg order: (leg $(1) \rightarrow$ leg (2)), other gaits are possible, like hopping (leg $(1,2) \rightarrow$ leg $(1,2)$) and skipping (leg $(1) \rightarrow$ leg (1)), but also more complex gaits like the triple jump (leg $(1) \rightarrow$ leg $(1) \rightarrow$ leg $(2) \rightarrow$ leg $(1,2)$).

With the current implementation of the Max-Plus algebra, these gaits are not possible. However, when the gait construction is extended, much more gaits can be constructed using the Max-Plus system. In section 5-1, the construction of strictly running gaits is done. Some gaits, like the gallop gait of horses, contains both aerial and double stance phases. Therefore, the section 5-2 contains the method for constructing a much wider array of gaits, extending the method for gait generation by allowing for more synchronization options in the creation of the walking patterns.

5-1 Max-Plus Gait Generation with Aerial Time

For the Max-Plus implementation of the dynamic gait, several aspects of the Q and P matrices need to be considered. The main aspects of successful implementation are the transition from walking gaits to dynamic and the frequency at which the different gaits take place. In order to use Max-Plus for dynamic gaits, some factors come into play that were not considered in the generation of the walking gaits with double stance times.
If the aerial phases are implemented, this will imply changes for the times in the succeeding

lift-off and touchdown vectors. For example, some lift-off times of vector $x(k+1)$ can be smaller (and happen earlier) than the touchdown times of vector $x(k)$. This has consequences for the generation of the A-matrix, but also for the implementation on the ZeBRo and further processing of the leg instructions.

5-1-1 Constructing the A-matrix

For the implementation of the running gaits, the construction of A-matrix needs to be considered again. In Chapter 3 can be seen that the A_1-matrix contains the coupling between the touchdown times of $x(k)$ and the lift-off times of $x(k+1)$, and that the A_0 contains the coupling between the lift-off times of $x(k+1)$ and its touchdown times.
For this system, the synchronization will happen during the double stance time. The synchronization is done every instance of $x(k)$.
Now, for a running gait, we want the synchronization to happen during the aerial phase time (τ_a) instead of during the double stance phase. This would require changes in the method of constructing the A_0 and A_1 matrix, but will solve the problem of non-chronological succeeding lift-off/touchdown vectors (i.e. $\max(x(k)) > \min(x(k+1))$. Another method, proposed by M. Shahbazi Aghbelagh [25], is to simply switch the meaning of the first half of the vector from being touchdown moments to being lift-off moments and vice versa. However, altering the A_0 and A_1 matrices will have the same results, without changing the make-up of the lift-off/touchdown vector. This method requires less shifting in methods of controlling the timing, as the changing of gaits from static to dynamic will be the same as changing one static gait for another; by changing the A-matrix.

We start again using the first assumption that the legs will lift off after they have been on the ground for a certain time (τ_g). After this, the legs will stay in the air for some time (τ_f), and finally will touch down. Contrary to Equations 3-1 and 3-2, we will invert the order of touchdown and lift-off. This results in Equations 5-1-5-2. For a six-legged system, $i = 1...6$.

$$t_i(k+1) = l_i(k) \otimes \tau_f \tag{5-1}$$
$$l_i(k+1) = t_i(k+1) \otimes \tau_g \tag{5-2}$$

Now, if we again want to implement the wave gait ($1 \to 2 \to 3 \to 4 \to 5 \to 6$) as an example, we have additional terms to indicate the order in which the legs move. This time however, we want two consecutive leg groups in the air at the same time for τ_a seconds

$$t_i(k+1) = \begin{cases} l_i(k) \otimes \tau_f \oplus l_6(k) \otimes \tau_a & \text{for } i = 1 \\ l_i(k) \otimes \tau_f & \text{for } i = 2, ..., 6 \end{cases} \tag{5-3}$$

$$l_i(k+1) = \begin{cases} t_i(k+1) \otimes \tau_g & \text{for } i = 1 \\ t_i(k+1) \otimes \tau_g \oplus l_i(k+1) \otimes \tau_a & \text{for } i = 2, ..., 6 \end{cases} \tag{5-4}$$

Again, rewriting this results in the following Max-Plus linear system:

$$x(k+1) = \left(\begin{array}{c|c} \mathcal{E}_{6\times6} & P \\ \hline \tau_g \otimes E_{6\times6} & \mathcal{E}_{6\times6} \end{array}\right) \otimes x(k+1) \oplus \left(\begin{array}{c|c} E_{6\times6} & \tau_f \otimes E_{6\times6} \oplus Q \\ \hline \mathcal{E}_{6\times6} & E_{6\times6} \end{array}\right) \otimes x(k) \tag{5-5}$$

The generation of P and Q matrices remains the same, as the same order is still being maintained, but implemented differently. The only difference is the replacement of the double stance time τ_d by the double air time τ_a. The construction of these matrices can be seen in Section 3.2.

A comparison between the two methods of making the A-matrix can be seen in Appendix A-3. An important result of generating matrices using this method is the switching of the order of touchdown and lift-off inside each event vector $x(k)$. For the walking gait, the last time instant of touchdown is always larger than the last time instant of lift-off, for the running gait, vice versa. For implementation purposes, this order switching can be solved by making an event list (Section 4-2-2), which updates the next touchdown and lift-off times, regardless of their order.

5-1-2 Gait Schedules

In the standard walking of six-legged animals, the tripod gait is the gait containing the least legs on the ground while still maintaining a constant state of static balance. However, for the running implementation, this changes the situation. This gait is used as an example of how Max-Plus can be used for running gaits.
For the Max-Plus lift-off/touchdown vector generation, the Max-Plus matrix of the running tripod gait needs to be constructed. The P and Q matrices are again determined using the method supplied in Chapter 3-2. This results in the following matrices:

$$P = \begin{bmatrix} \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \tau_\alpha & \epsilon & \epsilon & \tau_\alpha & \tau_\alpha & \epsilon \\ \tau_\alpha & \epsilon & \epsilon & \tau_\alpha & \tau_\alpha & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \tau_\alpha & \epsilon & \epsilon & \tau_\alpha & \tau_\alpha & \epsilon \end{bmatrix}, \qquad Q = \begin{bmatrix} \epsilon & \tau_\alpha & \tau_\alpha & \epsilon & \epsilon & \tau_\alpha \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \tau_\alpha & \tau_\alpha & \epsilon & \epsilon & \tau_\alpha \\ \epsilon & \tau_\alpha & \tau_\alpha & \epsilon & \epsilon & \tau_\alpha \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix} \tag{5-6}$$

Then, Equation 5-5 and 3-8 to 3-12 can be followed to construct the A-matrix. This results in the following gait schedule of Figure 5-1

For the tetrapod gait, the aerial phases concern the biggest part of the locomotion, as the leg groups wait for each other before touching down and lifting off again. The gait schedule can be seen in Figure 5-2.

5-1-3 Delay Handling

When legs are hindered to reach their position on time, the Max-Plus algebra automatically shifts the next events forward. For the static implementation, this poses no problem, as the

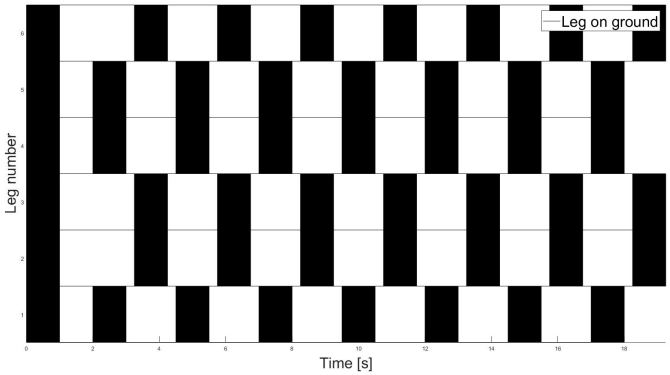


Figure 5-1: Schematic overview of the state of the legs in the running tripod gait. The black squares represent the time the legs are on the ground, while the white space represents the time the legs are in the air (with $\tau_f = \tau_g = 1$, τ_α=0.25)

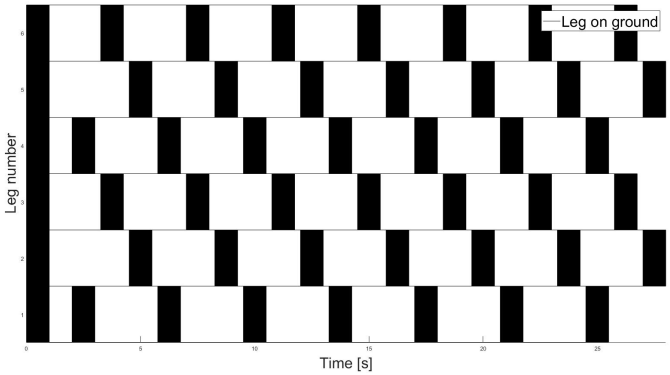


Figure 5-2: Schematic overview of the state of the legs in the running tetrapod gait. The black squares represent the time the legs are on the ground, while the white space represents the time the legs are in the air (with $\tau_f = \tau_g = 1$, τ_α=0.25), and yes, the lines look like they are diagonal instead of horizontal!

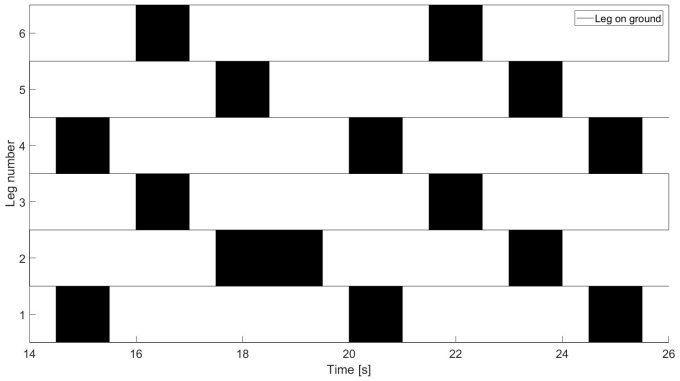


Figure 5-3: Delay of leg 2 during the running tetrapod gait. As can be seen, the touchdown events of the next leg-group are shifted further in time.

stability is not affected. For the dynamic implementation, this can result in problems, as these delays can shift moments of touchdown forward. An example of a delayed lift-off can be seen in Figure 5-3.

The main purpose of the implementation of the automatic delay handling in Max-Plus is to hinder the system from getting in unwanted states. Implementing this delay handling results in unwanted behavior, and should only be used to delay a legs lift-off, not their touchdown event. When a choice can be made, the ground time should be extended. This can be done by making the gaits synchronize on their lift-off times, instead of their touchdown times.

5-1-4 Static to Dynamic Transition

The transition of walking gaits to running gaits requires understanding the dynamics of the walking system. However, the transition from static to dynamic gaits for the Max-Plus gait generation requires the implementation of a transition in order to correctly switch. This is due to the swapping of the order of the lift-offs and touchdowns with respect to the static gait generation.
The other way around might be less important, but still, the A-matrix for both transitions needs to be determined. The symbols $\mathbb{G}_s$ and $\mathbb{G}_d$ are the static and dynamic implementations of the respective gaits. For walking to running, the A-matrix sequence becomes, where $\tau_{x\mathbb{G}_s}$ and $\tau_{x\mathbb{G}_d}$ are the time parameters for respectively the static and dynamic implementations of their gaits for $\forall x = d, g, t$:

$$A(\mathbb{G}_s, \tau_{f\mathbb{G}s}, \tau_{g\mathbb{G}s}, \tau_{d\mathbb{G}s}) \tag{5-7}$$

$$A(\text{Static-to-Dynamic Transition}) \tag{5-8}$$

$$A(\mathbb{G}_d, \tau_{f\mathbb{G}d}, \tau_{g\mathbb{G}d}, \tau_{d\mathbb{G}d}) \tag{5-9}$$

And vice versa:

$$A(\mathbb{G}_d, \tau_{f\mathbb{G}d}, \tau_{g\mathbb{G}d}, \tau_{d\mathbb{G}d}) \tag{5-10}$$
$$A(\text{Dynamic-to-Static Transition}) \tag{5-11}$$
$$A(\mathbb{G}_s, \tau_{f\mathbb{G}s}, \tau_{g\mathbb{G}s}, \tau_{d\mathbb{G}s}) \tag{5-12}$$

Without a transition matrix, the transition happens non-optimally. This can be seen in Figure 5-4. The static part of the walking is completed, but the dynamic walking has not started yet to allow for dynamic movement. To resolve this problem, it is necessary to start all gaits with the lift-off of the first leg-group.

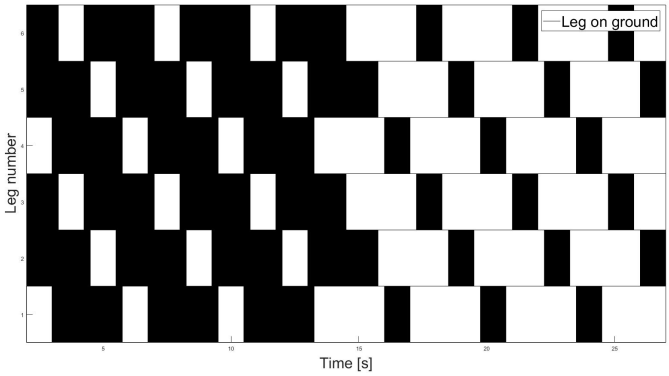


Figure 5-4: Schematic overview of the state of the legs transitioning from the walking tetrapod the running tetrapod gait. The black squares represent the time the legs are on the ground, while the white space represents the time the legs are in the air (with $\tau_f = \tau_g = 1$, τ_α=0.25). The aerial times start before the rest of the dynamic behaviour is in place.

To solve the drawbacks regarding transitions and delays, and complete the list of possible gaits which cannot be constructed using the method suggested in this chapter, the next section will add the insights of this section and the regular Max-Plus gait generation together to create a full gait generation method.

5-2 Max-Plus Complete Gait Generation

The use of Max-Plus as the base for gait construction has interesting upsides. The automatic delay handling, easy gait construction and good documented gait transition possibilities allow for use in legged robots, no matter the amount of legs. However, as can be seen in Chapter 5-1, the delay handling can also have downsides. It is therefore chosen to make a total gait construction method which relies on the synchronization of the lift-off events of the leg. The most important reason for choosing the lift-off events instead of the touchdown events, is the overall stability of the robots, as this is (most of the time) negatively influenced by delaying the touchdown events, not by delaying lift-off events.
For the construction of the gaits in this section, the physics of the implemented system are

not taken in consideration, merely the Max-Plus method of defining them. We start again with the Max-Plus linear discrete event system:

$$x(k+1) = A_0 \otimes x(k+1) \oplus A_1 \otimes x(k) \tag{5-13}$$

Because the lift-off moments are the moments of synchronization, all these moments are either coupled to a touchdown or a lift-off. The touchdowns however, are only coupled to their preceding lift-offs. Every gait always starts with the lift-off of the first leg-group.

Using the system of lift-off synchronization, the A-matrices can be constructed following Equations 5-14 & 5-15.

$$A_0 = \left(\begin{array}{c|c} \mathcal{E}_{n\times n} & T_f \\ \hline P_t & P_l \end{array}\right) \tag{5-14}$$

$$A_1 = \left(\begin{array}{c|c} E_{n\times n} & \mathcal{E}_{n\times n} \\ \hline T_g \oplus Q_t & E_{n\times n} \oplus Q_l \end{array}\right) \tag{5-15}$$

It is important to notice the change made to the A_0 matrix, and whether this impacts the possibility of using the Kleene Star operation (creating A_0^*). Because the P_l matrix contains its non-ϵ instances on the same places as the P_t matrix, and always has an epsilon on its diagonal, the A_0^*-matrix can still be constructed:

Theorem 3.20 of [26]: *If A of the precedence graph $\mathcal{G}(A)$ has no circuit with positive weight, then*

$$A^* = E \oplus A \oplus A^{\otimes 2} \oplus ... \oplus A^{\otimes n-1} \tag{5-16}$$

where n is the dimension of A.

Due to the way the matrix and its sub-matrices are build, there is no circuit in the A_0-matrix. This can be observed when calculating $P_l^{\otimes b}$, with $b \in \mathcal{N}$ the amount of leg-groups or, because the instances of P_t are always smaller than P_l, any combination $P_l^{\otimes c} \otimes P_t^{\otimes d}$, with $c, d \in \mathcal{N}$ and $c + d = b$. The resulting matrix will always devolve to the $\mathcal{E}$-matrix, due to the lack of circuits in the P-matrices.

In order to generate a full array of gaits, the standard system of Max-Plus static gait generation is taken, and complemented by the P_l and Q_l matrices, which contain the lift-off to lift-off synchronization. The name of the P and Q matrices is changed to P_t and Q_t to indicate the touchdown to lift-off synchronization. The gallop gait will be used as an example for the construction of the A-matrix [27], due to the difference in construction with respect to the standard gait construction method developed by [4].

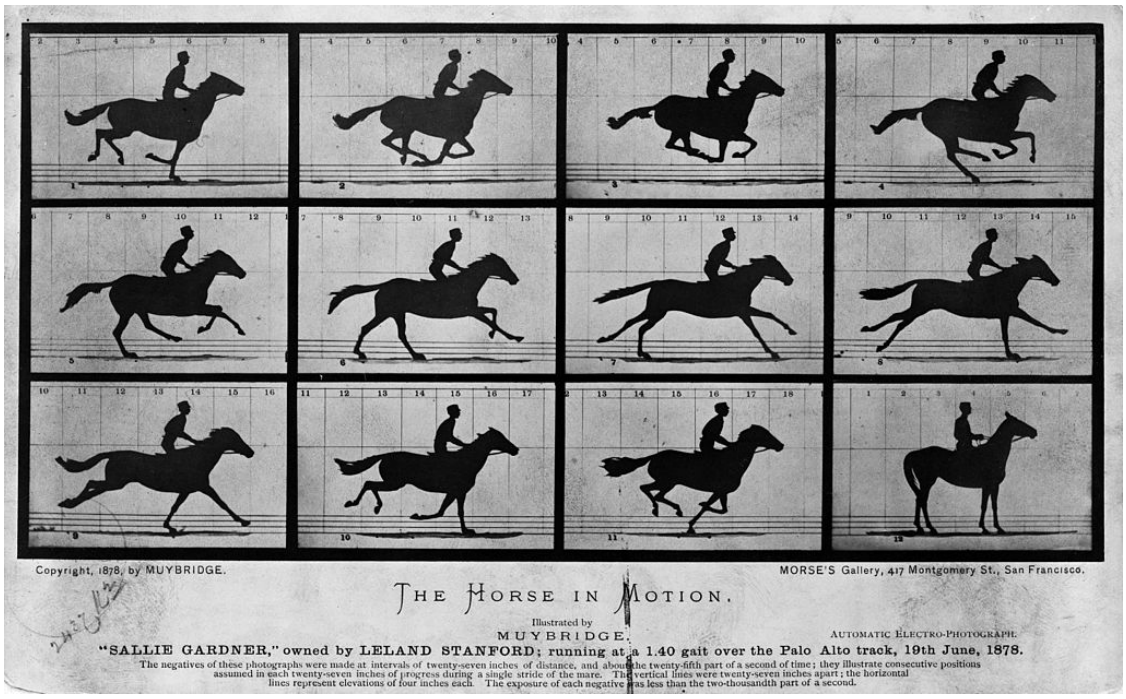


Figure 5-5: Famous animation sequence of a galloping horse by Eadweard Muybridge

5-2-1 Gait Notation

In order to define gaits that contain both aerial and stance phases, the notation of the gait order needs to be defined properly. For a random gait describing a walking method for a n-legged system, the gait order consists of multiple leg-groups lifting off after each other. The time in between these successive lift-offs of the leg-groups can vary.
It is important to emphasize the gait order is defined by the order of the **lift-offs** of the gait. The reason for this choice is the discrepancy between the method of Section 3 and Section 5-1, where the focus is either on starting with all legs on the ground, or all legs in the air. With the insights gained from Section 5-1, having the synchronization of the lift-off based on the touchdown results in problems regarding delays. Therefore, constructing a gait around the lift-off is chosen because of the stable performance regarding the delay.
For the construction of the gait, it is important to not make legs lift-off more than once in one gait instance, as one state vector can only contain the lift-off and touchdown information of one cycle. The notation will be of the form:

$$\{\underset{\tau_{f1}}{\text{leg } t}, ..., \text{leg } w\} \underset{\tau_{s1}}{\prec} \dots \underset{\tau_{s(m-1)}}{\prec} \{\underset{\tau_{fm}}{\text{leg } p}, ..., \text{leg } q\}, \underset{\tau_{sm}}{\prec}, \quad \text{or} \tag{5-17}$$

$$\underset{\tau_{f1}}{l_1} \underset{\tau_{s1}}{\prec} \dots \underset{\tau_{s(m-1)}}{\prec} \underset{\tau_{fm}}{l_m} \underset{\tau_{sm}}{\prec} \tag{5-18}$$

With $m \leq n \in \mathcal{N}$ the amount of leg-groups, l_i leg-group i with $i \in m$, $t, w, p, q \leq n \in \mathcal{N}$. The τ_{si} signifies a 2×1 vector which contains the synchronization between the lift-off of the next leg-group on one side, and the touchdown of and lift-off of the earlier leg-group on the other side. These times can be written as τ_{sli} (the lift-off synchronization) and τ_{sti} (the touchdown

synchronization).

$$\tau_{si} = \begin{pmatrix} \tau_{sli} \\ \tau_{sti} \end{pmatrix} \qquad \text{with:} \tag{5-19}$$

$$\tau_{sli} = \begin{cases} \text{Lift-off to lift-off synchronization times between leg-groups } i \text{ and } i+1 & \text{for } 0 < i < m \\ \text{Lift-off to lift-off Synchronization time between leg-groups } m \text{ and } 1 & \text{for } i = m \end{cases} \tag{5-20}$$

$$\tau_{sti} = \begin{cases} \text{Touchdown to lift-off synchronization times between leg-groups } i \text{ and } i+1 & \text{for } 0 < i < m \\ \text{Touchdown to lift-off Synchronization time between leg-groups } m \text{ and } 1 & \text{for } i = m \end{cases} \tag{5-21}$$

The τ_{fi} are the flight times of the legs in leg-group i.
For most gaits, only one of these synchronizations is necessary for the gait to work. However, more difficult gaits can be made to rely on both the lift-off and the touchdown of the previous leg-group. Especially running gaits can benefit from this addition, as the gait can rapidly be adjusted for body movement or other influences from the outside. The τ_{sti} can be determined to be negative for this implementation. This will be more extensively explained in Section 5-2-4. The gait notation of the gallop gait can be seen in Equation 5-22.

$$\underset{t_{f1}}{\{4\}} \underset{t_{s1}}{\prec} \underset{t_{f1}}{\{3\}} \underset{t_{s2}}{\prec} \underset{t_{f1}}{\{2\}} \underset{t_{s3}}{\prec} \underset{t_{f1}}{\{1\}} \underset{t_{s4}}{\prec} \tag{5-22}$$

When the double synchronization is not necessary, the synchronization vector can be constructed following Equations 5-23 - 5-24. The moment of lift-off and touchdown of leg-group i are noted by lo_i and td_i

$$\tau_{sli} = \begin{cases} \epsilon & \text{if } \text{lo}_{i+1} \geq \text{td}_i \\ a & \text{if } \text{lo}_{i+1} < \text{td}_i \text{ with } 0 < a < \tau_{fi} \end{cases} \tag{5-23}$$

$$\tau_{sti} = \begin{cases} b & \text{if } \text{lo}_{i+1} \geq \text{td}_i \text{ with } b > 0 \\ \epsilon & \text{if } \text{lo}_{i+1} < \text{td}_i \end{cases} \tag{5-24}$$

5-2-2 Constructing A_0 and A_1-matrices

Now, for the construction of both matrices, certain differences are made with regards to the regular walking gaits Chapter(3-1). The notation has added the synchronization time (τ_{si}), which signifies the relation between the lift-off of group i and the lift-off of $i+1$. When the synchronization time is smaller than the flight time, the next lift-off happens before the touchdown of the current leg-group.

Alongside the gait order, three vectors containing the flight, ground and synchronization time need to be determined. These vectors are the flight time vector $\mathbf{t_f} = [\tau_{f1}, ..., \tau_{fm}]$, ground time vector $\mathbf{t_g} = [\tau_{g1}, ..., \tau_{gm}]$, and synchronization time vector $\mathbf{t_s} = [\tau_{s1}, ..., \tau_{sm}] = [\begin{bmatrix} \tau_{sl1} \\ \tau_{st1} \end{bmatrix}, ..., \begin{bmatrix} \tau_{slm} \\ \tau_{stm} \end{bmatrix}]$. These times correspond with the legs contained in leg-group 1 to m.

P_l, P_t, Q_l and Q_t matrices

The method for constructing the P_l, P_t, Q_l and Q_t matrices is similar to the construction of the P and Q matrices determined by Section 3-2. Let P_l, P_t, Q_l and Q_t be square matrices of the size $n \times n$ with n the number of legs. To define P_l and P_t, we use $j = \{1, ..., m-1\}$, with m the amount of leg groups, $\forall p \in l_{j+1}$ and $\forall q \in l_j$, where p and q represent the separate legs in the leg-groups.

$$P^l_{x,y} = \begin{cases} \tau_{slj} & \text{for} \quad P^l_{p,q} \\ \epsilon & \text{for every other} \quad P^l_{x,y} \end{cases} \tag{5-25}$$

$$P^t_{x,y} = \begin{cases} \tau_{stj} & \text{for} \quad P^t_{p,q} \\ \epsilon & \text{for every other} \quad P^t_{x,y} \end{cases} \tag{5-26}$$

To determine Q_l and Q_t, for $\forall v \in l_1$ and $\forall w \in l_m$:

$$Q^l_{x,y} = \begin{cases} \tau_{slm} & \text{for} \quad Q^l_{v,w} \\ \epsilon & \text{for every other} \quad Q^l_{x,y} \end{cases} \tag{5-27}$$

$$Q^t_{x,y} = \begin{cases} \tau_{stm} & \text{for} \quad Q^t_{v,w} \\ \epsilon & \text{for every other} \quad Q^t_{x,y} \end{cases} \tag{5-28}$$

T_f & T_g matrices

Constructing the T_f and T_g matrices is a lot more straight-forward. The desired ground and air times can be filled in an diagonal matrix. For the T_f matrix, the construction depends on the values of τ_{fi}. The legs in the different leg-groups determine the position of the flight times. Again, $i = 1, ..., m$, with m the amount of leg-groups, l_i leg-group i.

$$T^f_{x,y} = \begin{cases} \tau_{fi} & \text{for } y = x, x \in l_i \\ \epsilon & \text{for every other} \quad T^f_{x,y} \end{cases} \tag{5-29}$$

For the T_g matrix, a comparable method is used:

$$T^g_{x,y} = \begin{cases} \tau_{gi} & \text{for } y = x, x \in l_i \\ \epsilon & \text{for every other} \quad T^g_{x,y} \end{cases} \tag{5-30}$$

The construction of the diagonal matrices is simple, the method of determining the variables τ_{fi} and τ_{gi} is more difficult. Depending on the type of gait, either the flight time or the ground time is leading for the performance of the gait. For example, for static gaits with multiple leg-groups, the time legs spend on the ground mainly depends on the amount of leg-groups and their respective flight times.
For the static implementation, described in Section 3, the flight times need to be shorter than the synchronization times with respect to the lift-off (Equations 5-23 and 5-24). This would result in the return of the rotating leg-group before the next group takes off.

Horse Gallop Gait

Now, having all these sub-matrices, the A_0 and A_1 of the gallop gait can be constructed using the values of Table 1 of [27]. This results in:

$$\underset{\tau_{f1}}{\{4\}} \underset{t_{s1}}{\prec} \underset{\tau_{f2}}{\{3\}} \underset{\tau_{s2}}{\prec} \underset{\tau_{f3}}{\{2\}} \underset{t_{s3}}{\prec} \underset{\tau_{f4}}{\{1\}} \underset{t_{s4}}{\prec}, \text{ with:} \tag{5-31}$$

$$\mathbf{t_f} = [0.268 \quad 0.273 \quad 0.278 \quad 0.274] \tag{5-32}$$

$$\mathbf{t_g} = [0.107 \quad 0.102 \quad 0.097 \quad 0.101] \tag{5-33}$$

$$\mathbf{t_s} = \begin{bmatrix} 0.057 & 0.064 & 0.076 & 0.133 \\ \epsilon & \epsilon & \epsilon & \epsilon \end{bmatrix} \tag{5-34}$$

And, when implemented correctly, this results in the A_0 and A_1 matrices in Equations 5-35 and 5-36 and the gait schedule which can be seen in Figure 5-6. The resulting A-matrix is calculated using the method described in Equations 3-7 to 3-12.

$$A_0 = \left(\begin{array}{c|c} \mathcal{E}_{n\times n} & T_f \\ \hline P_t & P_l \end{array}\right) = \left(\begin{array}{cccc|cccc} \epsilon & \epsilon & \epsilon & \epsilon & \tau_{f4} & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \tau_{f3} & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \tau_{f2} & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \tau_{f1} \\ \hline \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \tau_{s3} & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \tau_{s2} & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \tau_{s1} \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{array}\right) \tag{5-35}$$

$$A_1 = \left(\begin{array}{c|c} E_{n\times n} & \mathcal{E}_{n\times n} \\ \hline T_g \oplus Q_t & E_{n\times n} \oplus Q_l \end{array}\right) = \left(\begin{array}{cccc|cccc} 0 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & 0 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & 0 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & 0 & \epsilon & \epsilon & \epsilon & \epsilon \\ \hline \tau_{g4} & \epsilon & \epsilon & \epsilon & 0 & \epsilon & \epsilon & \epsilon \\ \epsilon & \tau_{g3} & \epsilon & \epsilon & \epsilon & 0 & \epsilon & \epsilon \\ \epsilon & \epsilon & \tau_{g2} & \epsilon & \epsilon & \epsilon & 0 & \epsilon \\ \epsilon & \epsilon & \epsilon & \tau_{g1} & \tau_{s4} & \epsilon & \epsilon & 0 \end{array}\right) \tag{5-36}$$

Implementing this result in the equation $v(k+1) = A \otimes v(k)$ results in the gait schedule portrayed in 5-6, with $v(0) = [0,0,0,0,0,0,0,0]^T$.

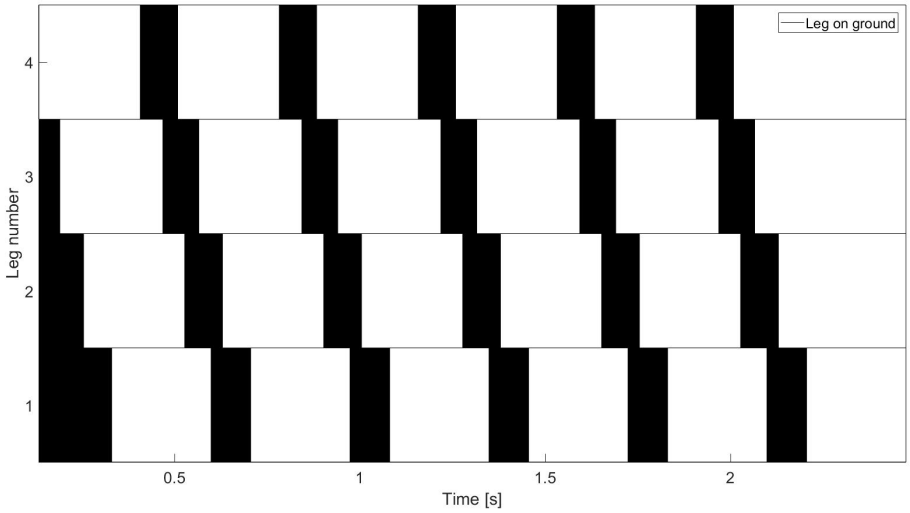


Figure 5-6: Gait schedule of the gallop gait with time constants supplied by the research of N. Deuel and L. Lawrence [27] . This is the schematic representation of the galloping horse of Figure 5-5.

5-2-3 Gait Transitions

In order to improve the transition between two different gaits, the method explained in Section 3-3-2, uses a change in the individual flight times of the legs to accommodate improved transitions [21]. This can be extended to the new system. Due to the variable flight times of the leg-groups, slight changes need to be made in the algorithm derived by the research of B. Kersbergen et al. [21].
The main difference is the determination of the T_f-matrix, which is slightly altered with respect to Equation 5-29, due to the difference between the movement of the legs within the same leg-group (which is necessary to optimize the transition).
Adapted from the research of [21], the system then becomes:

$$x(k+1) = \left(\begin{array}{c|c} \mathcal{E}_{6\times6} & R \\ \hline P_t & P_l \end{array}\right) \otimes x(k+1) \oplus \left(\begin{array}{c|c} E_{6\times6} & \mathcal{E}_{6\times6} \\ \hline \tau_g \otimes E_{6\times6} \oplus Q_t & E_{6\times6} \oplus Q_l \end{array}\right) \otimes x(k) \tag{5-37}$$

Where the sub-matrix R is the diagonal matrix made up according:

$$R = \begin{bmatrix} \tau_{trans1} & & \mathcal{E} \\ & \ddots & \\ \mathcal{E} & & \tau_{transn} \end{bmatrix} \tag{5-38}$$

Where the transition flight times which fills the R-matrix is calculated as the τ_{trans} in Equations 3-35 to 3-41. The main purpose of this method is to decrease the extra ground time in transitions between two static gaits.
For transitioning between running gaits to walking gaits, the method is effective for decreasing the transition time, and with either the starting point or ending point in the transition

having all legs on the ground, no major problems can be expected in this implementation.
Running gaits show more difficulties, as the dynamics of the gaits need to be observed in order to achieve a reliable transition.
However, new possibilities in gaits also result in more possibilities regarding the transitions between them. Constructing in-between gaits that share similarities with both gaits may yield better results in the transitions.
For example, consider the transition from the pentapod to the tripod gait. When the legs rotate in a comparable leg order, the in-between gait can contain behavior which has elements from both gaits.
The tripod($\mathcal{G}_t$) and pentapod gait ($\mathcal{G}_p$) are defined by the following gait-orders. For both gaits, $\tau_{fi} = 0.75$, $\tau_{gi} = 1$, $\tau_{si} = [\epsilon, 0, 25]^T, \forall i$:

$$\mathcal{G}_p = \underset{\tau_{f1}}{\{1\}} \underset{\tau_{s1}}{\prec} \underset{\tau_{f2}}{\{4\}} \underset{\tau_{s2}}{\prec} \underset{\tau_{f3}}{\{5\}} \underset{\tau_{s3}}{\prec} \underset{\tau_{f4}}{\{2\}} \underset{\tau_{s4}}{\prec} \underset{\tau_{f5}}{\{3\}} \underset{\tau_{s5}}{\prec} \underset{\tau_{f6}}{\{6\}} \underset{\tau_{s6}}{\prec} \tag{5-39}$$

$$\mathcal{G}_t = \underset{\tau_{f1}}{\{1,4,5\}} \underset{\tau_{s1}}{\prec} \underset{\tau_{f2}}{\{2,3,6\}} \underset{\tau_{s2}}{\prec} \tag{5-40}$$

Because of the similarities in leg order of both gaits, a comparison can be made to show the differences in the synchronization time between the lift-offs of the separate legs:

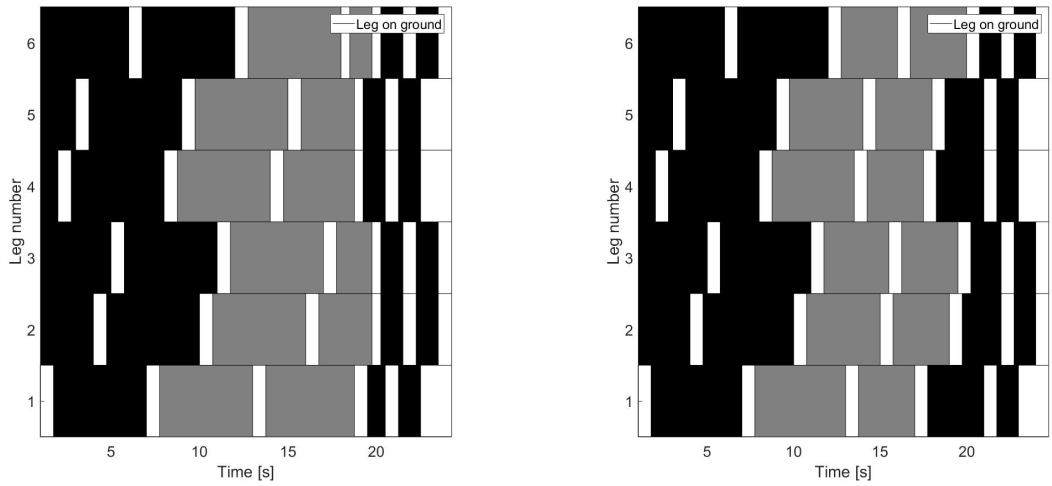
$$\mathcal{G}_p = \{1\} \underset{1}{\prec} \{4\} \underset{1}{\prec} \{5\} \underset{1}{\prec} \{2\} \underset{1}{\prec} \{3\} \underset{1}{\prec} \{6\} \underset{1}{\prec} \tag{5-41}$$

$$\mathcal{G}_t = \{1\} \underset{0}{\prec} \{4\} \underset{0}{\prec} \{5\} \underset{1}{\prec} \{2\} \underset{0}{\prec} \{3\} \underset{0}{\prec} \{6\} \underset{1}{\prec} \tag{5-42}$$

Simply taking the average of the ground, flight and synchronization time results in the gait:

$$\mathcal{G}_{trans} = \{1\} \underset{0.5}{\prec} \{4\} \underset{0.5}{\prec} \{5\} \underset{1}{\prec} \{2\} \underset{0.5}{\prec} \{3\} \underset{0.5}{\prec} \{6\} \underset{1}{\prec} \tag{5-43}$$

The transition gait has the same ground and flight times as the other two gaits. This results in the gait scheme of Figure 5-7.



(a) Gait schedule of the six-legged system with the transition following Equation 5-37.

(b) Gait schedule of the six-legged system with the transition gait following Equation 5-43.

Figure 5-7: Transitioning from the pentapod gait defined in Equation 5-41 to the tripod gait defined in Equation 5-42 using two different methods.

5-2-4 Time Delays

The handling of time delays can be done as as explained in Section 4-2-3. To illustrate the impact of the double synchronization on the delay in running gaits, several examples of the same gait are shown. The gaits are characterized by:

$$\mathcal{G} = \{1,4,5\}_{\tau_{f1}} \prec_{t_{s1}} \{2,3,6\}_{\tau_{f2}} \prec_{\tau_{s2}}, \text{ with:} \tag{5-44}$$

$$\mathbf{t_f} = [1 \quad 1] \tag{5-45}$$

$$\mathbf{t_g} = [0.5 \quad 0.5] \tag{5-46}$$

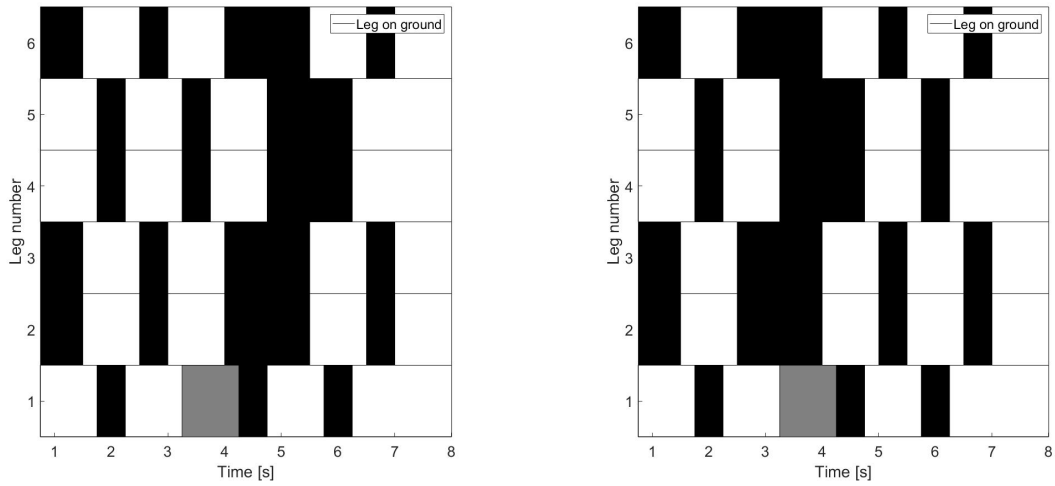
With changing τ_s vector for the three implementations:

$$\mathbf{t_{s\mathcal{G}1}} = \begin{bmatrix} 0.75 & 0.75 \\ \epsilon & \epsilon \end{bmatrix} \tag{5-47}$$

$$\mathbf{t_{s\mathcal{G}2}} = \begin{bmatrix} \epsilon & \epsilon \\ -0.25 & -0.25 \end{bmatrix} \tag{5-48}$$

$$\mathbf{t_{s\mathcal{G}3}} = \begin{bmatrix} 0.75 & 0.75 \\ -0.25 & -0.25 \end{bmatrix} \tag{5-49}$$

The difference in response of the systems to the one second delay in the (projected!) touchdown of leg 1 can be seen in Figure 5-8.



(a) Gait schedule of the system with delay updating, $\mathcal{G}_1$

(b) Gait schedule of the system with delay updating ($\mathcal{G}_2$ and $\mathcal{G}_3$)

Figure 5-8: System response of the Max-Plus gait generation with touchdown delay (in grey) of one second on Leg 1 during the running tripod gait.

As can be seen in the difference of the system response between gait 1 on one side and gait 2 & 3 on the other side, the response to the projected delay in the touchdown has a different impact. The systems with the negative touchdown synchronization recognizes the delay and deal with the consequences, while the system without the negative touchdown synchronization handles the delay for the next step.
The reason for adding the double synchronization comes from the possibility of creating a time frame in which lift-off happens. This possibility will be illustrated using a two-legged system, with the following parameters:

$$\mathcal{G} = \underset{\tau_{f1}}{\{1\}} \underset{t_{s1}}{\prec} \underset{\tau_{f2}}{\{2\}} \underset{\tau_{s2}}{\prec}, \text{ with:} \tag{5-50}$$

$$\mathbf{t_f} = [1 \quad 1] \tag{5-51}$$

$$\mathbf{t_g} = [0.5 \quad 0.5] \tag{5-52}$$

$$\mathbf{t_s} = \begin{bmatrix} 0.75 & 0.75 \\ -0.35 & -0.35 \end{bmatrix} \tag{5-53}$$

The minimal lift-off time of leg two is limited by two synchronized moments:

$$\mathrm{lo}_2 = \mathrm{lo}_1 \otimes 0.75 \oplus \mathrm{td}_1 \otimes -0.35 \tag{5-54}$$

$$\mathrm{lo}_2 = \mathrm{lo}_1 \otimes 0.75 \oplus \mathrm{lo}_1 \otimes 1 \otimes -0.35 \tag{5-55}$$

$$\mathrm{lo}_2 = \mathrm{lo}_1 \otimes 0.75 \oplus \mathrm{lo}_1 \otimes 0.65 = \mathrm{lo}_1 \otimes 0.75 \tag{5-56}$$

Without delays, the synchronization with respect to the lift-off time is leading in the timing of leg 2. The difference between these two times, is a time window which allows for consequence free delay in the touchdown. The gait schedule can be seen in Figure 5-9.

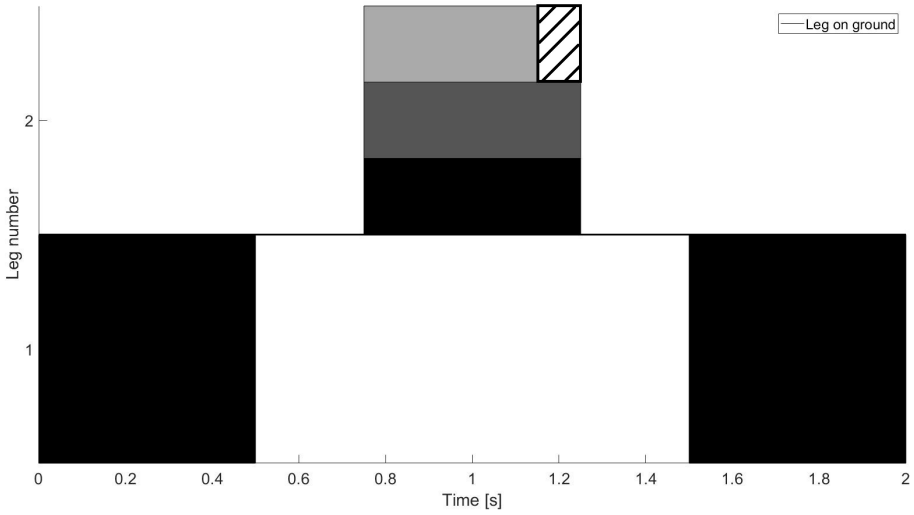
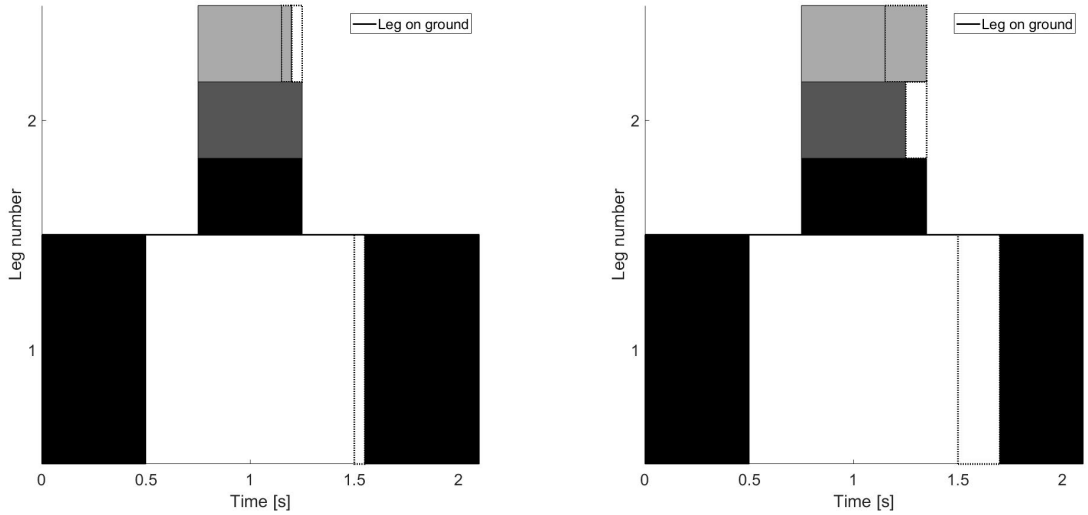


Figure 5-9: Gait schedule of the two legged system with double synchronization. The light-gray part represents the touchdown-based synchronization, the dark-grey part represents the lift-off based synchronization. The black part represents the complete synchronization (which the leg follows). The size of the delay window can be seen as the striped part.

Now, when the system has a delay in the lift-off of leg 1, the lift-off of leg 2 shifts further to the future. If the leg has already experienced its lift-off, and the future touchdown of leg 1 has an anticipated delay, the lift-off of leg 2 is extended. This is only when the delay is larger than the difference in synchronization time (the **time window**) (for the example of Equation 5-2-4, this means the delay has to be larger than 0.1 second). The resulting gait schedules after a delay of 0.05 and 0.2 seconds in the touchdown of leg 1 can be seen in Figure 5-10.



(a) Delay of 0.05 seconds, or half the time window. As can be seen, the ground time of leg 2 is not adjusted.

(b) Delay of 0.2 seconds, or twice the time window. The total ground time of leg 2 is increased due to the touchdown synchronization.

Figure 5-10: Gait schedule of the two legged system with double synchronization subject to different lengths of delay. The light-gray part represents the touchdown-based synchronization, the dark-grey part represents the lift-off based synchronization. The black part represents the complete synchronization which the leg will follow. The delays are represented by the dotted outline.

5-3 Hopping gait

Jumping with all legs at the same time, or hopping, is mainly done by two-legged animals, like kangaroos or humans. However, 4-legged animals can also jump, and some are even known for it. For instance, the African springboks. More common (at least, for Dutch people) animals like lambs and goats can be seen hopping when excited. In the domain of insects, jumping happens more often than not. However, for testing and implementing the performance of the dynamic gaits in Max-Plus, the hopping is one of the most interesting gaits to research in terms of stability and transitioning. This is mainly due to the (in the ideal situation) lack of roll, pitch and yaw while running. However, small irregularities can disturb this ideal situation.
Sadly, the hopping cannot be implemented on the ZeBro in its current state. The legs are not able to contain the springy behavior required for the hopping, and the motors of the legs are not strong or fast enough to lift the ZeBro from the ground. Therefore, this section will be strictly about the theoretical implementation of hopping in walking robots.

The leg order of the hopping gait consists of one single group:

$$\mathcal{G}_h = \{1,2,3,4,5,6\} \underset{\tau_{f1}}{\prec} \underset{\tau_{s1}}{} \tag{5-57}$$

The determination of the ground and flight times depend heavily on the dynamics of the system. For the synchronization times, it is enough to assume that the touchdown/lift-off synchronization is equal to the ground time, and the lift-off/lift-off synchronization equal to the total of the ground and flight time, $\tau_{st1} = \tau_g$ and $\tau_{sl1} = \tau_f \otimes \tau_g$.

Following the procedure of Chapter 5-2-2, the P and Q matrices are defined by:

$$P_l = P_t = \mathcal{E}_{6\times 6} \tag{5-58}$$

$$Q_t = \begin{pmatrix} \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} \\ \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} \\ \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} \\ \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} \\ \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} \\ \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} & \tau_{st1} \end{pmatrix} \tag{5-59}$$

$$Q_l = \begin{pmatrix} \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} \\ \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} \\ \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} \\ \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} \\ \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} \\ \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} & \tau_{sl1} \end{pmatrix} \tag{5-60}$$

(5-61)

Using these matrices, the A-matrix is constructed. The resulting lift-off and touchdown schedule of the robot transitioning from tripod gait to the hopping gait can be seen in Figure 5-11. The gait parameters are of big importance when applying the hopping gaits. In order to determine the flight and ground times, the dynamics of the robot need to be incorporated in the Max-Plus system.

5-3-1 SLIP model

The use of the Spring-Loaded Inverted Pendulum model (SLIP-model) in the modeling of legged locomotion allows for better understanding of the dynamics of the legs hitting the ground. The SLIP-model represents the legs as loss-less springs, which support the body. This model can be extended to include the inevitable energy losses in the legs (due to deformation), by extending the system to contain a damping factor. A schematic overview of the SLIP model with one leg connected to the center of mass of the robot can be seen in Figure 5-12.

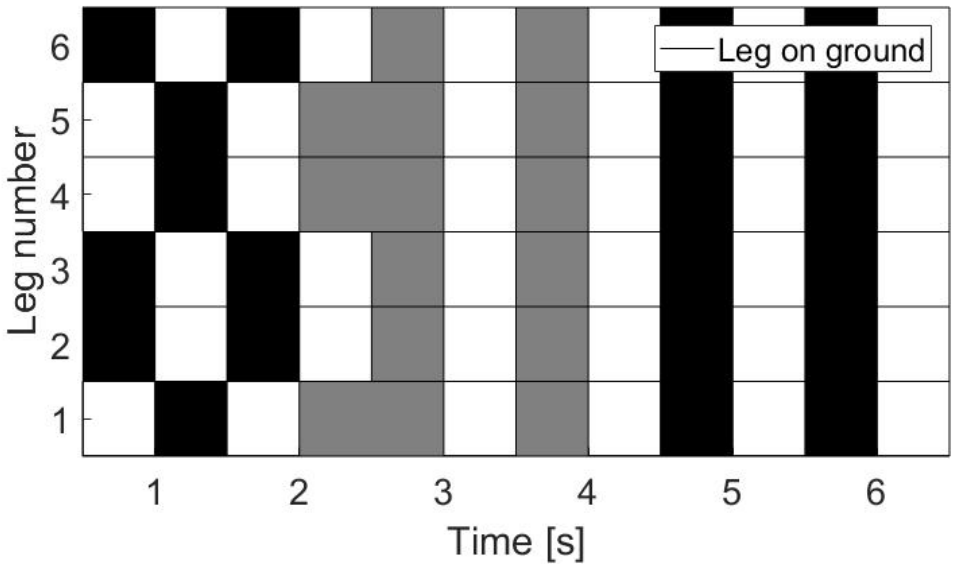


Figure 5-11: Gait schedule of a six-legged system transitioning from the tripod gait to the hopping gait, with the same ground and flight times of the legs.

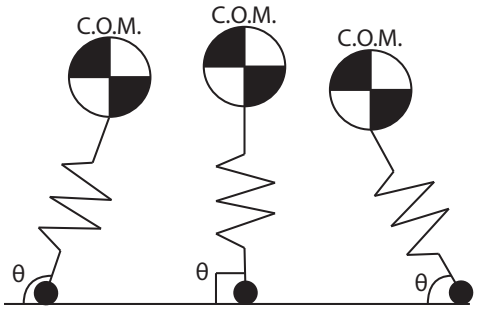


Figure 5-12: SLIP-model of a robot with one leg, acting under different angles

The system can be expended to have more than one leg, and to determine the physics of the system in different situations. For example, a two-legged robot can have 3 different states, the flight state (no legs on the ground), the single-stance state and the double stance state. Using the dynamics of the legs and the rest of the system, simulations can be made to help the implementation of a controller for hopping. For the dynamics simulations in Matlab, the aim is to portray the dynamics of the ZeBRo. For dynamic gaits, the SLIP model is extended to contain the forward and upward movement of the system, while also including the time no leg makes contact with the ground.

5-3-2 One-Dimensional Hopper

The easiest form to describe the ZeBRo is by describing it as a hopping mass, consisting of a mass and a (lossless) spring, which moves in one direction, and consists of two phases: an aerial phase, when the spring is not in contact with the ground, and the ground phase, when

the spring is being under compression and is exerting a force on the body.
However, this description does not capture the body rotations which portray the movements resulting from asymmetric actuations or ground fluctuations.

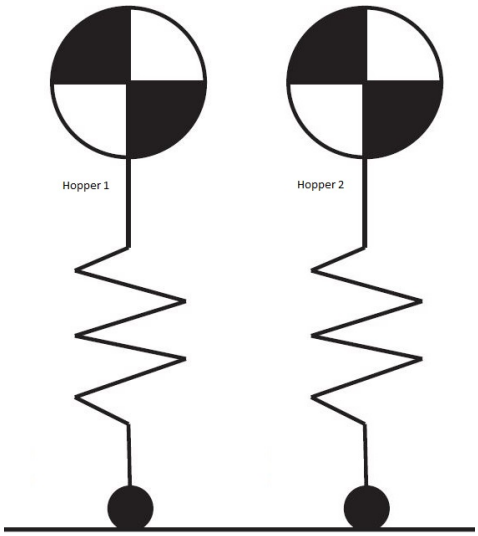


Figure 5-13: Schematic overview of the independent hopper model

Therefore, for the first (simple) model, we use an independent hopper. The dynamics of the spring are modeled as follows:

$$F_y = f(y) \cdot k_0 \cdot (y_- l_0) \tag{5-62}$$

$$\text{where } f(y) = \begin{cases} 1 & \text{if } 0 < y \leq l_0 \\ 0 & \text{if } y > l_0 \end{cases} \tag{5-63}$$

With the following variables and constants:

Sign	Variable/Constant
m	Mass of hopper
g	Gravitational constant
y	y-coordinate mass of hopper
x	x-coordinate mass of hopper
F	Vertical force delivered by the spring of the hopper
l_0	Uncompressed length of the spring of the hopper
k_0	Spring constant of the spring of the hopper
c_0	Spring damping constant of the hopper
ω	Natural frequency of the spring-mass system ($\omega^2 = k/m$)

Table 5-1: Constants and variables of the hopper-simulation

As the Max-Plus hopping gait tries to keep the system in a regular pattern, it is tried to keep the system in this repeating state, 'repairing' small fluctuations in the body movement. The

system is modeled to contain a spring that exerts a regularly timed force. Before this force is implemented though, the natural frequency of the hopper needs to be determined. Following [28] for Equations 5-64 to 5-82, using a linear spring, the equation of motion when in contact with the ground is defined as:

$$\ddot{y} + ky = mg \tag{5-64}$$

Which has the following solution for the differential equation (the i subscript is discarded due to the identical dynamics of both hoppers):

$$y(t) = a\sin(\omega t) + b\cos(\omega t) + g/\omega^2 \tag{5-65}$$

Setting $y(t=0) = 0$ for the moment of touchdown (where $F = 0$) results in:

$$y(0) = b + g/\omega^2 = 0 \tag{5-66}$$
$$b = -g/\omega^2 \tag{5-67}$$

Now, using $\dot{y}(t=0) = \dot{y}_{td}$ for the velocity while landing, a can be determined:

$$\dot{y}(t) = a\omega\cos(\omega t) - b\omega\sin(\omega t) \tag{5-68}$$
$$\dot{y}(0) = a\omega = \dot{y}_{td} \tag{5-69}$$
$$a = \dot{y}_{td}/\omega \tag{5-70}$$

Which results in the complete differential equation:

$$y(t) = \dot{y}_{td}/\omega\sin(\omega t) - g/\omega^2\cos(\omega t) + g/\omega^2 \tag{5-71}$$

Because the velocity of the mass is equal to zero halfway the time on the ground (when all energy is contained in the compressed spring, $t = t_h$), this moment can be calculated using:

$$\dot{y}(t_h) = \dot{y}_{td}\omega\cos(\omega t_h) + g/\omega\sin(\omega t_h) = 0 \tag{5-72}$$
$$\tan(\omega t_h) = -\dot{y}_{td}\omega/g \tag{5-73}$$
$$t_h = \tan^{-1}(y_{lo}\omega g)/\omega \tag{5-74}$$

The hopping height is limited to 5 cm above the length of the springs (l_{0i}), in order to not damage the legs and the rest of the system. Using this information, the (vertical) lift-off velocity can be determined using the energy balance:

$$\frac{m\dot{y}_{lo}^2}{2} = 0.05 \times mg \tag{5-75}$$
$$\dot{y}_{lo} = -0.99 \text{ [m/s]} \tag{5-76}$$

Due to energy conservation, the touchdown velocity is equal to the negative lift-off velocity ($\dot{y}_{td} = -\dot{y}_{lo}$).
Substituting the constants $k = 5000$ [F/m], $m = 1$ [kg] per hopper, $\dot{y}_{td} = 0.99$[m/s] and $g = 9.81$ [m/s^2] in Equation 5-74 results in $t_h = 0.025$ [s]. These constants are roughly the constants of the current ZeBRo, where the spring, and further in this chapter, the damping constant, are both roughly estimated.
After lift-off, the robot spends some time in the air. This aerial time can be calculated using the lift-off velocity (ignoring the $t_{ae} = 0$ result):

$$y(t) = \dot{y}_{lo} t_{ae} - \ddot{y}\frac{t_{ae}^2}{2} = 0 \tag{5-77}$$

$$\dot{y}_{lo} t = -g\frac{t_{ae}^2}{2} \tag{5-78}$$

$$t_{ae} = 0.20 \tag{5-79}$$

The total time of one hopping cycle t_{hc} will then take $t_{hc} = t_{ae} + 2t_h = 0.25$ [s]. Using these results, the τ_f, τ_g and τ_δ can be determined.

Adding damping to the legs is necessary to achieve a closer likeness to the actual system. Like the spring constant k, the damping constant is hard to determine. For now, it is set on 15 [Fs/m], as the bouncing behavior shows similarities with the real system. For the implementation in Chapter 7, these constants will be examined further. The equations of motion are slightly different now with respect to Equation 5-64, adding the damping term.

$$m\ddot{y} + c\dot{y} + ky = mg \tag{5-80}$$

This also changes the natural frequency slightly, due to the adding of the damping. In order to determine this change, the damping ratio ζ needs to be determined:

$$\zeta = \frac{c}{2\sqrt{mk}} \tag{5-81}$$

$$\zeta = 0.12 \tag{5-82}$$

Using the ζ value to compute the damped frequency, results in a comparable value and is not significant different than the result of Equation 5-79. A comparison of the system response with and without damping can be seen in Figure 5-14. Note that the initial velocity of the hoppers is different, in order to illustrate the continuous effect of a slight difference in the frequency of the system.

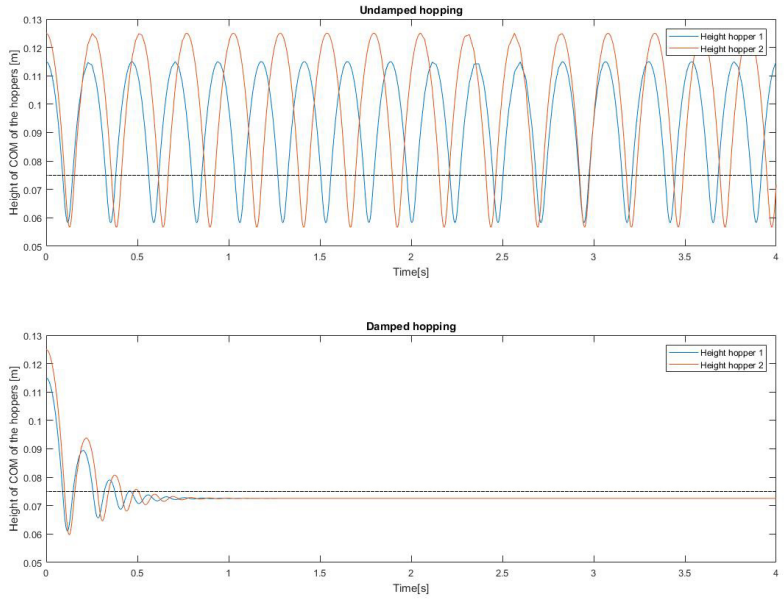


Figure 5-14: Vertical height of two hoppers with different initial conditions in two hopping simulations: without damping and actuation and with damping but no actuating (with $y_1 = 0.115$, $y_2 = 0.125$, $\dot{y}_1 = 0$ and $\dot{y}_2 = -0.5$)

In order to keep the system hopping, energy needs to be added to the hoppers. During the descent of the hopper, the spring is used to decelerate the mass. After the hopper has reached its lowest point, a force can be exerted by the legs to propel the hopper upwards again. The energy loss (E_{loss}) after one hop cycle can be calculated by observing the difference in height the hopper reaches at the apex.

$$E_{loss} = mg\Delta h \tag{5-83}$$

$$E_{loss} = 1 * 9.81 * 0.031 = 0.3[\text{J}] \tag{5-84}$$

This energy needs to be added again in the upwards motion. Using the compressed distance (Δs_{comp}), the force that needs to be added can be calculated.

$$E_{loss} = F_{leg}\Delta s_{comp} \tag{5-85}$$

$$F_{const} = \frac{E_{loss}}{\Delta s_{comp}} \tag{5-86}$$

$$F_{const} = \frac{0.3}{0.015} = 20[\text{N}] \tag{5-87}$$

When this is implemented in the simulation, the results start to look like the undamped hopping.

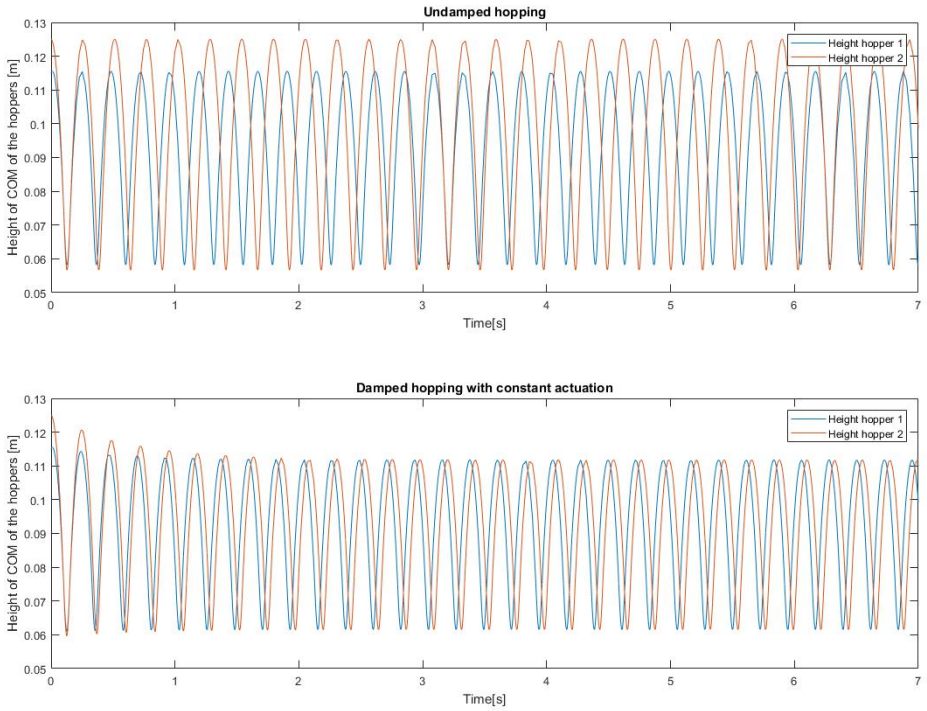


Figure 5-15: Vertical height of two hoppers with different initial conditions in two hopping simulations: without damping and actuation and with both damping and constant actuating (with $y_1 = 0.115$, $y_2 = 0.125$, $\dot{y}_1 = 0$ and $\dot{y}_2 = -0.5$)

In Figure 5-15, it can be seen that the height difference, and alongside, the frequency difference, is solved by implementing the damping and actuation. The resulting lower height than anticipated is probably due to the extra damping caused by the additional speed of the hopper. This can easily solved by slightly increasing the actuation force.

5-3-3 Gait Dependent Implementation

To implement the running behavior in the ZeBRo, the time delay between the different legs and the movement of the body need to understood. To find out how the controllers need to be implemented, the expected body movement resulting from the gait with aerial phases can be analyzed. Because the body movement is heavily influenced by the leg order, three different gaits with aerial or non-static phases will be reviewed. In order to keep the ZeBRo leveled during the implementation of these gaits, a controller is implemented. The working principles of this controller can be seen in Section 5-4.

Hopping

The gait with the biggest likeness to the independent hoppers is the simple hopping gait, characterized by the following leg order:

$$\mathcal{G}_{hopping_4} = \{1, 2, 5, 6\} \prec \{1, 2, 5, 6\} \tag{5-88}$$
$$\mathcal{G}_{hopping_6} = \{1, 2, 3, 4, 5, 6\} \prec \{1, 2, 3, 4, 5, 6\} \tag{5-89}$$

The resulting behavior will be like the hopper, with four or six parallel springs acting at the same time. Without disturbances in the ground height and with stable body movement, the movement of the body will roughly be comparable to the movement of Figure 5-15.

Bounding Gait

The bounding gait is a gait mostly used by predators, as it allows for rapid acceleration. The corresponding leg sequence is given by:

$$\mathcal{G}_{bounding} = \{1, 2\} \prec \{5, 6\} \tag{5-90}$$

Because of the alternation of the actuation between the front and hind legs, the system will act like a double hopper system. Therefore, the reference trajectory of the robot should be two hoppers in anti-phase. This would then result in similar parameters as are applicable on the hopping gait, but another Max-Plus implementation.
The Max-Plus parameters can then again be calculated using the method of Section 5-3-2. If the aerial time and the ground time of the example (Equations 5-74 and 5-79) is followed, the parameters are as follows: $\tau_f = t_{ae}$, $\tau_g = 2t_h$.

5-4 Touchdown Controller

Because the computing power of the ZeBRo is too small to implement a controller which can accurately use a model of the system to calculate the full state of the robot, it is chosen to focus on a relative simple PID-like controller for body stability, focusing on the input of an IMU (inertia measurement unit) and the leg encoder. The research of Chun-Kai Huang et al.[29] shows a method for dynamic hopping with a Rhex-like robot. Important parts of the control loop are processed by an external computer, even though the robot itself should allow for more calculating power inside its body due to the difference in size.
The research of M. Shahbazi and G.A.D. Lopes [30] shows a controller making use of Max-Plus for the synchronization of hoppers, which shows very promising results. The controller is made to synchronize multiple mass-hopper systems, which could be seen as an abstraction of a walking system. Part of the modeling tactics of this report are used for the behavior of the system, but the overall controller requires more accurate sensing and computing power to incorporate in the ZeBRo than it currently possesses.
The controller should be able to make changes in the timing or leg angles, and should make the ZeBRo more stable in its performance during gaits with unstable phases. The implementation will mainly rely on slightly changing the leg angles to make sure the touchdown and lift-off of

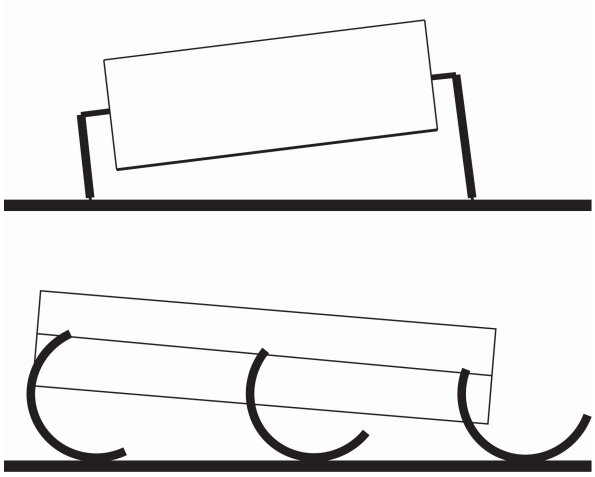


Figure 5-16: Change of the touchdown angle of the leg due to body roll (up) and body pitch (down)

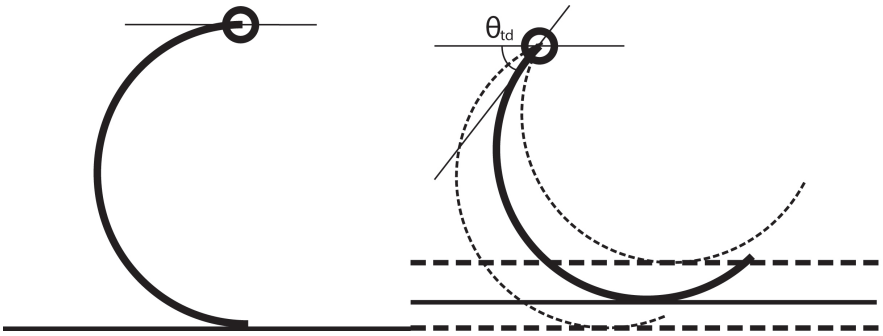


Figure 5-17: Left: Leg in neutral position, right: Illustration of the change of the touchdown angle and the resulting change in height. It can be noticed that the point of ground contact of the leg changes with a changing touchdown angle.

the leg-groups happen with a smaller time difference, to keep the ZeBRo as leveled as possible.

As can be seen in Figure 5-16, the height of the touchdown angle of the legs is slightly changed for the body roll and pitch. It has to be noted that there should be a limit on the leg angle change, due to the actuation distance of the legs. Therefore, saturation is introduced to limit the influence of the controller up to a certain degree. The controller consists of two parts, one with respect to the pitch, and one with respect to the roll. The pitch controller adjusts the front and rear legs, while the roll controller adjusts the right and left legs. To minimize the difference in distance traveled by the individual legs (to prevent yaw), the lift-off angle is changed with the same delta angle as the touchdown angle. For the determination of k_r and k_p, the ratio between the body width or length on one side, and the length of the legs on the other side.

$$\theta_{tdC} = \theta_{td} + k_p \begin{bmatrix} -\theta_p & -\theta_p & 0 & 0 & \theta_p & \theta_p \end{bmatrix} + k_r \begin{bmatrix} \theta_r & -\theta_r & \theta_r & -\theta_r & \theta_r & -\theta_r \end{bmatrix} \tag{5-91}$$

$$\theta_{loC} = \theta_{lo} - k_p \begin{bmatrix} -\theta_p & -\theta_p & 0 & 0 & \theta_p & \theta_p \end{bmatrix} - k_r \begin{bmatrix} \theta_r & -\theta_r & \theta_r & -\theta_r & \theta_r & -\theta_r \end{bmatrix} \tag{5-92}$$

Part III

Implementation

Chapter 6

Implementation of the Locomotion Module in the ZeBRo

In order to make the ZeBRo walk, the locomotion module needs to be physically implemented in the ZeBRo. This locomotion module needs to be capable of controlling the separate legs of the robot, in order to move as efficient as possible. For the implementation of the locomotion module, the information transmission between the surrounding modules needs to be clearly defined, and will be explained in section 6-1. After this, the structure of the locomotion program will be explained.

6-1 Control Structure

For the current iteration of the ZeBRo, multiple levels of control are defined. The top-level controller is responsible for making decisions about where the ZeBRo has to go, what the ZeBRo should do, and how it needs to complete these tasks. The top-level controller can be autonomous, or can contain manual instructions of an operator.
Below this top-level controller are the mid-level controllers. The locomotion module is one of these. The main function of the mid-level controllers is to give the top-level controller feedback about the current state of affairs, while also performing the tasks defined by the top-level controller. These tasks will then be processed in order to make workable instructions for the low-level controllers.
Finally, the low-level controllers are responsible for gathering information and executing commands. The leg modules are examples of low-level controllers, and are responsible for the actuation of the legs.

6-1-1 Overview

To give an overview of the systems architecture, Figure 6-1 can be observed. This simplified overview is made to illustrate the information streams in the robot system. The locomotion can be viewed as a mid-level controller. An other examples of mid-level controllers are the communication controller for interacting with other ZeBRos. The information contained in these streams are explained in the next section.

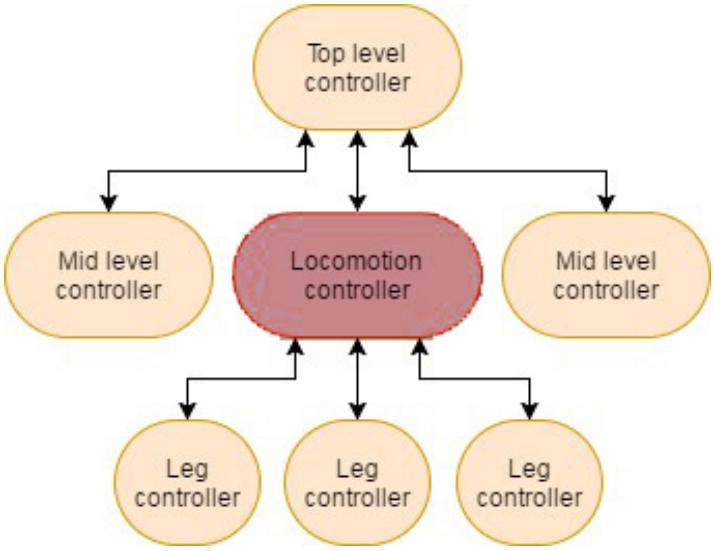


Figure 6-1: A simplified overview of the control system in the ZeBRo

6-1-2 Information transferral

For the transfer of information, the ZeBRo-bus system is used[verwijzing naar]. This system uses the I2C-protocol to transfer the data between the modules.

Top-level controller to locomotion controller

The information of the locomotion module receives is defined by the vector Ψ, which contains information about the direction, speed and the mode of operation:

$$\Psi = \begin{pmatrix} \text{Turning radius} \\ \text{Speed} \\ \text{Surface} \end{pmatrix} \tag{6-1}$$

Turning radius Contains the turning radius in the direction the ZeBRo wants to go. Turning will be done on the spot if a sharp turn is required. More gentle ways of turning will be done by increasing or decreasing the size of steps taken on both size, creating a difference in traveled distance on the two sides. Another method of turning is by implementing the research of W. Suriana [23].

Speed The speed is a number from $0-9$, and will influence the gait (for example, a tripod gait for high speeds, a pentapod gait for low speeds), but also the ground, flight and double stance times of the ZeBRo.

Surface Contains information regarding the roughness of the terrain.

The turning radius is now divided up in several discrete setting for the turning radius. This ranges from a turning radius of 40 cm to a turning radius of about 3 meter, and is done by increasing and decreasing the step sizes on both sides. For a correct implementation of the research of Suriana[23], more research needs to be done to the possibilities en limitations.
The surface information can mainly be divided into three distinct categories: normal terrain, difficult terrain and stairs. The recognizing of the terrain will be added later to the robot itself, for now the recognition is done by gathering data of the inertia measurement unit.

Locomotion controller to top-level controller

The information send to the top-level controller is mostly based on the current state of the locomotion. If both the locomotion controller and the separate legs are operating at their required performance, the communication can be limited to sending a signal of the current state of operations and the ability to full-fill these properly.
When the locomotion controller is not able to perform its required tasks however, the information needs to contain the nature of malfunctioning. The system should also be able to convey the possibilities left for performance. This could for instance be that one of the legs is not operating as required, but can be solved by slowing down and altering the gait used.

$$\Pi = \begin{pmatrix} \text{State of operation} \\ \text{Error Situation} \end{pmatrix} \tag{6-2}$$

The state of operation is limited to being in walking mode and resting. This resting is done to prevent overheating of the leg-motors, but also allows other moving parts to cool down.

Locomotion controller to leg-module

The main information send from the locomotion controller to the legs will be the required location of the leg at a specific time. Along with the direction, this message will contain all necessary information to make the legs function. The information send by the locomotion controller to the legs is defined by Γ:

$$\Gamma = \begin{pmatrix} \text{Required leg location} \\ \text{Time of arrival} \\ \text{Rotational direction} \\ \text{Special instructions} \end{pmatrix} \tag{6-3}$$

Required leg location The leg location which the specific leg needs to move to in order to make steps. For a walking gait, the locations of lift-off and touchdown are determined using performance measurements.

Time of arrival In order to make the ZeBRo walk, the time the legs need to be at a specific location needs to be send to the legs. The time of the locomotion module will constantly be used to update and synchronize the time on the leg modules.

Rotational direction The direction of the legs determines which way the individual legs turn. For turning the robot or walking backwards, the legs need to rotate backwards.

Special instructions For special operations like calibrating, stopping, laying down or emergency stopping, some special instructions which overrides the current operations are necessary.

Using the location, time and direction, the leg modules can make a trajectory to follow. The reason the trajectory making is done in the leg modules is to limit the amount of information send between the modules, effectively making the modules more independent. All the information related to walking is gathered from the Max-Plus calculations running on the background. More information about the processing of the lift-off and touchdown times can be seen in Chapter 6-2.

Leg-module to locomotion controller

The feedback of the independent legs to the locomotion controller allows the controller to check the performance and location of the legs. Moderation of the movement of the legs is necessary for knowing when to update the required position. The information send from the leg-modules to the locomotion controller is defined by χ:

$$\chi = \begin{pmatrix} \text{Leg location} \\ \text{Temperature} \\ \text{Leg state} \end{pmatrix} \tag{6-4}$$

Leg location The current location of the legs are important in order to find out if the legs are at the required location on time.

Temperature Returns the temperature to give the locomotion module a signal if the motors become to warm.

Leg state The state of the leg gives information about the current situation the legs is in. For instance, if the leg encounters internal problems, it may stop functioning and will go in an emergency state.

If the legs function properly, the feedback of the location of the leg is necessary to keep on walking. If the robot's leg are constricted or malfunctioning in any way, the feedback of the legs can be combined in order to determine the current problem.

6-2 Locomotion program

In order to process the information received from the top-level controller, the locomotion module needs to take multiple steps. These steps are divided in blocks, which are explained in this section. Each part of the locomotion program is briefly explained here. The technical

report can be seen as the more detailed explanation, and can provide a manual to the use of the locomotion program.

Initialization

Before the ZeBRo can start its locomotion routine, connections between the Top-level controller, the IMU (inertia measurement unit) and the separate leg modules need to start. Furthermore, multiple different walking gaits are defined and calculated before the start of the walking sequence. This is done to decrease the workload of the processor while walking. The clock is also started for the internal time measurements.

Input processing

The input for the locomotion can come from three different sources:

Top-level The input received from the top-level include the direction and speed the ZeBRo is required to walk.

IMU The IMU constantly updates the body rotation of the ZeBRo. The body roll and body pitch are monitored to check whether the ZeBRo is scaling uneven surface.

Leg Modules The leg modules are checked for their position and temperature. The position is used for calculating possible delays, while the temperature is monitored in order to determine when the motors are overheating and the robot needs to cool down.

Vector and Event-list updating

Within the locomotion program, the system checks whether the Max-Plus A-matrix needs updating due to changing speed and direction requirements. The Max-Plus vectors are compared with the internal time measurements to calculate the new lift-off/touchdown vector when necessary. Using these vectors, an event-list is filled with the current and next events, containing the information whether movement to the lift-off or touchdown position is required, following the method proposed in Section 4-2-2.

Delay Processing

When legs do not reach their position on time, several steps need to be taken. First and foremost, the delay needs to be estimated. The estimation is done by dividing the angle which needs to be traveled by the maximal angular speed of the leg. With this estimation, the method of Section 4-2-3 can be used to determine the updated current lift-off/touchdown vector. This vector can then be used for the calculation of the next event vector.

Turning

Three different methods of turning are implemented, where the first method is used for turning on the spot. This is simple done by swapping the lift-off and touchdown position, and reversing the direction of actuation on the side the robot wants to turn.
For the gentle turning, both the method of changing the step sizes and the Max-Plus turning implementation have been tested. For more detailed information about the Max-Plus implementation for the turning, the exact method and the simulation results, the research of W. Suriana[23] can be read. Because of the limited implementation in the actual robots, several small tests were done in order to confirm the results of the simulations. Four different implementations of Suriana's method are tested, with the following gaits and gait parameters:

Tripod 1: $\tau_f = 0.8$, $\tau_g = 0.40$, $\tau_\delta = 0.10$, $\tau_p = 0.2$

Tripod 2: $\tau_f = 0.8$, $\tau_g = 0.40$, $\tau_\delta = 0.10$, $\tau_p = 0.4$

Tetrapod 1: $\tau_f = 0.8$, $\tau_g = 0.40$, $\tau_\delta = 0.10$, $\tau_p = 0.2$

Tetrapod 2: $\tau_f = 0.8$, $\tau_g = 0.40$, $\tau_\delta = 0.10$, $\tau_p = 0.4$

All these gaits are tested to go left and to go right, in order to see how the different parameters result in behavioral differences. This is especially interesting regarding the non-symmetrical tetrapod gait. The results can be seen in Table 6-1. The > 300 cm radius describes a turning radius way bigger than 3 meter, or almost walking straight.
It has to be noticed the tetrapod gait can be made up with different leg orders. In order to investigate the leg order on the performance, two tetrapod gaits are researched. The names in Equations 6-5 and 6-6 are given by respectively the left and right front foot that starts the gait.

$$\text{Tetrapod Left Gait} = \{1,6\} \prec \{2,3\} \prec \{4,5\} \tag{6-5}$$

$$\text{Tetrapod Right Gait} = \{2,5\} \prec \{1,4\} \prec \{3,6\} \tag{6-6}$$

Gait	Direction	Turning Radius [cm]	Expected Turning Radius [cm]
Tripod 1	Left	158 ± 21	176
Tripod 1	Right	160 ± 27	176
Tripod 2	Left	68 ± 10	108
Tripod 2	Right	87 ± 6	108
Tetrapod Left 1	Left	186 ± 29	324
Tetrapod Left 1	Right	> 300	324
Tetrapod Left 2	Left	188 ± 16	196
Tetrapod Left 2	Right	> 300	196
Tetrapod Right 1	Left	> 300	324
Tetrapod Right 1	Right	> 300	324
Tetrapod Right 2	Left	> 300	196
Tetrapod Right 2	Right	> 300	196

Table 6-1: Average turning radius with number of tests $n = 3$. After each test, the legs are re-calibrated.

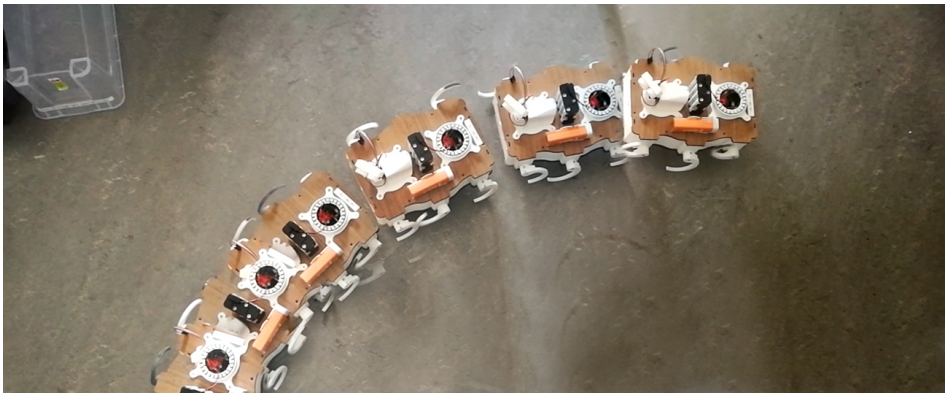


Figure 6-2: Example of the turning of the ZeBRo using Max-Plus based turning

As can be seen from the results, the effectiveness of turning using the Max-Plus implementation depends on the gait. Due to unwanted body movements in the tetrapod gaits, the left and right hind leg sometimes lose their contact with the ground, resulting in small changes within the yaw of the ZeBRo. This behavior was also noticed in the V-Rep tests, and is illustrated in Figure 4-6c. When the tripod gait is used however, the turning performance is more in accordance with the expected turning radius.

Output

The output of the locomotion module is defined by the communication with the leg-controllers and the top-level controller. For now, the leg controllers are updated regularly with information on the position they need to be at within a certain time. The frequency of updating the instructions depends on the required performance; if a gait is faster, the system might need more frequent updates regarding the timing. The output to the top-level is for now limited

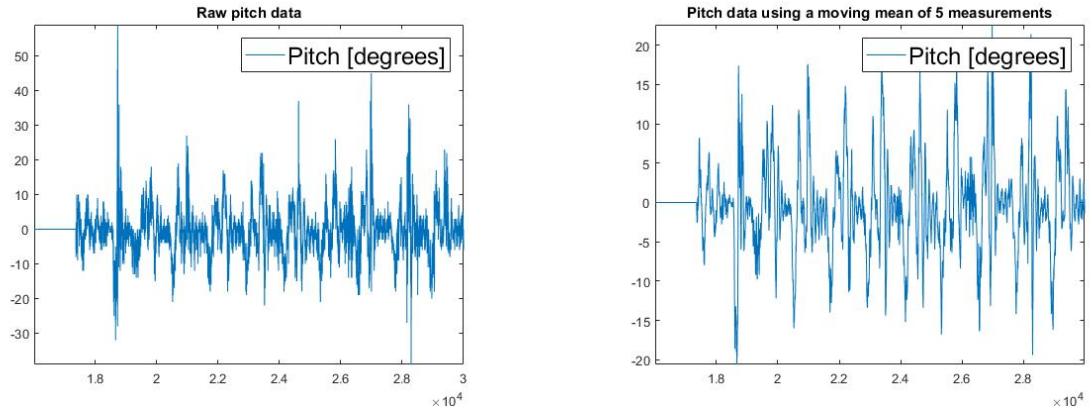
to communicating the state of the system, which indicates whether the ZeBRo is in rest or able to walk. The resting is mainly to prevent overheating of the leg motors.

6-3 Performance Gaits

For the verification of the results of Section 4-4, testing the body movement of the actual ZeBRo needs to be done. Testing is done by gathering the internal IMU data, while making the ZeBRo walk through several different situations. This is done to test for the optimal settings for the different gaits, while also checking their performance on the more rugged terrain. The performance is measured by using the success rate, the body rotations and the time of completing the track. The reason body rotations are used, and not the body accelerations is due to the direct coupling in the calculation of the accelerations and body rotations in the IMU. Using both metrics, the results would be similar. Each test is performed as many times until 5 successful runs have been recorded.

6-3-1 Flat Terrain

To check the performance of the different gaits on flat terrain, the gaits are not extensively tested on different settings. Using the τ_f, τ_g and τ_d, the approximate same speed is used to cover 2 meters on flat terrain. The resulting performance is mainly tested on body movement and speed of the ZeBRo. While processing the IMU-data, it became clear the noise of the system is relative high, due to the impact forces of every step. Therefore, a moving mean of the data of the last 5 data points is used to dismiss faulty measurements. An example of the difference between the measured and the processed data can be seen in Figure 6-3.



(a) Measured values of the pitch

(b) Moving mean values of the pitch

Figure 6-3: Difference between the measured values of the IMU (a) and the moving mean values (b) of the same test.

Where peaks of more than 50° can be seen in the unfiltered data, the moving mean data shows much more realistic value, as the pitch in this particular test did not actually exceed a body pitch of 20°.

Gait	Average standard deviation body roll	Average standard deviation body pitch	Average time elapsed for completion track
Tripod ($\tau_f = 0.7$, $\tau_g = 0.7$, $\tau_d = 0.05$)	0.126 ± 0.004 [rad]	0.048 ± 0.004 [rad]	7.23 [s]
Tetrapod ($\tau_f = 0.5$, $\tau_g = 0.5$, $\tau_d = 0.05$)	0.162 ± 0.006 [rad]	0.068 ± 0.010 [rad]	7.64 [s]
Metapod ($\tau_f = 0.4$, $\tau_g = 0.4$, $\tau_d = 0$)	0.121 ± 0.007 [rad]	0.079 ± 0.006 [rad]	8.05 [s]
Pentapod ($\tau_f = 0.3$, $\tau_g = 0.3$, $\tau_d = 0$)	0.113 ± 0.006 [rad]	0.095 ± 0.009 [rad]	7.14 [s]

Table 6-2: Average performance of the ZeBRo using different gaits on flat terrain (the office floor) on a 2-meter track with number of tests $n = 5$

As expected, the resulting body movement of every gait shows periodic behavior of steps being taken, which is most easy to spot in the body roll. The average standard deviation of the body pitch and roll is portrayed in Table 6-2, with an additional average time in which the 2 meter was completed.

As can be seen from the results, the body roll is relative small and the difference between the different gaits are small as well. It has to be noted that even though the completion times are similar in this test, the speed of the tripod gait can still be doubled, while the speed of the robot using the pentapod gait was nearly at its max.

6-3-2 Difficult Terrain

For the testing of the performance on more difficult terrain, and the resulting movement of the ZeBRo itself, a small test track has been made for benchmarking how difficult the terrain needs to be for changing the gait and walking methods. The test track contains multiple height differences along the way, about the same height as the legs (± 7 cm height differences). Additionally, a small incline and some moving objects are also placed in the test track.
The same gaits as for the flat terrain are used for the testing, with an additional fast walking tripod gait. This gait is added for checking the influence of the legs rotation speed on the system. It should be noted that the slow tripod ($\tau_f = 0.7$, $\tau_g = 0.7$, $\tau_d = 0.05$) was the only gait which failed 3 of its runs, where the rest of the gaits only failed one time.

The results are somewhat surprising, as the faster moving tripod gait shows less body rotational movement than the slower implementation. Additionally, the failure rate of the slower tripod makes for a interesting result, as the change of the flight, ground and double stance time seems to have a bigger impact on the performance than the gaits themselves.

Gait	Average standard deviation body roll	Average standard deviation body pitch	Average time elapsed for completion track
Tripod ($\tau_f = 0.7$, $\tau_g = 0.7$, $\tau_d = 0.05$)	0.334 ± 0.086 [rad]	0.275 ± 0.101 [rad]	8.54 [s]
Tripod ($\tau_f = 0.4$, $\tau_g = 0.4$, $\tau_d = 0$)	0.285 ± 0.024 [rad]	0.205 ± 0.020 [rad]	4.25 [s]
Tetrapod ($\tau_f = 0.5$, $\tau_g = 0.5$, $\tau_d = 0.05$)	0.256 ± 0.024 [rad]	0.136 ± 0.017 [rad]	11.5 [s]
Metapod ($\tau_f = 0.4$, $\tau_g = 0.4$, $\tau_d = 0$)	0.273 ± 0.024 [rad]	0.153 ± 0.012 [rad]	12.7 [s]
Pentapod ($\tau_f = 0.3$, $\tau_g = 0.3$, $\tau_d = 0$)	0.261 ± 0.026 [rad]	0.140 ± 0.015 [rad]	13.4 [s]

Table 6-3: Performance of the ZeBRo using different gaits on difficult terrain on a 2-meter track with number of tests $n = 5$

6-4 Gait Switch Function

In order to implement the switch decision function suggested in Section 4-4-5, the implementation in the actual system requires an IMU (Inertia Measurement Unit) to register the body rotations of the ZeBRo. The IMU is a relative simple and small component, and some also allow for measuring more data that could benefit the robots possibilities, like having a compass and magnetic field sensor on board.
The IMU is coupled to a simple Arduino board, which can read-out the data provided, and if necessary, can also make the required computations to check whether gait change is necessary. Communications with the Raspberry Pi will be required to transfer this information.
The registration of when the gait switch is needed, depends on the body rotation values over a certain amount of time. Because of the sensor inaccuracies of the ZeBRo, the relative simple gait change algorithm of Section 4-4-5 cannot be used, and therefore, a more robust implementation is required.
Using the data from both the flat terrain and the difficult terrain measurements, the major differences can be noticed quite easily. The average deviation with respect to the leveled body is bigger, and the (absolute) biggest measured rotations are bigger. In Table 6-4, the average of the ten biggest measured body rotations are portrayed. From this table, it can be clearly seen that the difference in measured body rotations is present. Therefore, the ZeBRo can use the measurements of the IMU in order to classify rough terrain.

However, when the robot walks on sloped terrain, there will be an offset for the system, even when the sloped terrain is relative flat. This leads to the conclusion that not only the absolute maximal values of the IMU-measurements need to be monitored, but the difference between the peaks and valleys in both directions.
Using this for the implementation of the gait switch function, the quality of the difficult terrain testing can be researched. For this test, the amount of good and false terrain identifiers need to be determined by testing the ZeBRo on different kind of surfaces.
Because of too many gait changes, the final ZeBRo has been implemented to have a dead-zone regarding gait changes. This results in less gait changing, along with less false positives.

Gait	Roll on flat terrain	Roll on difficult terrain	Pitch on flat terrain	Pitch on difficult terrain
Tripod ($\tau_f = 0.7$, $\tau_g = 0.7$, $\tau_d = 0.05$)	0.399 [rad]	1.03[rad]	0.231[rad]	1.37[rad]
Tetrapod ($\tau_f = 0.5$, $\tau_g = 0.5$, $\tau_d = 0.05$)	0.456[rad]	0.764[rad]	0.323[rad]	0.650[rad]
Metapod ($\tau_f = 0.4$, $\tau_g = 0.4$, $\tau_d = 0$)	0.402[rad]	0.906[rad]	0.412[rad]	0.744[rad]
Pentapod ($\tau_f = 0.3$, $\tau_g = 0.3$, $\tau_d = 0$)	0.348[rad]	0.802[rad]	0.405[rad]	0.704[rad]

Table 6-4: Comparison of the average of the ten (absolute) biggest body rotations on both the difficult and flat terrain with number of tests $n = 5$.

Difficult terrain instances	Correct difficult terrain identifiers	False difficult terrain identifiers
10	10	2

Table 6-5: Results of outdoor terrain recognition .

Because the tetrapod gait resulted in problems on flat terrain, resulting in false flags for difficult terrain, the pentapod gait is chosen for as the gait for difficult terrain, along with the gait transition described in Equation 5-43. When the ZeBRo does not detect large body movements for a certain amount of time, the switch back to the tripod (with the same gait transition) is made.

6-4-1 Difficult terrain recognition

During testing, the ZeBRo was taken out for difficult terrain in the outdoors. The test ground was mainly flat, with some obstacles littering about, like large branches and ditches. In 30 minutes of walking the ZeBRo encountered 10 instances of difficult terrain, which where all recognized (Table 6-5). However, during the walking, there were still several false positives.

In this test, the main result shows the superior performance of the tripod gait, which was performing better in the woods than the pentapod gait. Comparing this with the performance on the test track of Section 6-3-2, it only shows the importance of the terrain recognition. Using other feedback than strictly the body movement, more detailed terrain recognition can be utilized.

6-5 Conclusions

For the implementation of a total locomotion program using Max-Plus in a walking robot, the system contains all necessary requirements. The relative low computational power required for the updating of the timing of the legs, combined with the simplicity to change gait, the delay handling and turning makes for a powerful, complete locomotion program.
The results of the gait testing are interesting, as the gait performance shows to rely heavily

on the gait parameters, and less on the gaits themselves. For implementation purposes, this would mean the system is performing better while changing its gait when another speed is required, than simple changing the gait parameters.
Other interesting results, regarding the turning using Max-Plus gait change, shows to be a new, but reliable and powerful tool for complete robotic walking. Using this total system, a Max-Plus locomotion path planner can be build, and could be used for a lot of different legged robots, with a varying amount of legs. The turning remains somewhat more difficult to implement in the strict Max-Plus sense. Even though the Max-Plus turning shows great promise, the characteristics are not fully understood, and therefore, the implementation cannot be implemented for high precision walking. So currently, the ZeBRo remains equipped with a change of leg-angles, as the step size change (and consequently, the turning radius) can easily be calculated.
For now, due to implementation issues regarding the position feedback, the Max-Plus delay handling cannot be used as reliable as should be. The framework for dealing with the delays using the Max-Plus is however capable of correcting the walking when necessary, and makes the performance of the ZeBRo (when implemented), much more reliable.

Chapter 7

Simulating Running in the Zebro

The framework for constructing a wider arrangement of gaits is in place, which can now be used for running with the Zebro. Because the actual Zebro does not have enough power and speed to implement the running behavior, this is tested on the V-Rep model.
While the V-Rep model does not capture the correct physics of the actual Zebro, the performance can give an indication of how the system handles certain inputs, and how the limitations in the motors influence the possibilities the robots has for dynamic movement. For the determination of the gait parameters,

7-1 V-REP Simulations

In order to test the performance of the controller on the Zebro, it is chosen to first implement the controller in the V-Rep environment. Due to speed restrictions in the actuation, hopping with all legs is not possible, as the maximal allowed rotation speed of the motors in V-Rep would not be enough to recirculate before the robot falls down. This is in lesser extent also applies to the pacing and bounding. Therefore, the controller will be tested using the tripod gait, where the legs only need to recirculate half as much, and will not be affected by this speed restriction. The implementation is somewhat different than the implementation of the controller for the hopper. Instead of varying the length of the legs, the angles where the actuation happens change. For this test, the weight of the Zebro is altered to 2 kg. The legs however, remain rigid in this simulation, so the only results from these tests show the difference in actuation by the legs.

In order to test the performance of the controller, 4 different scenarios are tested:

- Dynamic Tripod without controller
- Dynamic Tripod with pitch controller

- Dynamic Tripod with roll controller
- Dynamic Tripod with full controller

In order to compare them, the distance traveled is compared. More importantly, the body rotations are also compared, using the variance with respect to the horizontal. It has to be noted that the system without controller is already stable. Therefore, the goal of the controller is to decrease the body rotations of the Zebro, and to reject potential disturbance.

The dynamic tripod used is given by:

$$\mathcal{G}_{dt} = \underset{\tau_{f1}}{\{1,4,5\}} \underset{\tau_{s1}}{\prec} \underset{\tau_{f2}}{\{2,3,6\}} \underset{\tau_{s2}}{\prec} \tag{7-1}$$

$$\tau_f = [0.2 \quad 0.2] \tag{7-2}$$

$$\tau_g = [0.1 \quad 0.1] \tag{7-3}$$

$$\tau_s = \begin{bmatrix} 0.1 & 0.1 \\ \epsilon & \epsilon \end{bmatrix} \tag{7-4}$$

These parameters are determined by trial and error, as the limitations regarding rotation speed impact the performance of the Zebro. It is chosen to take gait parameters which are in the range of the real gait parameters.

7-1-1 Flat Ground

To test the basic functionality of the controller the first stability tests are performed on flat ground, starting with a static tripod gait, the system runs 10 seconds with a running tripod gait. The implementation of the controller is relative simple, as it changes its lift-off and touchdown angles using the measurements of the V-Rep simulation of the body rotation. The full implementation will then result in the recalculation of the leg angles (θ_{lo} and θ_{td}) using the pitch and roll angles (θ_p and θ_r). These are stored in a vector for all separate legs (θ_{loC} and θ_{tdC}).

Pitch Controller:

$$\theta_{tdC} = \theta_{td} + k_1 \begin{bmatrix} -\theta_p & -\theta_p & 0 & 0 & \theta_p & \theta_p \end{bmatrix} \tag{7-5}$$

$$\theta_{loC} = \theta_{lo} - k_1 \begin{bmatrix} -\theta_p & -\theta_p & 0 & 0 & \theta_p & \theta_p \end{bmatrix} \tag{7-6}$$

Roll Controller:

$$\theta_{tdC} = \theta_{td} + k_2 \begin{bmatrix} \theta_r & -\theta_r & \theta_r & -\theta_r & \theta_r & -\theta_r \end{bmatrix} \tag{7-7}$$

$$\theta_{loC} = \theta_{lo} - k_2 \begin{bmatrix} \theta_r & -\theta_r & \theta_r & -\theta_r & \theta_r & -\theta_r \end{bmatrix} \tag{7-8}$$

Full Controller:

$$\theta_{tdC} = \theta_{td} + k_3 \begin{bmatrix} -\theta_p & -\theta_p & 0 & 0 & \theta_p & \theta_p \end{bmatrix} + k_4 \begin{bmatrix} \theta_r & -\theta_r & \theta_r & -\theta_r & \theta_r & -\theta_r \end{bmatrix} \tag{7-9}$$

$$\theta_{loC} = \theta_{lo} - k_3 \begin{bmatrix} -\theta_p & -\theta_p & 0 & 0 & \theta_p & \theta_p \end{bmatrix} - k_4 \begin{bmatrix} \theta_r & -\theta_r & \theta_r & -\theta_r & \theta_r & -\theta_r \end{bmatrix} \tag{7-10}$$

The $\{k_1, k_2, k_3, k_4\} \in \mathbb{R}$ are all different, in order to accommodate for the difference between the length of the side and front of the Zebro. Furthermore, the values of full controller are smaller than the values of the independent controllers, due to the possibility of a too large difference with the leveled leg angles, which influences the behavior negatively. The tests are performed multiple times. The most important the results can be seen in Table 7-1. Note that the traveled distance does not change with the changing of the controller. For comparison, the walking gait with almost the same parameters except for introducing double stance time only covers half the distance of the dynamic gait implementation.

Dynamic tripod gait	Average standard deviation body roll	Average standard deviation body pitch	Average distance
Without controller	0.101 ± 0.014 [rad]	0.135 ± 0.014 [rad]	7.74 [m]
With pitch controller	0.097 ± 0.029 [rad]	0.110 ± 0.028 [rad]	7.74 [m]
With roll controller	0.102 ± 0.017 [rad]	0.135 ± 0.016 [rad]	7.74 [m]
With full controller	0.096 ± 0.013 [rad]	0.119 ± 0.014 [rad]	7.74 [m]

Table 7-1: Performance of the Zebro using different controllers for the dynamic tripod gait

From the results, it is hard to conclude how much the controller contributes to the decrease of body rotations. However, the system shows to be slightly more leveled using the pitch and full controller. It is interesting to see the performance of the roll controller, which does not seem to have a significant impact on the response of the system. The performance of the controller might be more pronounced when disturbance is added, and more control action is needed.

7-1-2 Disturbance Rejection

Similar tests are performed with the addition of low bars across the field. These low bars (2 cm in height) are supposed to impose the system to slight disturbances in the hopping.

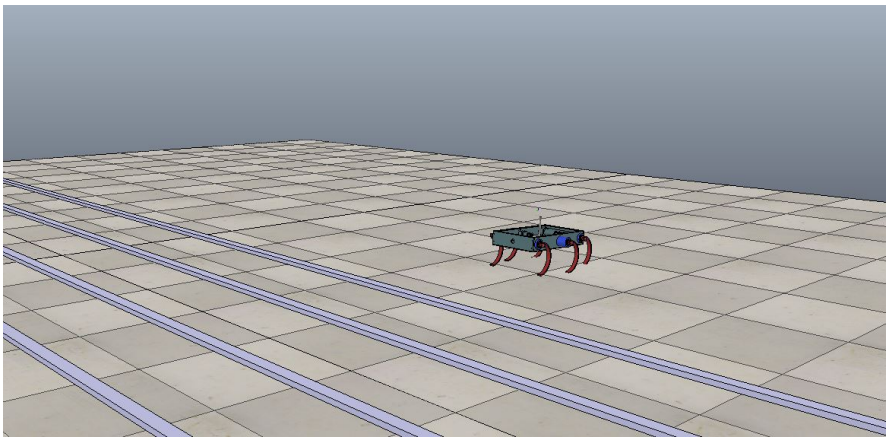


Figure 7-1: Disturbances added in the V-Rep simulation environment for the dynamic tripod gait

The results are not that different from the results of the test on flat ground. Note that the tests are performed in a shorter time period to capture a bigger part of the effects of the disturbance. The results of these tests can be seen in Table 7-2.

Dynamic tripod gait	Average standard deviation body roll	Average standard deviation body pitch	Average distance traveled
Without controller	0.106 ± 0.013 [rad]	0.150 ± 0.019 [rad]	5.94 [m]
With pitch controller	0.097 ± 0.018 [rad]	0.118 ± 0.019 [rad]	5.94 [m]
With roll controller	0.103 ± 0.015 [rad]	0.125 ± 0.021 [rad]	5.94 [m]
With full controller	0.111 ± 0.019 [rad]	0.133 ± 0.011 [rad]	5.94 [m]

Table 7-2: Disturbance rejection of the Zebro using different controllers for the dynamic tripod gait

Surprisingly, the controller with the overall best results is the pitch controller, which shows a very limited increase in body movement after the addition of the disturbances. The response of the full controller is somewhat disappointing, but still has a better performance than the uncontrolled system, especially regarding the disturbance rejection.
However, some connotations need to be placed with respect to the results of the simulations. The V-Rep simulation environment does not capture deformations of the legs of the Zebro, what can result in behavior which is not accurately describing the actual response of the system. Furthermore, the resulting distance being exactly the same every instance raises questions about the accuracy of the simulation, as the distance traveled does not seem to be changing between tests, which may be an indicator of the limited difference the controller makes.

Chapter 8

Conclusion

The main goal of this thesis has been to develop a complete, working locomotion system based on the switching Max-Plus discrete event systems. The locomotion system should be the basis of the walking DeciZeBRo, but should also be adaptable to other implementations of walking robots, with varying amounts of legs. Following this goal, the resulting product is a working locomotion program, which can be implemented in a microprocessor and is suitable for the use in embedded systems.
The contributions for the completion of this task let to a novel method of using the theoretical Max-Plus framework and the implementation of proven and unproven concepts of the Max-Plus framework as a whole:

1. A new method for the calculation of the timing of running gaits using the Max-Plus framework (Chapter 5)
2. Proving the working principles of the research of W. Suriana in an actual walking robot ([23], Chapter 6)
3. Delivering a complete locomotion system implementing optimal gait changing ([21]) and Max-Plus based turning ([23]) for the walking Deci-ZeBRo.

8-1 Max-Plus Complete Locomotion

In order to allow for constructing a wide array of gaits, the Max-Plus discrete event system for legged locomotion is extended to contain additional synchronization possibilities for the different leg-groups. These added synchronization is visible in the addition of the P_l and Q_l matrices, which allow for synchronization between the lift-offs of the consecutive leg-groups. The discrete event system is defined as:

$$x(k+1) = A_0 \otimes x(k+1) \oplus A_1 \otimes x(k) \tag{8-1}$$

With the matrices A_0 and A_1 defined by:

$$A_0 = \left(\begin{array}{c|c} \mathcal{E}_{n\times n} & T_f \\ \hline P_t & P_l \end{array}\right) \tag{8-2}$$

$$A_1 = \left(\begin{array}{c|c} E_{n\times n} & \mathcal{E}_{n\times n} \\ \hline T_g \oplus Q_t & E_{n\times n} \oplus Q_l \end{array}\right) \tag{8-3}$$

For this method to be implemented, τ_s is introduced, which contains the synchronization times τ_{sl} (the synchronization time between the lift-offs of the successive leg-groups), and τ_{st} (the synchronization time between the touchdown of leg-group i and lift-off of leg-group $i+1$).

$$\tau_{si} = \begin{pmatrix} \tau_{sli} \\ \tau_{sti} \end{pmatrix} = \begin{cases} \text{Synchronization times between leg-groups } i \text{ and } i+1 & \text{for } 0 < i < m \\ \text{Synchronization times between leg-groups } m \text{ and } 1 & \text{for } i = m \end{cases} \tag{8-4}$$

The addition of this set of synchronizations also allows for better delay handling in the more complex gaits, as the synchronization can be doubled to account for both delays in the lift-off and touchdown. The construction of the P_l, P_t,Q_l and Q_t matrices remains similar to the construction of the P and Q matrices of the research of [4]: Let P_l, P_t, Q_l and Q_t be square matrices of the size $n \times n$ with n the number of legs. To define P_l and P_t, we use $j = \{1, ..., m-1\}$, with m the amount of leg groups, $\forall p \in l_{j+1}$ and $\forall q \in l_j$, where p and q represent the separate legs in the leg-groups.

$$P^l_{x,y} = \begin{cases} \tau_{slj} & \text{for} \quad P^l_{p,q} \\ \epsilon & \text{for every other} \quad P^l_{x,y} \end{cases} \tag{8-5}$$

$$P^t_{x,y} = \begin{cases} \tau_{stj} & \text{for} \quad P^t_{p,q} \\ \epsilon & \text{for every other} \quad P^t_{x,y} \end{cases} \tag{8-6}$$

To determine Q_l and Q_t, for $\forall v \in l_1$ and $\forall w \in l_m$:

$$Q^l_{x,y} = \begin{cases} \tau_{slm} & \text{for} \quad Q^l_{v,w} \\ \epsilon & \text{for every other} \quad Q^l_{x,y} \end{cases} \tag{8-7}$$

$$Q^t_{x,y} = \begin{cases} \tau_{stm} & \text{for} \quad Q^t_{v,w} \\ \epsilon & \text{for every other} \quad Q^t_{x,y} \end{cases} \tag{8-8}$$

The introduction of the lift-off based synchronization allows for more complete delay handling. It also allows the system to make delay windows, which allow small delays without changing the gait pattern. This allows for easier controller implementation and more control over the gaits.

8-2 Locomotion and gait switching

One of the best qualities of a Max-Plus algebra based locomotion system is the portability. For example, even though the system is currently used for six-legged robots, small changes can make the system perform as well on robots with any amount of legs, purely by changing the leg-group order and the size of the sub-matrices of the A_0 and A_1 matrices. While other robots depend on other types of the central pattern generators for the complete movement of the system, which require (manual) changes for other walking rhythms, the Max-Plus timing can easily be changed for different types of locomotion systems.
The main downside of the Max-Plus generation is that the information is relative limited (containing the lift-off/touchdown events), and additional movement details regarding how to perform a step has to be made more concrete in layers below the Max-Plus system. This does allow for low computational strain on the system, which is especially useful in small embedded systems, where walking is a sublayer of the system.
The switching function $\mu(k)$ depends on the dimensions of the platform implemented on. The main use for optimal gait change using Max-Plus in legged locomotion, and to a certain extent the whole use of the Max-Plus framework for legged locomotion, is the ability to quickly and efficiently change gaits. This extends from changing the order of leg groups, to the change of the individual parameters and the speed of the platform.
One of the main purposes however, is to effectively use the gait switch function. Because of limitations in feedback of the current Deci-ZeBRo, the information available for gait change is relatively limited. The framework for gait changing however, is fully functional and able to change the walking gaits on the fly, maintaining stability and working principles.
For now, the main gait change parameters considered are the body movement of the ZeBRo and resulting gait performance when terrain is detected that influences the walking. The resulting effects of the gait change in difficult situations show an increase in stability of the Deci-ZeBRo, and make the robot able to conquer heavy terrain better.

8-3 Max-Plus based turning

For more information about the Max-Plus based turning, which is researched by W. Suriana, the research [23] can be viewed. The performance of this method relies on creating a difference in speed between the two different sides of the walking robot, by extending the ground and double stance time of the legs on one side, and decreasing the ground and double stance time on the other side. The side on which the ground time is decreased, the speed of the steps itself is increased, which results in a bigger distance traveled. The result of the tests done can be seen in Table 8-1, the explanation of the tested gaits can be seen in Section 6-2.

Gait	Direction	Turning Radius	Expected Turning Radius
Tripod 1	Left	158 ± 21 cm	1.76 cm
Tripod 1	Right	160 ± 27 cm	1.76 cm
Tripod 2	Left	68 ± 10 cm	1.08 cm
Tripod 2	Right	87 ± 6 cm	1.08 cm
Tetrapod Left 1	Left	186 ± 29 cm	3.24 cm
Tetrapod Left 1	Right	>300 cm	3.24 cm
Tetrapod Left 2	Left	188 ± 16 cm	1.96 cm
Tetrapod Left 2	Right	> 300 cm	1.96 cm
Tetrapod Right 1	Left	> 300 cm	3.24 cm
Tetrapod Right 1	Right	> 300 cm	3.24 cm
Tetrapod Right 2	Left	> 300 cm	1.96 cm
Tetrapod Right 2	Right	> 300 cm	1.96 cm

Table 8-1: Average turning radius with number of tests $n = 3$. After each test, the legs are re-calibrated.

Because the results are ambiguous, and the Max-Plus based turning does not work for the tetrapod gait under normal circumstances, it is for now chosen to have the legs slightly increase/decrease their step size for the turning. The Max-Plus based turning is however a very promising method for changing direction, and if researched further, might be a great alternative for the method with the variable step size. Currently, using a slightly modified locomotion, one of the ZeBRos is equipped with the Max-Plus turning. This robot is performing similar to the other ZeBRos.

8-4 Discussion and recommendations

With the extension of the Max-Plus locomotion framework, almost all situations can now be handled with the implementation of certain aspects of this system. Adding the possibility for a wider array of gaits, including running, and more synchronization options for the determination of the lift-off of the legs, allows for using Max-Plus in a great amount of different walking robots. Additionally, the Max-Plus implementation of the turning, following the research of [23], even extends this to make a complete path planning algorithm using Max-Plus possible. The handling of delays is one of the most powerful parts of using Max-Plus in the walking robot, and can be a major reason for the decision to use Max-Plus instead of other central pattern generation.
However, the main downside of the Max-Plus system is the relative big computational backbone that needs to be in place for the system to work. This is mainly caused by the calculation of the A-matrix, which requires a lot of calculations for the determination of the A_0^*. For embedded real-time calculations of this matrix, the calculation time may intervene the standard locomotion pattern. This can however be countered by calculating the gait matrices before walking.
The framework of the implementation of the Max-Plus locomotion system on a walking robot

also requires a centralized locomotion system, with a library which contains the necessary functions for Max-Plus algebra, gait-matrix construction and event vector processing. This requires a lot more overhead than applying time-based functions for the separate legs. Gait transitions, gait parameter change and delay handling make up for these shortcomings, but require relative accurate leg control to completely grasp the full possibilities.

8-4-1 Further research recommendations

- Research of a Max-Plus based path planning algorithm, using Max-Plus based turning, applying the research of W. Suriana [23]. This requires solid understanding of the mechanics of the robot, and the consequences of extending double stance times. The complete turning characteristics need to be understood, and implementation of this system on a actual robot with extended path planning need to be executed for testing. This path planning can be done by desiring separate speeds for the left and right side of the body. Coupled to the mechanics of the robot, this allows for specific turning radii.
- Applying Max-Plus running gaits in actual robots, using the full dynamics of the body and legs to increase the possible speed of the robot. Not only new (springy) legs are required for this to work, but the actuation of the robot legs need to be fast and strong enough for the robot to achieve certain accelerations and leg rotation speeds. Applying this would an increase in the cost of the robot, due to the costs of a more powerful set of electric motors. Additionally, the design of a controller which can be used for handling body rotations during different running gaits, like hopping or skipping, can be added and tested.
- Optimize gaits for certain environments, by applying self learning algorithms and allowing the robots to walk in these environments by themselves, applying the research of D. van Amstel [22]. This requires the addition of power and velocity measurements in the ZeBRo, and should optimize for both these parameters. The environment detection could be coupled to this.
- Use other methods for environment detection, by having more input from different sensors. This can include cameras, power consumption measurements or sound detectors. From the results of this thesis, it shows that adaptive gait patterns can increase the stability of the system as a whole. Because the inertia measurement unit can only be used after the surface has changed, an environment detection system which uses (stereo) cameras increases the detectability of more separate environments.

Appendix A

Max-Plus examples

In this part of the appendix, examples of calculations using Max-Plus algebra can be seen.

A-1 Basics

A-1-1 Basic operations

Some basic calculations of Max-Plus algebra can be seen below:

Max-Plus addition

$$5 \oplus 7 = 7 \tag{A-1}$$
$$2 \oplus -1 = 2 \tag{A-2}$$
$$-1 \oplus -5 = -1 \tag{A-3}$$

Max-Plus multiplication

$$4 \otimes 5 = 9 \tag{A-4}$$
$$3 \otimes -2 = 1 \tag{A-5}$$
$$-2 \otimes -3 = -5 \tag{A-6}$$

Commutativity

$$1 \oplus 2 = 2 \oplus 1 = \max(2, 1) = 2 \tag{A-7}$$
$$1 \otimes 2 = 2 \otimes 1 = 1 + 2 = 2 + 1 = 3 \tag{A-8}$$

Associativity

$$2 \oplus (4 \oplus 7) = \max(2, \max(4, 7)) = 7 \tag{A-9}$$
$$(2 \oplus 4) \oplus 7 = \max(\max(2, 4), 7) = 7 \tag{A-10}$$
$$3 \otimes (4 \otimes 7) = (3 \otimes 4) \otimes 7 = 3 + 4 + 7 = 14 \tag{A-11}$$

Distributivity:

$$2 \otimes (3 \oplus 4) = 2 + \max(3, 4) = 2 + 4 = 6 \tag{A-12}$$
$$(2 \otimes 3) \oplus (2 \otimes 4) = \max(2 + 3, 2 + 4) = \max(5, 6) = 6 \tag{A-13}$$

A-1-2 Matrix calculations

Matrix addition:

$$\begin{bmatrix} 2 & 1 \\ 4 & 2 \end{bmatrix} \oplus \begin{bmatrix} 3 & 1 \\ 2 & 8 \end{bmatrix} = \begin{bmatrix} \max(3,2) & \max(1,1) \\ \max(4,2) & \max(2,8) \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 4 & 8 \end{bmatrix} \tag{A-14}$$

Matrix-vector Multiplication:

$$\begin{bmatrix} 6 & 1 & 5 \\ 2 & 4 & 3 \end{bmatrix} \otimes \begin{bmatrix} 3 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 6 \otimes 3 \oplus 1 \otimes 2 \oplus 5 \otimes 2 \\ 2 \otimes 3 \oplus 4 \otimes 2 \oplus 3 \otimes 2 \end{bmatrix} = \begin{bmatrix} \max(6+3, 1+2, 5+2) \\ \max(2+3, 4+2, 3+2) \end{bmatrix} = \begin{bmatrix} 9 \\ 6 \end{bmatrix} \tag{A-15}$$

Matrix multiplication

$$\begin{bmatrix} 6 & 1 \\ 2 & 4 \end{bmatrix} \otimes \begin{bmatrix} 3 & 1 & 4 \\ 1 & 2 & 4 \end{bmatrix} = \begin{bmatrix} 6 \otimes 3 \oplus 1 \otimes 1 & 6 \otimes 1 \oplus 1 \otimes 2 & 6 \otimes 4 \oplus 1 \otimes 4 \\ 2 \otimes 3 \oplus 4 \otimes 1 & 2 \otimes 1 \oplus 4 \otimes 2 & 2 \otimes 4 \oplus 4 \otimes 4 \end{bmatrix} \tag{A-16}$$

$$= \begin{bmatrix} \max(6+3, 1+1) & \max(6+1, 1+2) & \max(6+4, 1+4) \\ \max(2+3, 4+1) & \max(2+1, 4+2) & \max(2+4, 4+4) \end{bmatrix} = \begin{bmatrix} 9 & 7 & 10 \\ 5 & 6 & 8 \end{bmatrix} \tag{A-17}$$

A-1-3 Eigenvalues & Eigenvectors

Define matrix A and vector x_0 as:

$$A = \begin{pmatrix} 2 & 6 & \epsilon \\ 5 & \epsilon & 3 \\ \epsilon & 2 & \epsilon \end{pmatrix} \tag{A-18}$$

$$x_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{A-19}$$

Now multiply x_0 several times with A, we get the following sequence:

$$x_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad x(1) = \begin{pmatrix} 6 \\ 5 \\ 2 \end{pmatrix}, \quad x(2) = \begin{pmatrix} 11 \\ 11 \\ 7 \end{pmatrix}, \quad x(3) = \begin{pmatrix} 17 \\ 16 \\ 13 \end{pmatrix}, \quad x(4) = \begin{pmatrix} 22 \\ 22 \\ 18 \end{pmatrix} \tag{A-20}$$

Now, to check whether there is a returning periodic behaviour, the difference between each instance of x is calculated:

$$\Delta x(1) = \begin{pmatrix} 6 \\ 5 \\ 2 \end{pmatrix}, \quad \Delta x(2) = \begin{pmatrix} 5 \\ 6 \\ 5 \end{pmatrix}, \quad \Delta x(3) = \begin{pmatrix} 6 \\ 5 \\ 6 \end{pmatrix}, \quad \Delta x(4) = \begin{pmatrix} 5 \\ 6 \\ 5 \end{pmatrix} \tag{A-21}$$

As can be seen, after the first step, the system shows periodic behaviour with a period of 2 time steps. Now, if we calculate the increase in one period, we can define the eigenvalue:

$$x(4) = 11 \otimes x(2) \tag{A-22}$$

Now, the eigenvalue is calculated by dividing the increase in one period by the duration of one period: $\lambda = \frac{11}{2} = 5.5$. The eigenvector is then calculated as:

$$v = \frac{1}{2} \begin{bmatrix} 17 \oplus 11 \\ 16 \oplus 11 \\ 13 \oplus 7 \end{bmatrix} = \begin{bmatrix} 14 \\ 13.5 \\ 10 \end{bmatrix} \tag{A-23}$$

Now, to check whether the eigenvector and eigenvalue are correct:

$$A \otimes v = \begin{bmatrix} 19.5 \\ 19 \\ 15.5 \end{bmatrix} = \lambda \otimes v \tag{A-24}$$

A-2 Graphs and Petri-nets

Two examples regarding graphs and Petri-nets can be followed in this section.

A-2-1 Graphs

To explain the use of graphs with respect to Max-Plus algebra further, the example in Figure A-1 will be regarded as a (overly) simplified hypothetical train network, where the nodes signify stations and the routes signify the distance between the stations.

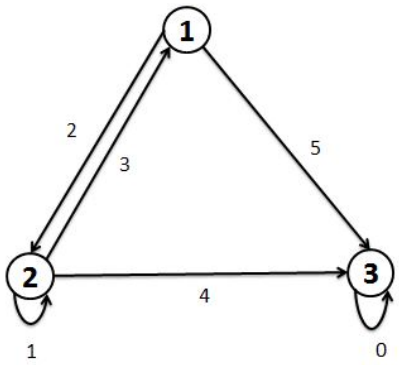


Figure A-1: Example of a directed weighted graph

In this system, it is possible to go from station 1 to station 2, from station 2 to station 1, to change tracks in station 2 (which is the distance it takes to go from station 2 to station 2) and to go from all stations to station 3. The next trains can depart when all incoming trains have arrived. Therefore, it is important to have a good overview of moment that the last trains have arrived.

Now, imagine having enough trains at every station. The moment they are ready to departure from their current location can be described in a vector $x(k)$, where k is the current iteration. For calculation purposes, we define $x(1)$ as:

$$x(1) = \begin{bmatrix} 2 \\ 1 \\ 8 \end{bmatrix} \tag{A-25}$$

Now, using the matrix of 2-28, the arrival times of the **latest** trains ($y(k)$) can be calculated by using Max-Plus matrix multiplication:

$$y(1) = A \otimes x(1) = \begin{bmatrix} \epsilon & 3 & \epsilon \\ 2 & 1 & \epsilon \\ 5 & 4 & e \end{bmatrix} \otimes \begin{bmatrix} 2 \\ 1 \\ 8 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 8 \end{bmatrix} \tag{A-26}$$

A-2-2 Paths

For the determination of the weight of a certain path with a certain length, the Max-Plus algebraic matrix power is used. First, we want to calculate the weight of the longest path from node 2 to node 3 in 4 steps:

$$A^{\otimes 4} = \begin{pmatrix} 10 & 9 & \epsilon \\ 8 & 7 & \epsilon \\ 11 & 13 & e \end{pmatrix} \tag{A-27}$$

$$|p_{ij}^k|_w = |p_{2,3}^4|_w = [A^{\otimes 4}]_{3,2} = 13 \tag{A-28}$$

$$|p_{2,3}^4|_{aw} = \frac{|p_{2,3}^4|_w}{|p_{2,3}^4|_l} = \frac{13}{4} = 3.25 \tag{A-29}$$

And indeed, the longest path from node 2 to node 3 is the sequence:

$$\text{node } 2 \xrightarrow{3} \text{node } 1 \xrightarrow{2} \text{node } 2 \xrightarrow{3} \text{node } 1 \xrightarrow{5} \text{node } 3 \tag{A-30}$$

Now, to define the circuit from node 1 to node 1 in three steps, we use the same method as before:

$$A^{\otimes 4} = \begin{pmatrix} 6 & 8 & \epsilon \\ 4 & 6 & \epsilon \\ 10 & 9 & e \end{pmatrix} \tag{A-31}$$

$$|p_{ij}^k|_w = |p_{1,1}^3|_w = [A^{\otimes 3}]_{1,1} = 6 \tag{A-32}$$

$$|p_{1,1}^3|_a w = \frac{|p_{1,1}^3|_w}{|p_{1,1}^3|_l} = \frac{6}{3} = 2 \tag{A-33}$$

Where again, the path in three steps is given by:

$$\text{node } 1 \xrightarrow{2} \text{node } 2 \xrightarrow{1} \text{node } 2 \xrightarrow{3} \text{node } 1 \tag{A-34}$$

A-3 Gait Matrices P and Q

Following the procedure of Section 3-2, a fully worked out example may improve the understanding of the generation of the P and Q matrix.
For demonstration purposes, a 5-legged gait in a 6 legged system will be explained. This may be necessary for situations where one of the legs is not functioning properly. The gait will be defined by:

$$\mathcal{G}_{r5} = \{1, 3\}, \{4, 6\}, \{5\} \tag{A-35}$$

For the observant reader, leg 2 is indeed missing. The actual performance of this gait would be questionable at best. This will not influence the method of constructing the P and Q

matrices.
Now, define the number of leg groups $m = 3$ and number of legs $n = 6$. j is defined as:

$$j = \{1, 2\} \tag{A-36}$$

The groups of legs are defined as:

$$l_1 = \{1, 3\} \tag{A-37}$$
$$l_2 = \{4, 6\} \tag{A-38}$$
$$l_3 = \{5\} \tag{A-39}$$

Now first, for every j, the p_j and q_j are defined following Equations 3-16 & 3-17:

$$p_1 = \{4, 6\} \qquad p_2 = \{5\} \tag{A-40}$$
$$q_1 = \{1, 3\} \qquad q_2 = \{4, 6\} \tag{A-41}$$

So, the elements of P that are equal to τ_δ are:

$$\text{For j=1:} \quad P_{4,1} = \tau_\delta \qquad P_{4,3} = \tau_\delta \tag{A-42}$$
$$P_{6,1} = \tau_\delta \qquad P_{6,3} = \tau_\delta \tag{A-43}$$
$$\text{For j=2:} \quad P_{5,4} = \tau_\delta \qquad P_{5,6} = \tau_\delta \tag{A-44}$$

Using the information of Equations A-42 - A-44, combined with the 6-legged system, we get:

$$P_{6\times6} = \begin{pmatrix} \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \tau_\delta & \epsilon & \tau_\delta & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \tau_\delta & \epsilon & \tau_\delta \\ \tau_\delta & \epsilon & \tau_\delta & \epsilon & \epsilon & \epsilon \end{pmatrix} \tag{A-45}$$

Now, for Q, we want to define the v and w.

$$v = \{1, 3\} \qquad w = \{5\} \tag{A-46}$$

So again, elements of Q that are equal to τ_δ are:

$$Q_{1,5} = \tau_\delta \qquad Q_{3,5} = \tau_\delta \tag{A-47}$$

Now, the Q matrix can be constructed:

$$Q_{6\times 6} = \begin{pmatrix} \epsilon & \epsilon & \epsilon & \epsilon & \tau_\delta & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \tau_\delta & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \end{pmatrix} \tag{A-48}$$

Example of dynamic walking with respect to static walking

For comparison, a two-legged system is used (for example, a human!). First, standard walking with double stance time will be implemented. After this, the A-matrix for running will be made. Following the standard procedure, the P and Q matrices are generated as:

$$P_s = \begin{pmatrix} -\infty & -\infty \\ \tau_d & -\infty \end{pmatrix} \qquad Q_s = \begin{pmatrix} -\infty & \tau_d \\ -\infty & -\infty \end{pmatrix} \tag{A-49}$$

$$P_d = \begin{pmatrix} -\infty & -\infty \\ \tau_a & -\infty \end{pmatrix} \qquad Q_d = \begin{pmatrix} -\infty & \tau_a \\ -\infty & -\infty \end{pmatrix} \tag{A-50}$$

Plugging in these matrices in Equation 3-7 for the walking gait and in Equation 5-5 for the running gait results in Equations A-51 and A-52 respectively:

$$x(k+1) = \left(\begin{array}{cc|cc} -\infty & -\infty & \tau_f & -\infty \\ -\infty & -\infty & -\infty & \tau_f \\ \hline -\infty & -\infty & -\infty & -\infty \\ \tau_d & -\infty & -\infty & -\infty \end{array}\right) \otimes x(k+1) \oplus \left(\begin{array}{cc|cc} 0 & -\infty & -\infty & -\infty \\ -\infty & 0 & -\infty & -\infty \\ \hline \tau_g & \tau_d & 0 & -\infty \\ -\infty & \tau_g & -\infty & 0 \end{array}\right) \otimes x(k) \tag{A-51}$$

$$x(k+1) = \left(\begin{array}{cc|cc} -\infty & -\infty & -\infty & -\infty \\ -\infty & -\infty & \tau_a & -\infty \\ \hline \tau_g & -\infty & -\infty & -\infty \\ -\infty & \tau_g & -\infty & -\infty \end{array}\right) \otimes x(k+1) \oplus \left(\begin{array}{cc|cc} 0 & -\infty & \tau_f & \tau_a \\ -\infty & 0 & -\infty & \tau_f \\ \hline -\infty & -\infty & 0 & -\infty \\ -\infty & -\infty & -\infty & 0 \end{array}\right) \otimes x(k) \tag{A-52}$$

Bibliography

[1] R. Institute, "Zebro." https://tudelftroboticsinstitute.nl/robots/zebro/, 2018. [Online; accessed 22-May-2018].

[2] U. Saranli, M. Buehler, and D. E. Koditschek, "Rhex: A simple and highly mobile hexapod robot," tech. rep., 2001.

[3] M. Otten, "Decizebro: the design of a modular bio-inspired robotic swarming platform," Master's thesis, TU Delft, 2017.

[4] G. Lopes, B. Kersbergen, T. van den Boom, B. de Schutter, and R. Babuska, "Modeling and control of legged locomotion via switching max-plus models," in *IEEE Transactions on Robotics*.

[5] BostonDynamics, "Boston dynamics robots," January 2018. https://www.bostondynamics.com/robots.

[6] BostonDynamics, "Boston dynamics atlas," January 2018. https://www.bostondynamics.com/atlas.

[7] BostonDynamics, "Boston dynamics spot," January 2018. https://www.bostondynamics.com/spot.

[8] BostonDynamics, "Boston dynamics rhex," January 2018. https://www.bostondynamics.com/rhex.

[9] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: system overview and integration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2478–2483 vol.3, 2002.

[10] J. de Bruijckere, R. Smit, and L. R. Valk, "Zebro light: Lightweight mobile hexapod robot," tech. rep., TU Delft, 2012.

[11] M. AE., "The biomechanics of skipping gaits: a third locomotion paradigm?," in *Proceedings of the Royal Society B: Biological Sciences 265*.

[12] J. Seipel and P. Holmes, "The intelligent asimo: system overview and integration," in *Regular and Chaotic Dynamics*, vol. 12, pp. 502–520, 2007.

[13] H. Yu, H. Gao, L. Ding, M. Li, Z. Deng, and G. Liu, "Gait generation with smooth transition using cpg-based locomotion control for hexapod walking robot," *IEEE Transactions on Industrial Electronics*, vol. 63, pp. 5488–5500, Sept 2016.

[14] B. de Schutter and T. van den Boom, "Max-plus algebra and max-plus linear discrete event systems: An introduction," in *Proceedings of the 9th International Workshop on Discrete Event Systems*, no. 36-42, IEEE, 28-30 May 2008.

[15] N. C. Heglund and C. R. Taylor, "Speed, stride frequency and energy cost per stride: how do they change with body size and gait?," *Journal of Experimental Biology*, vol. 138, no. 1, pp. 301–318, 1988.

[16] H. G. R. S. K. D. S. A. Wilshin S, Reeve MA, "Longitudinal quasi-static stability predicts changes in dog gait on rough terrain.," *The Journal of Experimental Biology*, vol. 220, no. 10, pp. 1864–1874, 2017.

[17] J. Braker and G. Olsder, "The power algorithm in max algebra," *LINEAR ALGEBRA AND ITS APPLICATIONS*, vol. 182.

[18] G. Haynes and A. Rizzi, "Gaits and gait transitions for legged robots," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on.*

[19] G. Lopes, T. van den Boom, B. D. Schutter, and R. Babuska, "Switching max-plus models for legged locomotion," in *Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics (ROBIO 2009).*

[20] G. A. D. Lopes, B. Kersbergen, B. De Schutter, T. van den Boom, and R. Babuška, "Synchronization of a class of cyclic discrete-event systems describing legged locomotion," *Discrete Event Dynamic Systems*, vol. 26, pp. 225–261, Jun 2016.

[21] B. Kersbergen, G. Lopes, T. van den Boom, B. D. Schutter, and R. Babuska, "Optimal gait switching for legged locomotion," in *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11).*

[22] D. D. van Amstel, "Feedback mechanisms in switching max-plus linear systems," Master's thesis, TU Delft, 2014.

[23] W. Suriana, "Turning of a legged robot via a switching max-plus linear system," Master's thesis, TU Delft, 2017.

[24] CoppeliaRobotics, "Coppelia robotics v-rep," January 2018. http://www.coppeliarobotics.com/.

[25] M. Shahbazi Aghbelagh, *A template-based control architecture for dynamic legged locomotion.* PhD thesis.

[26] G. O. F. Bacceli, G. Cohen and J. Quadrat, *Synchronization and Linearity: An Algebra for Discrete Event Systems*, ch. 10. John Wiley and Sons Ltd, 1st ed., October 1992.

[27] N. Deuel and L. Lawrence, "Gallop velocity and limb contact variables of quarter horses," *Journal of Equine Veterinary Science*, vol. 6, no. 3, pp. 143 – 147, 1986.

[28] R. Blickhan, "The spring-mass model for running and hopping," in *J. Biomechanics*, vol. 22, pp. 1217–1227, 1989.

[29] K.-J. H. Chun-Kai Huang and P.-C. Lin, "Rolling slip model based running on a hexapod robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013.

[30] M. Shahbazi and G. A. D. Lopes, "A max-plus based synchronization controller for multiple spring-mass hoppers," in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 1669–1674, Dec 2015.