

# Design and implementation of a control structure for a quadruped robot

C. (Chen) Ling

MSc Report

**Committee:**

Prof.dr.ir. S. Stramigioli  
D. Dresscher, MSc  
Dr.ir. J.F. Broenink  
Dr.ir. J. van Dijk

December 2015

038RAM2015  
Robotics and Mechatronics  
EE-Math-CS  
University of Twente  
P.O. Box 217  
7500 AE Enschede  
The Netherlands

# Summary

Quadruped robots have several potential advantages compared to wheeled and tracked ones. For example, they have better performance when it comes to locomotion over difficult terrain. In this research, the project focuses on the design and integration of several components to complete a control structure for a quadruped robot. Previous work includes the development of one of these components: the gait generator (van der Coelen. 2013). The gait generator calculates which leg should make a step at which point in time while the robot remains stable throughout the walking motion.

This research involves the design, implementation and testing of the other components of the control structure. The components can be divided into three subsystems. One of them is the “transfer leg” system. There are three components in this system: the foot trajectory generator, the foot controller and the Jacobian relations. The foot trajectory generator generates a step motion profile to transfer a foot. The foot controller controls the foot on the foot trajectory when required. And the Jacobian relations convert the force in legs to all the joints and convert the joint velocities to foot velocity. The second subsystem is the system of the support legs. In this system, Generalised inverse Jacobian maps a force applied on the robot body to torques that are required in the joints for the support legs. The third subsystem is the mode switcher. The mode switcher assures the robot legs switch from transfer leg to support leg based on the leg index signal that is calculated by the gait generator.

The implementations of these subsystems have been done in this research. The simulation results of the “transfer leg” subsystem show that the leg can move to the desired position in its reachable area. The recommendations are presented at the end of this report.

# Content

<b>Chapter 1 Introduction .....</b>	<b>1</b>
1.1 Approach and Goal.....	1
1.2 Tasks .....	2
1.3 Outline.....	3
<b>Chapter 2 Background .....</b>	<b>4</b>
2.1 Robot Model.....	4
2.2 Gait Generator.....	4
2.3 Homogeneous matrices.....	5
2.4 Twists and Wrenches.....	6
<b>Chapter 3 Design of components .....</b>	<b>8</b>
3.1 Foot trajectory generator .....	9
3.1.1 Analysis .....	9
3.1.2 Design.....	12
3.1.3 Simulation and evaluation.....	14
3.2 Feet controllers .....	16
3.2.1 Design.....	16
3.2.2 Simulation and evaluation.....	18
3.3 Jacobian relation .....	20
3.3.1 Design.....	21
3.3.2 Simulation and evaluation.....	23
3.4 Generalized inverse Jacobian relation .....	26
3.4.1 Design.....	26
3.4.3 Simulation and evaluation.....	30
3.5 Conclusions .....	33
<b>Chapter 4 Integration .....</b>	<b>34</b>
4.1 Integration of the components for a transfer leg .....	34
4.1.1 Implementation .....	35
4.1.1.2 Simulation parameters.....	35
4.2 Integration of the components for supporting the robot body .....	37
4.2.1 Implementation .....	<b>Error! Bookmark not defined.</b>
4.3 Integration of the entire quadruped robot .....	37
<b>Chapter 5 Conclusions and recommendation.....</b>	<b>39</b>
5.1 Conclusions .....	39
5.2 Recommendations.....	39
<b>References.....</b>	<b>40</b>



# Chapter 1 Introduction

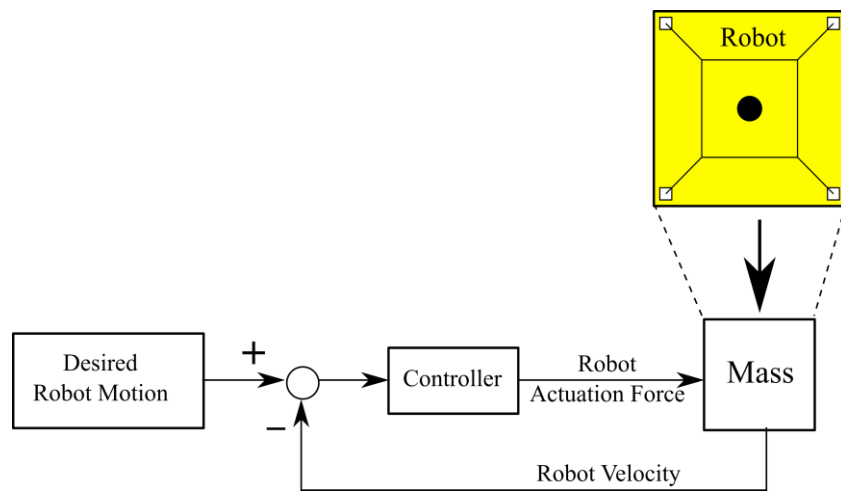
For traversing unknown terrains, legged robots are preferable compared to their wheeled and tracked counterparts. For example, when moving over uneven terrain, legged robots can keep the motion of the robot body fluid while wheeled and tracked robots would experience jerky disturbance. Another advantage is that in tight spaces, legged robots can change the direction of movement without changing the direction where its main body is facing to.

A legged robot can have any number of legs, but minimizing the number of legs would also reduce the mechanical complexity and weight of robot itself. A minimum of four legs is required to retain the ability for statically stable locomotion (McGhee and Frank, 1968).

As part of the ROSE project (Dresscher and Stramigioli, 2012), a legged robot is needed to traverse difficult terrain (e.g. dikes) and carry sensitive measurement equipments. And for the reason pointed out above, a quadruped robot was chosen. In the work described in this report, a control structure for such a robot has been designed and implemented.

## 1.1 Approach and Goal

A port-based approach is applied to provide a control structure to a quadruped robot in the paper (Dresscher et al. 2014). This control structure abstracts the quadruped robot to an omni-directional moving mass as shown in figure 1.1.

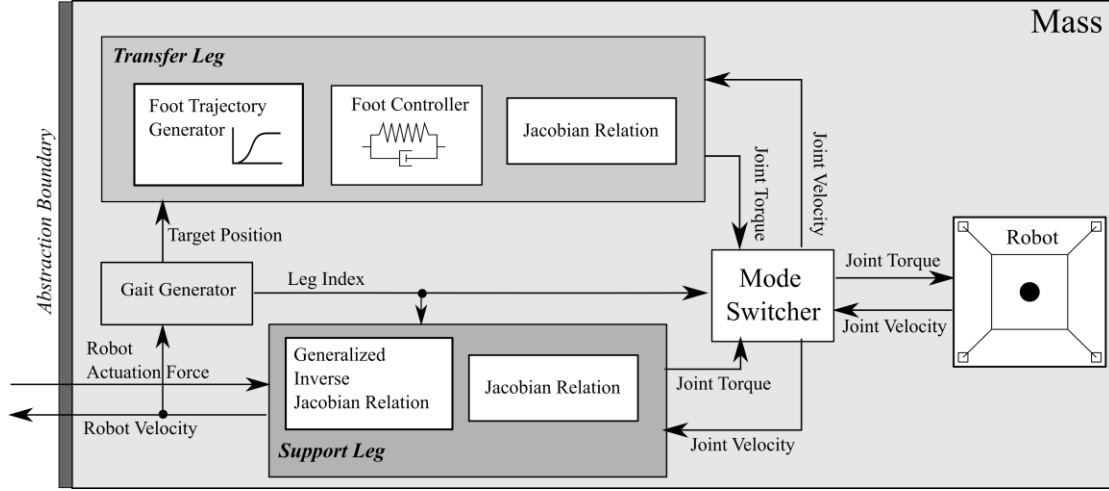


**Figure 1.1:** Abstraction of a legged robot to an omni-directional moving mass that can be controlled (Dresscher et al. 2014)

As can be seen in figure 1.1, the goal of this control structure is to control the motion of the robot by applying a force to the abstracted robot. This goal can

be achieved, for example, by using an impedance controller (Arevalo et al. 2012).

Several facilities are required in this control structure, see figure 1.2. One of the components is a gait generator. It ensures that legged locomotion remains stable throughout the walking motion. Previous work on the gait generator (van de Coelen. 2013) has been used in this research.



**Figure 1.2:** The abstraction layer with its components

The goal of this research is to develop the other components for this control structure and evaluate them separately. The integration of these components with a partially validated robot model has also been discussed.

## 1.2 Tasks

In order to make the control structure system, the following subsystems need to be developed and integrated:

The components for the transfer leg subsystem:

- The foot trajectory generator that can generate a step motion profile to transfer a foot.
- The foot controller that can control the foot on the feet trajectory when required.
- Jacobian relations that convert the force in legs to the joints torques and convert joint velocities to foot velocity.

The component for the support legs subsystem:

- Generalised inverse Jacobian which maps a force applied on the robot body to torques that are required in the joints.

The mode switcher:

- The mode switcher makes sure the robot leg can switch from a transfer leg and a support leg based on leg index signal that is calculated from the gait generator.

### **1.3 Outline**

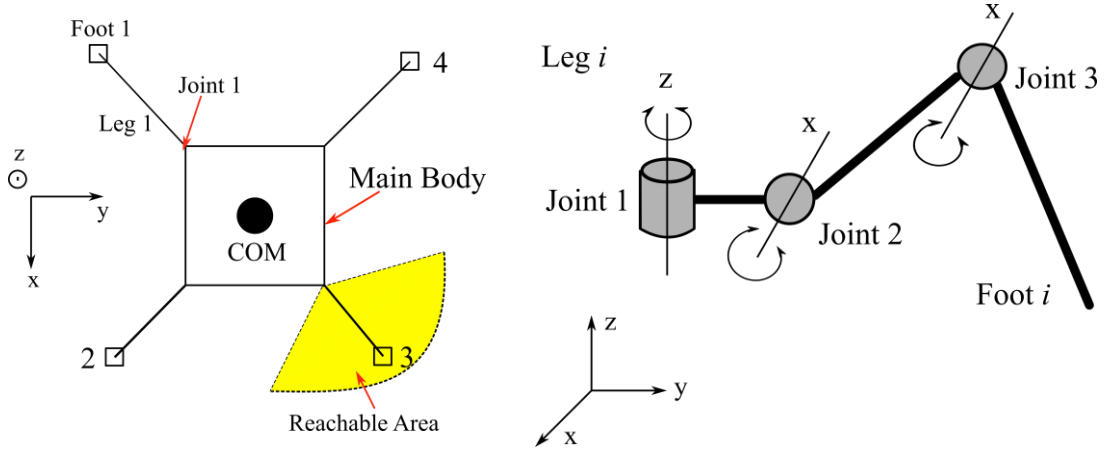
In Chapter2, the basic robot model and existing gait generator is briefly introduced as well as other background knowledge. In chapter 3, the design and evaluation of each components mentioned in section 1.2 is presented. The implementation and evaluation of the controlled quadruped robot system is provided in Chapter 4. The conclusion of this thesis and recommendations for the future research are stated in chapter 5.

## Chapter 2 Background

The background information on the basic robot model, the gait generator and the general theory is provided in this chapter.

### 2.1 Robot Model

The design of the controller is based on a quadruped robot model (Dresscher et al. 2014). Figure 2.1a showed the top view of this robot model.



**a.** The top view of the robot model.

Joint 1 is where leg 1 with the configuration shown in figure 2.1b is connected with the main body. COM refers to the center of mass of the robot.

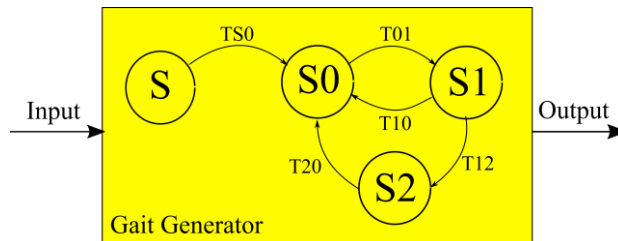
**b.** The leg configuration of the robot model. There are three degrees of freedom.  $i$  denotes the leg-number.

**Figure 2.1:** The robot model (Dresscher et al. 2014)

Each leg have three degrees of freedom as shown in figure 2.1b. The three joints of the leg will all have a limit once they are physically realized. This causes a limited reachable area for the foot as shown in Figure 2.1a. The coordinate system that is shown will be used throughout this text.

### 2.2 Gait Generator

The goal of the gait-generator is to provide timing and end-positions for the movement of the foot, which will enable the body to execute its intended movement without losing stability of the robot (van der Coelen, 2013).



**Figure 2.2:** The main gait-generator state-machine (Dresscher et al. 2014)



The gait generator is built around a state machine as shown in Figure 2.2 where, during normal operation, three states (S0, wait; S1, Calculate and; S2, Transfer) are executed sequentially based on three conditions. These conditions are: T01, The next leg can be lifted; T12, Foothold selection successful and; T20, leg transfer complete(van der Coelen, 2013).

And when the information of the location of all feet, the location and velocity of the COM of the robot and an indication that the current transfer of a leg has finished is provided, the gait generator can compute a leg index for a leg that should be moved, and it also calculates the position where the foot should be placed.

For more detailed information on the Gait Generator, please refer to (van der Coelen, 2013).

### 2.3 Homogeneous matrices

All coordinates used in the calculations are represented with 3D homogeneous matrices. The homogeneous transformation matrix (H-matrix) can be used to represent a coordinates transformation between two frames. It is a  $4 \times 4$  matrix that includes the information of the position and orientation of a rigid body. H-matrices have the following form:

$$H_i^j = \begin{bmatrix} R_i^j & \begin{matrix} P_x \\ P_y \\ P_z \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

The H-matrix  $H_i^j$  describes the orientation and position of a frame  $i$  relative to a frame  $j$ .  $R$  is a  $3 \times 3$  rotational matrix that shows the relative orientation from the frame  $i$  to the frame  $j$ , and  $P_x$ ,  $P_y$  and  $P_z$  are the coordinates of the frame  $i$  with respect to the frame  $j$  expressed in the frame  $j$ . If there is no rotational motion, then the rotational matrix  $R$  is an identity matrix  $I$ , and this H-matrix represents pure relative translation between two frames.

The inverse transformation can be expressed by the inverse of the H-matrix:

$$H_i^j = (H_j^i)^{-1} \quad (2.2)$$

The chain rule of the homogeneous matrix is:

$$H_n^0 = H_1^0 H_2^1 \dots H_n^{n-1} \quad (2.3)$$

By applying this, we can calculate the subsequent coordinate changes by multiplying the H-matrices.

## 2.4 Twists and Wrenches

In mechanics,  $T$  is called twist, and it represents the velocity of a rigid body motion geometrically (i.e., coordinate-free) and it has the 6 dimensional column vector in a form of

$$T = \begin{pmatrix} \omega \\ v \end{pmatrix} \quad (2.4)$$

where  $\omega$  represents an angular velocity vector and  $v$  a linear velocity vector.

Twists of the form

$$\hat{T} = \begin{pmatrix} \hat{\omega} \\ 0 \end{pmatrix}, \quad \text{or} \quad \hat{T} = \begin{pmatrix} 0 \\ \hat{v} \end{pmatrix} \quad (2.5)$$

are called unit twists. They only have one-degree-of-freedom either in angular velocity or linear velocity.

A one-degree-of-freedom kinematic pair or joint constraints the relative motion of two objects with a unique twist:

$$T_i^{j,j} = \hat{T}_i^{j,j} \omega_i \quad (2.6)$$

where  $\hat{T}_i^{j,j}$  is a twist and represents the motion of a frame  $i$  with respect to a frame  $j$  expressed in a frame  $j$ .

The expression of twists from a coordinate  $i$  to a different coordinate  $j$  can be computed using the adjoint mapping  $Ad_{H_i^j}$ :

$$T_{\bullet}^{j,\bullet} = Ad_{H_i^j} T_{\bullet}^{i,\bullet} \quad (2.7)$$

where

$$Ad_{H_i^j} := \begin{pmatrix} R_i^j & 0 \\ \tilde{p}_i^j R_i^j & R_i^j \end{pmatrix} \quad (2.8)$$

the adjoint of the homogeneous matrix  $H_i^j$ . In 20sim, adjoint can be calculated with the command “Adjoint(H)”, where the H is the homogeneous matrix  $H_i^j$ .

Forces applied to rigid bodies are called wrench. A wrench in vector form is a 6 dimensional row vector:

$$W = (\tau \quad F) \quad (2.9)$$

where  $\tau$  represents a torque vector and  $F$  a linear force vector.

For the same motion of a body, changing coordinates from one frame to another does not change its power. So the power should stay constant and this implies that:

$$W_i T_j^{j,i} = W_j Ad_{H_i^j} T_j^{i,i} = (Ad_{H_i^j}^T(W_j)^T)^T T_j^{i,i} \quad (2.10)$$

From equation 2.10, the expression of wrenches from a coordinate  $i$  to a coordinate  $j$  can be found:

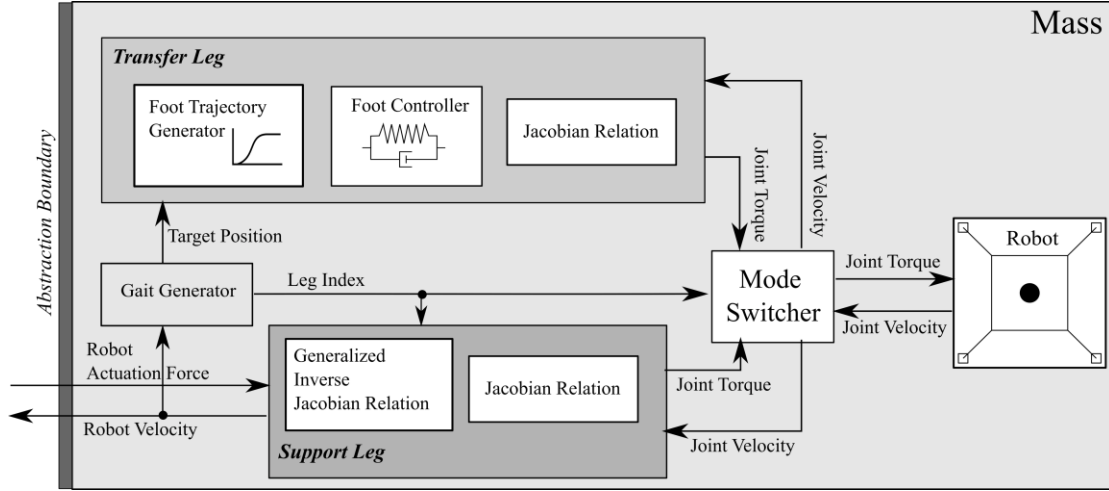
$$(W_i)^T = Ad_{H_i^j}^T (W_j)^T \quad (2.11)$$

where

$$Ad_{H_i^j}^T := \begin{pmatrix} R_j^i & -R_j^i \tilde{p}_i^j \\ 0 & R_j^i \end{pmatrix} \quad (2.12)$$

## Chapter 3 Design of components

In this chapter, the details of the abstraction control structure components are presented. As mentioned in chapter 1, we use a control structure to create a control layer such that the quadruped robot can be abstracted to an omni-directional moving mass. Several facilities are required to address the complexity of the controller. These are shown in figure 3.1.



**Figure 3.1:** The control structure with its components

The abstraction structure contains the following components:

1. **Foot trajectory generator:**  
The goal of the feet trajectory generator is to generate a step motion profile for a foot when it needs to move from its current foothold to a target foothold provided by the gait generator.
2. **Foot controller:**  
The goal of this feet controller is to make sure that the transfer foot follows the track generated by the trajectory generator.
3. **Jacobian relations:**  
The function of the Jacobian relation is that it converts the force exerts on leg to the force needed to exert in the joints. Meanwhile, it also converts the joint velocities to the foot velocity.
4. **Generalized inverse Jacobian relations:**  
The function of the generalized inverse Jacobian relations is that it converts the force exerted on the robot body to the force that needed to exert in the joints of each leg.

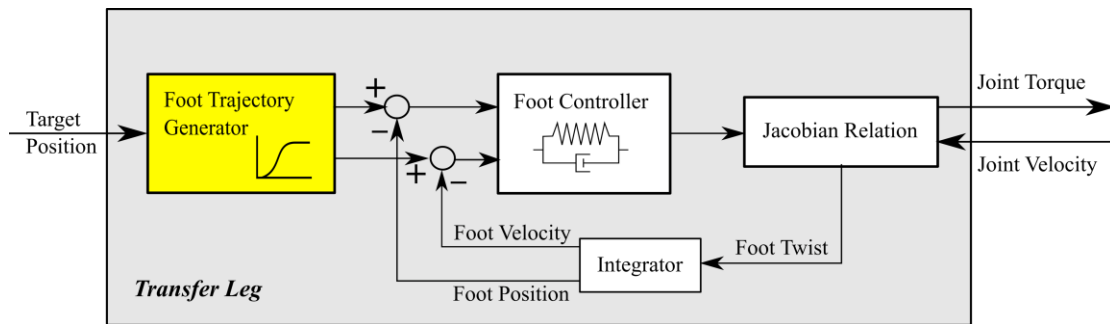
### 5. Gait Generator:

To provide timing and target positions for the movement of the foot while enabling the robot body to execute its intended movement without losing the stability.

Besides the gait generator that was designed and implemented before (van der Coelen, 2013), the other four components mentioned above are designed, implemented and evaluated separately in this chapter.

## 3.1 Foot trajectory generator

In this section, the foot trajectory generator for the transfer leg is discussed.

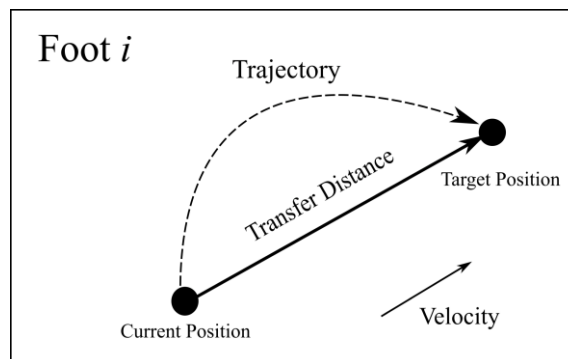


**Figure 3.2:** The subsystem of the transfer leg

After the gait generator calculated a leg index for which leg should be moved and the position where the foot should be placed, the foot trajectory generator needs to provide a trajectory profile in time. Hence the foot knows how to make a step from its current foothold to a target foothold.

### 3.1.1 Analysis

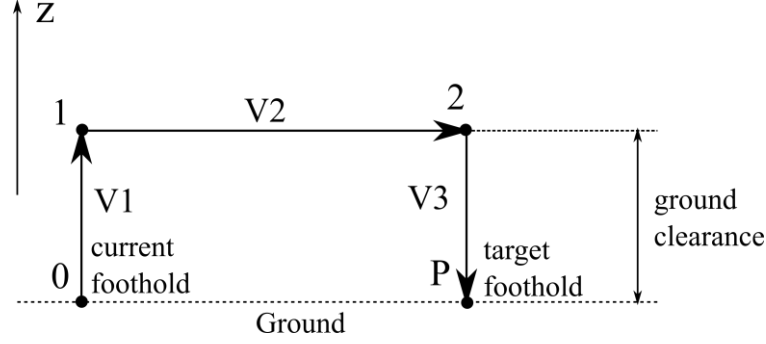
By design, the robot only moves one leg at a time, so one foot trajectory is enough for the whole system. Figure 3.3 showed the difference between foot trajectory and transfer distance.



**Figure 3.3:** Difference between foot trajectory and transfer distance

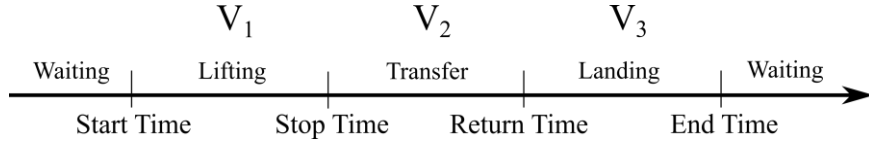
There are many possible basic trajectory shapes to move a robot leg, for example, the rectangular motion, the triangular motion and the sinusoidal

motion. In order to keep the consistency with the algorithms used in gait generator, we chose to build a trajectory with the rectangular motion.



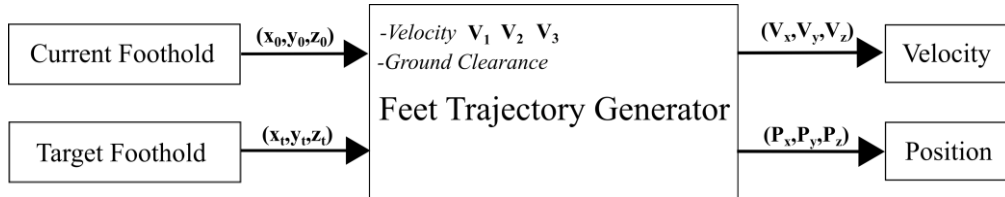
**Figure 3.4:** A step profile (side view). The transfer distance is from point 0 to point P. And the ground clearance is how high the foot lifted (van der Coelen. 2013)

The rectangular motion in figure 3.4 shows how a leg travels to its target foothold. It contains three separated movements: the foot is lifted straight off the ground with speed  $V_1$ , then moved horizontally to a point above the new foothold and then landing straight down with speed  $V_3$ . The timeline of this transfer cycle is shown in figure 3.5.



**Figure 3.5:** The timeline of the transfer cycle for a foot moves with the trajectory profile in figure 3.4

To generate the above trajectory profile, there are some relevant parameters that need to be taken into consideration: how fast the foot should move, how high the foot needs to be lifted and how big one step should be. These parameters play important roles of determining generated trajectory. Figure 3.6 shows the overview of the feet trajectory generator.



**Figure 3.6:** An overview of the foot trajectory generator

In Table 3.1, 3.2 and 3.3, the inputs, outputs and parameters of the feet trajectory generator are listed.

Name	Type	Description
Current foothold	Vector(3x1)	Coordinates of where a foot is currently located
Target foothold	Vector(3x1)	Coordinates of the next foot location

**Table 3.1:** The inputs of the foot trajectory generator

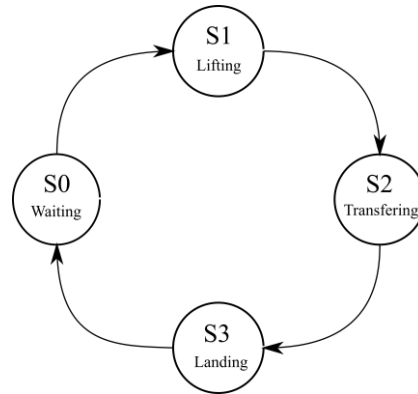
Name	Type	Description
V1	Scalar	Desired speed of the foot when a leg is lifting
V2	Scalar	Desired speed of the foot between lifting and landing
V3	Scalar	Desired speed of the foot when a leg is landing
Ground_clearance	Scalar	Distance of a foot lifted from the ground when transferring

**Table 3.2:** The parameters of the foot trajectory generator

Name	Type	Description
Velocity	Vector(3x1)	Speed of movement for the transferring foot
Position	Vector(3x1)	Position of the transferring foot

**Table 3.3:** The outputs of the foot trajectory generator

A state machine with the following states, as shown in figure 3.7, can represent the sequence of lifting, transfer and landing for each leg. Switching between states is done based on the timing.



**Figure 3.7:** The foot trajectory generator state machine

#### State 0: *Waiting*

This is the starting/ending state for the state machine. The leg that will be transferred is staying at its current position before the start time, and the leg that just finished the transfer cycle is also in this state waiting for the next movement.

#### State 1: *Lifting*

In this state, which is between “start time” and “stop time”, the foot moves only in the Z-axis, with speed V1 to the height of ground clearance. There is no movement in X-Y plane. So, the foot just lifts up straightly.

#### State 2: *Transfer*

In this state, which is between the “stop time” and before the “return time”, the foot moves horizontally in X-Y plane with speed V2 to a position above the target point. There is no movement in the Z-axis, the foot keeps a constant height.

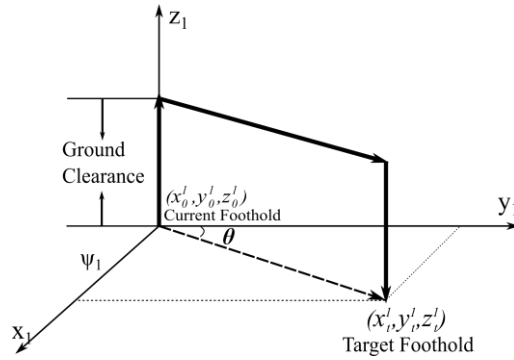
### State 3: *Landing*

During the landing state, which is between “return time” and “end time”, the foot again moves only at Z direction, going straight down with speed  $V_3$  on the ground where the target foothold is. There is no movement in X-Y plane.

#### 3.1.2 Design

The algorithm used to obtain the transfer profile mentioned in figure 3.4 is discussed in this section.

As can be seen in figure 3.4, the trajectory profile consists three motions. Two motions only exist in Z direction, corresponding to the lifting and landing movements. The other motion is in the X-Y plane, responsible with transferring.



**Figure 3.8:** A single step profile in the frame  $\psi$

Assume the starting foothold is at the origin  $(0, 0, 0)$  in frame 1, and the target position is  $(x_t^1, y_t^1, z_t^1)$ , as shown in figure 3.8. Now, we will discuss the algorithm that generates the motion profile. The states as discussed in figure 3.7 are treated separately.

### State 0: *Waiting*

Before the start time, the foot keeps standing still. The velocity is 0.

$$\begin{pmatrix} \dot{x}_t^1 \\ \dot{y}_t^1 \\ \dot{z}_t^1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (3.1)$$

### State 1: *Lifting*

After start time, the foot is lifting with the speed of  $V_1$  in Z direction. The foot finishes its lifting at the stop time. The velocity of the foot is described in the following equation:

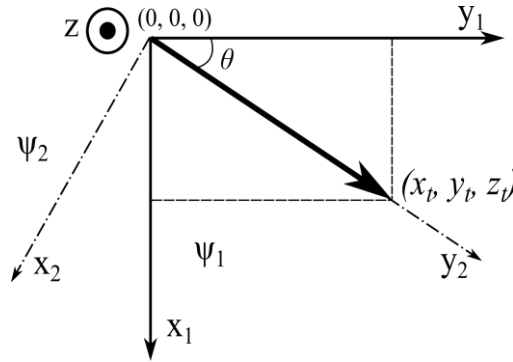
$$\begin{pmatrix} \dot{x}_t^1 \\ \dot{y}_t^1 \\ \dot{z}_t^1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ V_1 \end{pmatrix} \quad (3.2)$$



### State 2: *Transfer*

After the stop time, the foot starts to move in X-Y plane. And the transfer stops at the “return time”.

When projecting the trajectory on X-Y plane, the angle between the projection and the Y-axis is  $\theta$ . A new coordinate system, frame 2, can be introduced to simplify the trajectory calculation. See figure 3.9, the frame 2 has the same origin but a rotation around the Z axis relative to frame 1 such that the desired path was on the y axis of frame 2.



**Figure 3.9:** The trajectory profile projection on X-Y plane. Change of coordinates by rotating frame 1 to frame 2 with an angle  $\theta$

Hence, the target position can be described in frame 2 with the Z-axis in common but rotated over an angle  $\theta$ . The coordinate that expressed in frame 1 can be also expressed in frame 2 by multiply the rotational matrix:

$$\begin{pmatrix} x_t^2 \\ y_t^2 \\ z_t^2 \end{pmatrix} = R_1^2 \begin{pmatrix} x_t^1 \\ y_t^1 \\ z_t^1 \end{pmatrix} \quad (3.3)$$

with

$$R_1^2 = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3.4)$$

The angle  $\theta$  is defined by the angle between transfer trajectory projection in X-Y plane and the Y-axis of the original frame.

One thing that needs to be paid attention to is that the speed V2 is the general speed in X-Y plane, which means V2 is the speed at Y-axis of frame 2. As a result, a change of coordinates needs to be applied to represent the speed in frame 1. The velocity in this state is:

$$\begin{pmatrix} \dot{x}_t^1 \\ \dot{y}_t^1 \\ \dot{z}_t^1 \end{pmatrix} = R_2^1 \begin{pmatrix} 0 \\ V_2 \\ 0 \end{pmatrix} \quad (3.5)$$

where  $R_2^1 = (R_1^2)^{-1} = (R_1^2)^T$ , see equation 2.3.

### State 3: *Landing*

Landing is the final step of this process, and the foot starts landing at the “return time” and reaches the target position at ground at the “end time”. The velocity during this time is:

$$\begin{pmatrix} \dot{x}_t^1 \\ \dot{y}_t^1 \\ \dot{z}_t^1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -V_3 \end{pmatrix} \quad (3.6)$$

The trajectory is obtained by integrating the speed in different states.

### 3.1.3 Simulation and evaluation

In this section, the unit test of the feet trajectory generator is discussed.

#### 3.1.3.1 Simulation parameters

To check the foot trajectory generator, some step motion requirements are provided as the input of the foot trajectory generator: the speed of the foot (relatively to a world fixed frame), ground clearance, current foothold and target foothold. Here is the test case:

Speed	V1	2m/s
	V2	3m/s
	V3	1m/s
Ground clearance	0.2m	
Current foothold	(0.2, 0.3, 0)	
Target foothold	(0.5, 0.7, 0)	
Starting time	0.3s	

**Table 3.4:** The inputs and parameters used in simulation of the unit test of the foot trajectory generator

#### 3.1.3.2 Simulation results and evaluation

By applying the parameters in Table 3.4, the following results are expected based on the analysis and algorithm in the above sections.

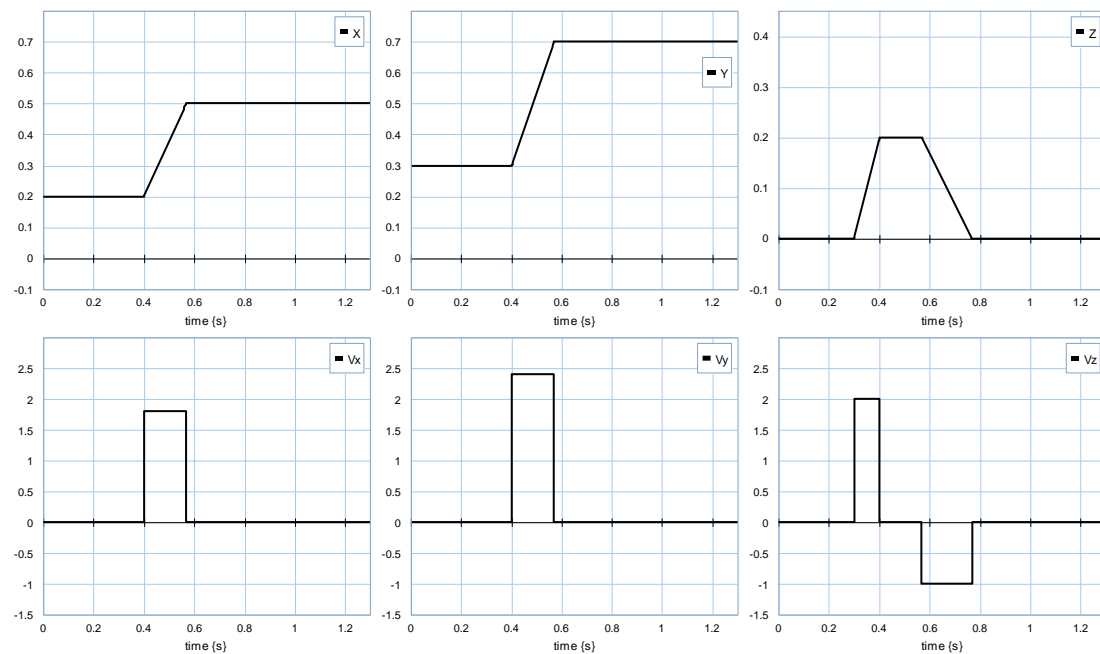
The foot should first move only in Z direction, right after that motion, it should be transferred in X-Y plane and there's no movement in Z axis. When the transfer in X-Y plane is finished, the landing of the foot is occurred in Z direction. The current as well as the target position indicate that in Z direction

the movement of this leg is trapezoidal while both in X and Y direction is a ramp.

	Expected Values
Lifting time	0.1s
Transfer time	0.17s
Landing time	0.2s
$V_x$	1.8m/s
$V_y$	2.4m/s
$V_z$	2 m/s
	1m/s

**Table 3.5:** The expected values based on the parameters in table 3.4 in the unit test of the foot trajectory generator

Figure 3.10 shows the results.



**Figure 3.10:** The simulation results of the unit test of the foot trajectory generator

The following can be observed:

1. The trajectory starts with a motion from 0 up to 0.2 meter in the Z-axis which indicates the lifting. Then the movement in Z direction stopped. Next, the leg is moving in both the X-axis and the Y-axis with initial position (0.2, 0.3) in X-Y plane. After completing the transfer in X-Y plane, the target position (0.5, 0.7) in the X-Y plane is reached. Finally the landing of the leg starts, the coordinates of the desired position in the Z direction is 0.
2. The whole transfer process starts at 0.3 second and stops at 0.77 second, it takes about 0.47 seconds. This fits the calculation from the theory in table 3.3.

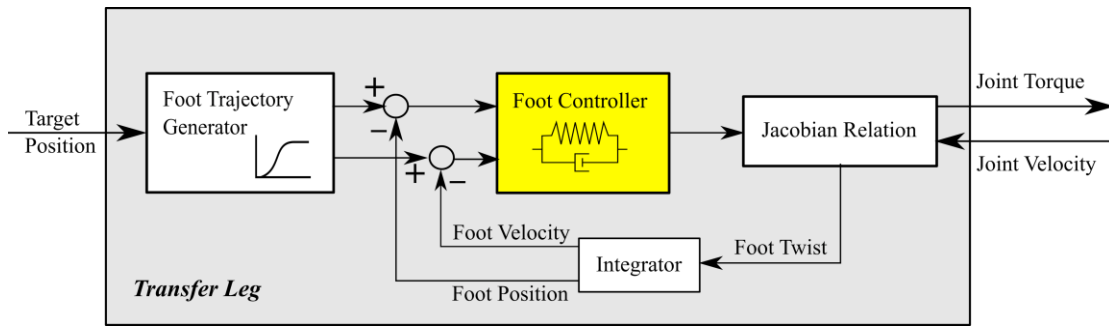
3. The profile shape for Z direction in time is trapezoidal and in X and Y direction is a ramp as expected.

4. The profile matches the design: there's no movement at X-Y plane when the foot is lifting or landing; and while transferring at X-Y plane, the foot is maintaining its height in Z direction.

Different values of speed ground clearance, current and target foothold in table 3.4 have been tried to simulate this model. They all gave the expected trajectory accordingly. Hence, this result verified that the feet trajectory generator is working properly.

### 3.2 Feet controllers

The design and evaluation of the foot controller is discussed in this section.

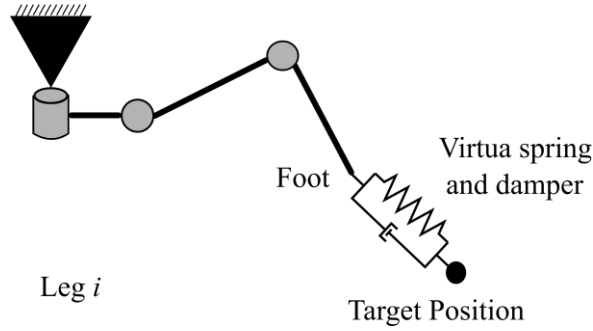


**Figure 3.11:** The subsystem of the transfer leg

The feet trajectory generator provides the profile of how is the leg transferring from its current foothold to a target foothold. And in order to make sure the leg of this robot can follow the trajectory profile in a stable and accurate way, a controller is needed.

#### 3.2.1 Design

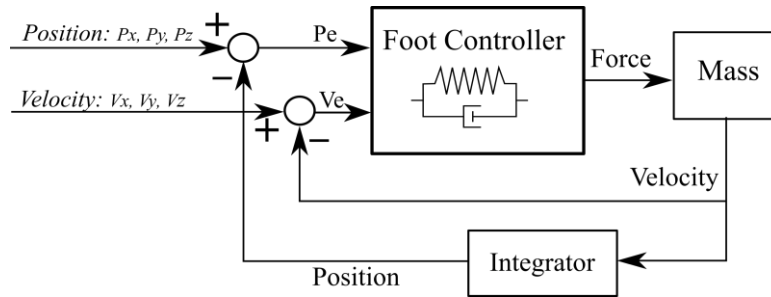
The main purpose of this controller is to make sure the motion of the leg follows the trajectory. By using the control scheme showed in figure 3.11, as the Jacobian relations provides the information about the foot velocity and the foot position, the transfer leg can be treated as if it is virtually be pushed or pulled at the foot. As a result, the leg motion can be modeled as being controlled by an impedance controller at its foot: a spring and a damper connected from the target position to the foot. See figure 3.12. And this spring and damper subsystem is the controller, which regarded as a PD controller.



**Figure 3.12:** The control system for the transfer leg. The foot is modeled being pulled to the target position by a virtual spring and a virtual damper

In this case, a PD controller is better than a PID controller. The I element is left out because it has the effect that make the error converge to zero. However, in reality, the ground, which is in the Z direction, is not always smooth. For example, if the leg is landing on a spot which is slightly higher than its former foothold, the I element in the controller would try to make the foot step deeper on the ground to minimize the error in the system. As it is in z direction, it will result in generating an unnecessary force between foot and the ground. This can be very risky to cause the instability in the system and might even damage the robot if the force is too large.

In this section, as we only focus on the controller itself, how the leg moves during the process is out of the scope. Hence, a 3D mass is used as the object of the controller. A 3D mass with PD controller system is represented in figure 3.13.



**Figure 3.13:** An overview of the foot controller of a 3D mass

For this impedance foot controller,  $R$  is the proportionality constant of the damper, and  $K$  is the spring constant. The errors in this feedback system are  $v_e$  and  $P_e$ . The output force of the controller is  $F$ . Hence, we have:

$$F = Rv_e + KP_e \quad (3.6)$$

where  $P_e = \int v_e$ .

Moreover, this 3D mass can move spatially, so equation 3.6 can be written as:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} R_x & 0 & 0 \\ 0 & R_y & 0 \\ 0 & 0 & R_z \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} + \begin{bmatrix} K_x & 0 & 0 \\ 0 & K_y & 0 \\ 0 & 0 & K_z \end{bmatrix} \begin{bmatrix} \int v_x \\ \int v_y \\ \int v_z \end{bmatrix} \quad (3.7)$$

It can be seen that the controller for this mass is actually consistent by a “spatial” spring and a “spatial” damper. And the controlled signals are the position error and velocity error in the system.

### 3.2.2 Simulation and evaluation

The unit test of the feet controller is discussed in this section.

#### 3.2.2.1 Simulation parameters

To check the performance of this controller, some parameters are needed. A 3D point mass of 0.98kg has been used in this simulation. And the reference signal is the output of the trajectory generator built in the last section.

Speed	V1	1m/s
	V2	1.5m/s
	V3	0.8m/s
Ground clearance	0.5m	
Current foothold	(0.1,0.5,0)	
Target foothold	(1.1,2,0)	
Start time	0.1s	

**Table 3.6:** The parameters of the trajectory

By trial and error, the parameters that were selected for this qualitative evaluation of the controller are listed in table 3.7. When applying this controller with the robot, the parameters in the controller need to be modified accordingly.

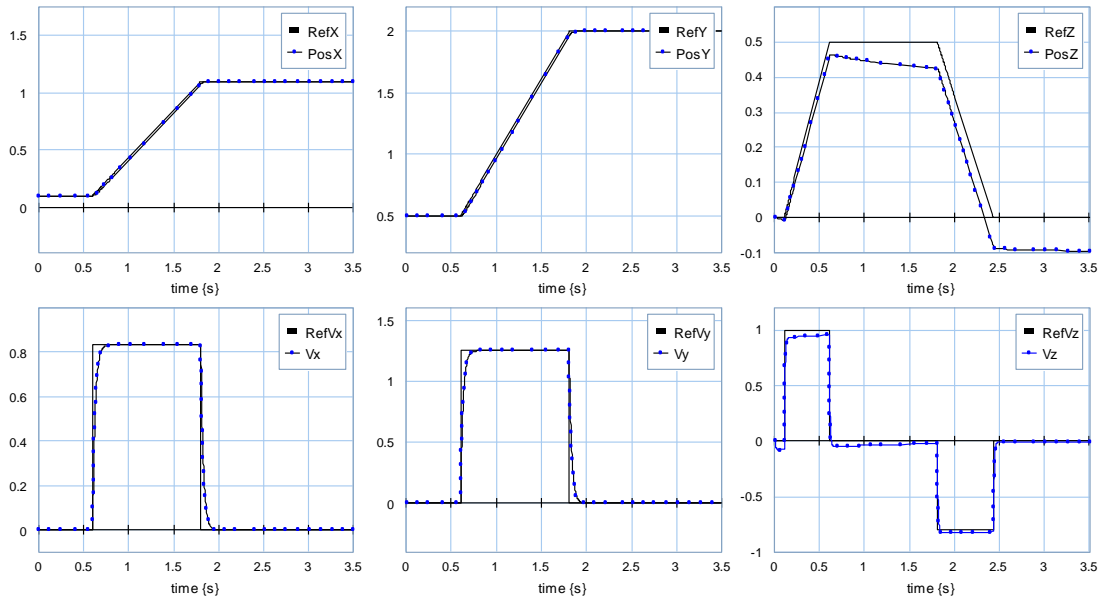
R (Proportional Gain)	$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 400 \end{bmatrix}$
K (Derivative Gain)	$\begin{bmatrix} 150 & 0 & 0 \\ 0 & 150 & 0 \\ 0 & 0 & 500 \end{bmatrix}$

**Table 3.7:** Parameters of the controller

By using the controller, the actual position and velocity should follow the reference signal.

#### 3.2.2.2 Simulation results and evaluation

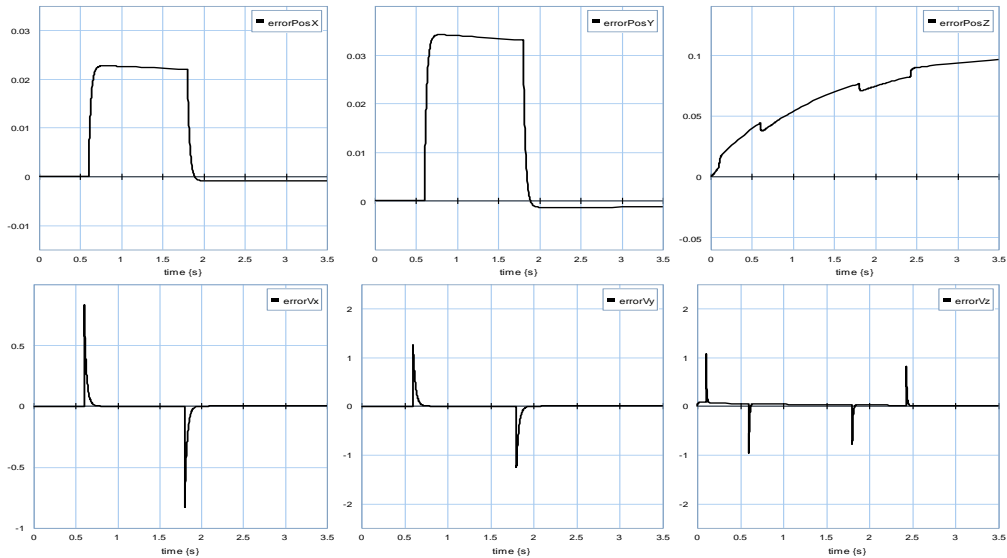
The simulation results are showed in the figure 3.14.



**Figure 3.14:** Simulation results of the foot controller

The figures in top left and top middle show that in both X and Y directions, the outputs are able to follow the reference quite well. And in the Z direction (the top right figure), although the gravity affects the lifting motion, the output trajectory still follows the reference signal closely.

Figure 3.15 shows the errors in velocity and position respectively.



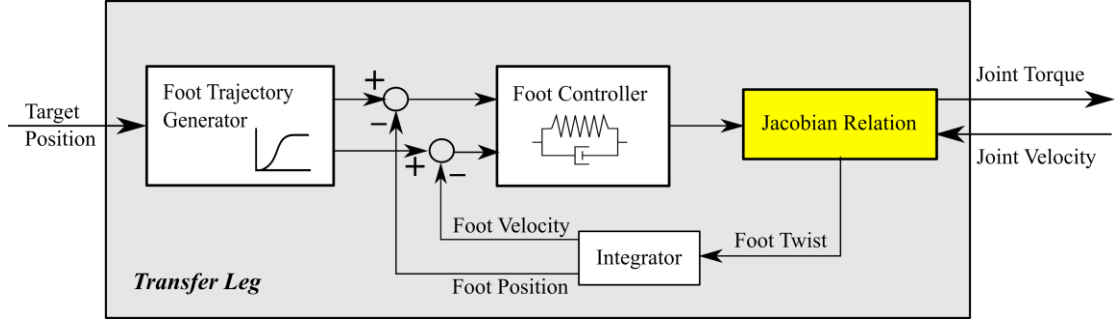
**Figure 3.15:** Simulation results of position error and velocity error

In figure 3.15, the errors of position and velocity also show that the foot follows approximately. In Z direction, there is a larger error due to gravity disturbance.

The parameters of different velocities and positions with combination of different masses have been tried in the simulation of this model. They all showed that the mass could follow the trajectory. Hence we can draw the conclusion that this controller is adequate.

### 3.3 Jacobian relation

In this section, construction of the Jacobian matrix is discussed.



**Figure 3.16:** The subsystem of the transfer leg

The goal of the jacobian relation is to convert the wrench in a leg to its joints torques and convert joint velocities to foot velocity.

For a serial robot structure like the legs of the robot in this project, the relationship between motor angles and the foot of the leg is non-linear. However, when the joint velocities are given, by using Jacobian relations, it is easy to calculate the relative position between the hip, which is the top of this leg, and the foot through its twist.

The relationship between joint velocities  $\omega = \dot{\theta}$ , to the twist of the foot, can be described as a Jacobian matrix  $J(\theta(t))$ .

$$T_i^0 = J \begin{pmatrix} \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_l \end{pmatrix} \quad (3.8)$$

$T_i^0$  expresses the foot twist in the frame 0, which is the frame where the hip is. The foot position can be derived by integrating its twist.

The relationship between the wrench applied to the foot  $W_i$  and joint torques  $\tau$  in this leg with the same motion can expressed by the same Jacobian relations:

$$\tau^T = J^T W_i^T \quad (3.9)$$

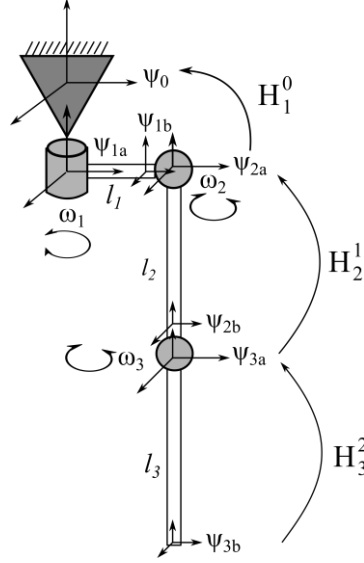
Hence, using Jacobian relations we can meet our goal.



### 3.3.1 Design

In order to calculate the general Jacobian relations for the legs of this robot, a serial link with the configuration of the robot leg is used. See figure 3.17.

From the figure, we can see that this serial link has three links and three joints. The top of which is assumed to be fixed. The first joint can only rotate around Z-axis with angular velocity of  $\dot{\theta}_1$ . The second and the third joint of this serial link can only rotate around X-axis, the angular velocity is  $\dot{\theta}_2$  and  $\dot{\theta}_3$  respectively. The length of first link is  $l_1$ . And the length of the second link is  $l_2$  and the third one is  $l_3$ .



**Figure 3.17:** The configuration and the coordinate system of the robot leg

With the configuration in figure 3.17, equation 3.8 can be written as:

$$T_3^{0,0} = J(\theta) \dot{\theta} \quad (3.10)$$

where  $\dot{\theta} = (\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3)^T$ . And  $J(q)$  is the Jacobian relation:

$$J(\theta) = [\hat{T}_1^{0,0} \quad Ad_{H_1^0} \hat{T}_2^{1,1} \quad Ad_{H_2^0} \hat{T}_3^{2,2}] \quad (3.11)$$

There is only rotational motion around the Z-axis for joint 1. For joint 2 and joint 3, there is also only rotational motion around X-axis. Hence the unit twist of this serial link for each joint is:

$$\hat{T}_1^{0,0} = (0, 0, 1, 0, 0, 0)^T \quad (3.12)$$

$$\hat{T}_2^{1,1} = \hat{T}_3^{2,2} = (1, 0, 0, 0, 0, 0)^T \quad (3.13)$$

Besides unit twists,  $Ad_{H_1^0}$  and  $Ad_{H_2^0}$  are needed to calculate the Jacobian relations in equation 3.11. As stated in section 2.4, in 20sim, they can be calculated when the H-matrices  $H_1^0$  and  $H_2^0$  are obtained.

The configuration of the serial link can be described by homogeneous transformation matrix. For link one, it is a rigid body with length  $l_1$  which rotates in Z-axis around the reference frame, there are two H-matrices to represent that. According to the chain rule, we have  $H_1^0$ :

$$H_1^0 = H_{1a}^0 H_{1b}^{1a} \quad (3.14)$$

It transfers the coordinate of the tip position of link 1 in the reference frame 0.

$H_{1a}^0$  describes the rotational motion of joint one between frame 1a and frame 0. Frame 0 is the reference frame and frame 1a is a body fixed frame fixed at the top of link one. The position of the origin for frame 0 and frame 1a are the same, so the relative motion between these two frames can represent by the following H-matrix:

$$H_{1a}^0 = \begin{bmatrix} & & 0 \\ & R_{1a}^0 & 0 \\ 0 & 0 & 0 \\ & & 1 \end{bmatrix} \quad (3.15)$$

$H_{1b}^{1a}$  describes the relation between two ends of link one, frame 1a and frame 1b:

$$H_{1b}^{1a} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.16)$$

These equations also hold for link 2 and link 3. For the second link, applying the chain rule, the relative motion between link one and link two is described in  $H_2^1$ , and the relative motion between link two and the reference frame is  $H_2^0$ :

$$H_2^1 = H_{2a}^{1b} H_{2b}^{2a} \quad (3.17)$$

$$H_2^0 = H_1^0 H_2^1 \quad (3.18)$$

In the second link, the body-fixed frame 2a and frame 2b are introduced respectively. Frame 2a is fixed with the joint which is on the top of link 2, and shares the same origins position as frame 1b, frame 2b is attached to the bottom of link 2. As this link is rotating around X-axis, then the H-matrix for frame 1b and frame 2a is:

$$H_{2a}^{1b} = \begin{bmatrix} & R_{2a}^{1b} & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.19)$$

The relation between bottom frame 2b and the joint frame 2a is:

$$H_{2b}^{2a} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.20)$$

And now, all components in equation 3.11 are obtained. Hence we can calculate the Jacobian matrix to convert the leg wrench to its joint torques.

However, in order to calculate the foot velocity and position, to convert the joint velocities to foot twist is necessary. As a result, the relative motion between link three and the reference frame is needed:

$$H_3^2 = H_{3a}^{2b} H_{3b}^{3a} \quad (3.21)$$

$$H_3^1 = H_2^1 H_3^2 \quad (3.22)$$

$$H_3^0 = H_1^0 H_2^1 H_3^2 \quad (3.23)$$

With the H-matrices:

$$H_{3b}^{2a} = \begin{bmatrix} & R_{3b}^{2a} & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.24)$$

$$H_{3b}^{3a} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.25)$$

### 3.3.2 Simulation and evaluation

The unit test of the Jacobian relation will be discussed in this section. The key point of the testing is to check if the mapping is correct.

#### 3.3.3.1 Simulation parameters

For the simulation, the following values were used for the length of each link:

Link 1	$l_1=0.1\text{m}$
Link 2	$l_2=0.3\text{m}$
Link 3	$l_3=0.4\text{m}$

**Table 3.8:** The length of the each link in the serial links

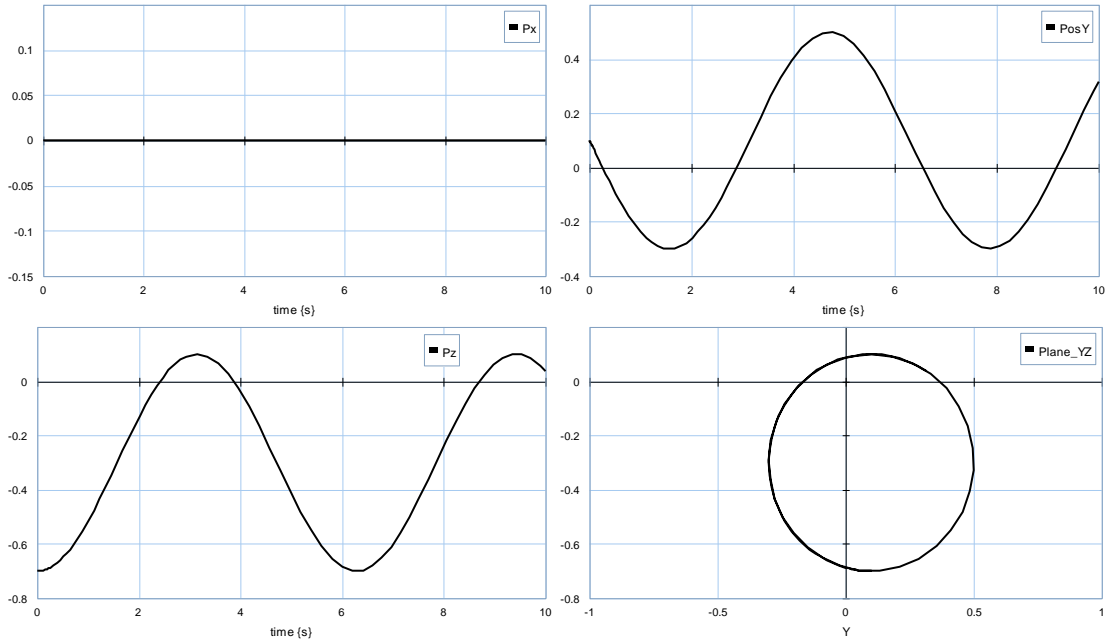
The initial configuration of the leg is that the hip, which is the top of the leg, is fixed at the origin. Link 2 and link 3 is “standing” straight next to the Z-axis, see figure 3.17. With these settings, the top of the leg is joint 1, and the coordinate is (0, 0, 0), and the coordinate of joint 2 is (0, 0.1, 0), and for joint 3, it is (0, 0.1, -0.3), the tip of this leg is at the bottom (0, 0.1, -0.7).

### 3.3.3.2 Simulation results and evaluation

The simulations contains three tests:

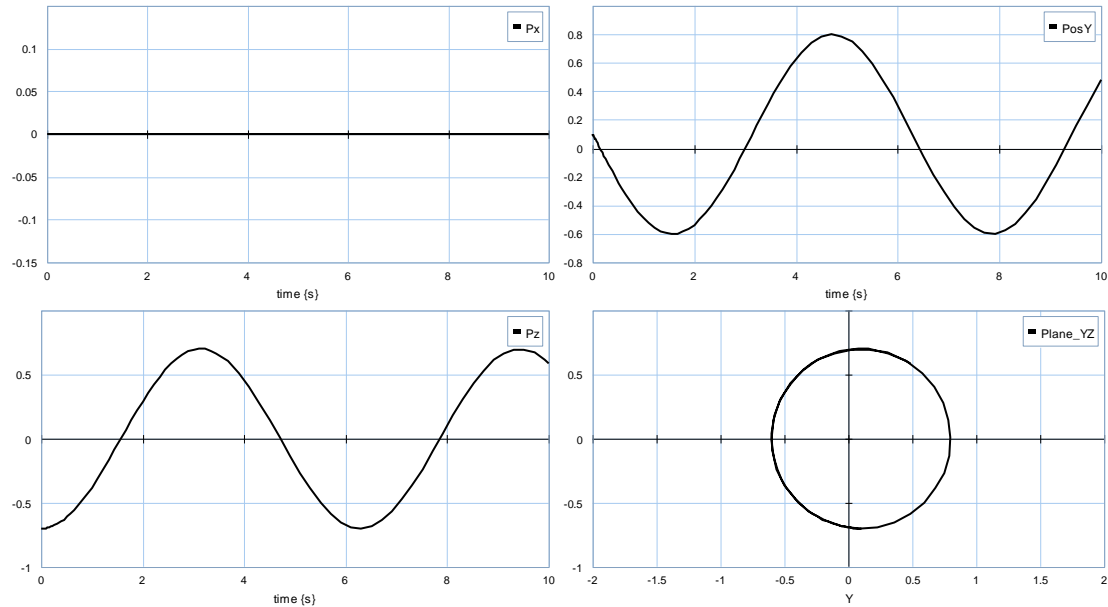
1. Only joint 3 is moving while joint 1 and joint 2 are steady:  
Since only link 3 is rotation around X-axis, the trajectory of the tip of the leg should be a circle with a center at where joint 3 is. The radius is the length of this part:  $l_3$ .
2. Only joint 2 is moving while joint 1 and joint 3 are steady:  
In this case, link 2 and link 3 can be regarded as a rigid body. Hence, the track of the tip should be a circle around X-axis with the center at where the joint 2 is. The radius is the sum of  $l_2$  and  $l_3$ .
3. Only joint 1 is moving while joint 2 and joint 3 are steady:  
This situation makes the entire leg behaved a rigid body with a shape shown in figure 3.17, and it should rotate around Z-axis. As a result, the tip will draw a circle in X-Y plane with the origin as its center and  $l_1$  as its radius.

The following figures show the simulation results of the three situations mentioned above.



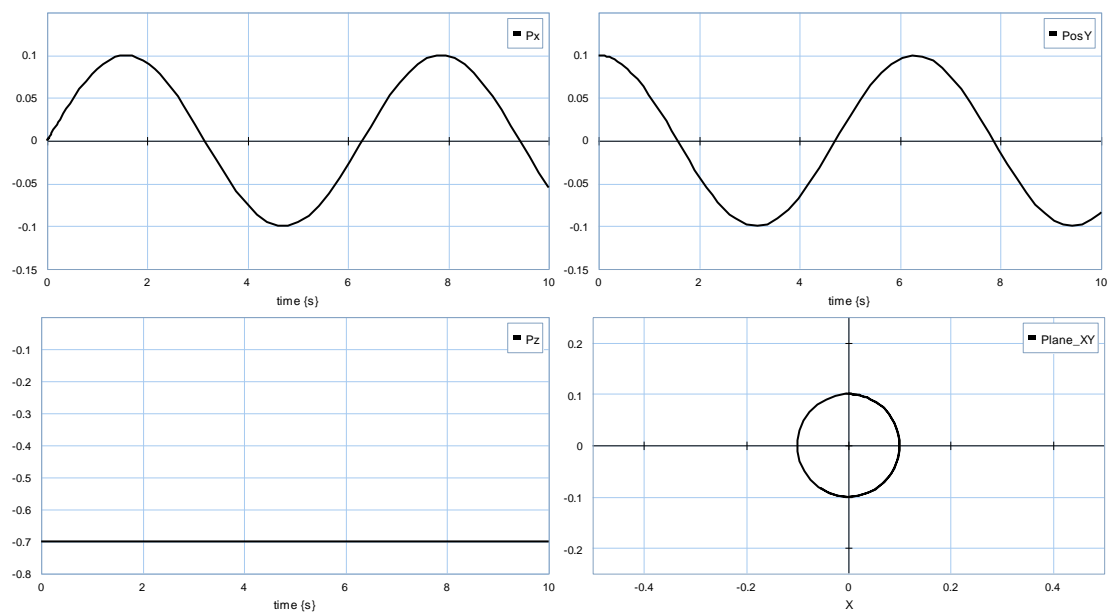
**Figure 3.18:** Simulation results of serial links when  $\omega_1 = \omega_2 = 0$ ,  $\omega_3 = 1$  rad/s

In figure 3.18, the joints have  $\omega_1 = \omega_2 = 0$ , and  $\omega_3 = 1$  rad/s as input. As expected, the tip trajectory is a circle at Y-Z plane since the joint 3 is defined rotates around X direction. The center of the circle is (0, 0.1, -0.3) which is the location of joint 3. And the radius is 0.4m.



**Figure 3.19:** Simulation results of serial links when  $\omega_1 = \omega_3 = 0$ ,  $\omega_2 = 1$  rad/s

Figure 3.19 shows the situation when  $\omega_1 = \omega_3 = 0$ , and  $\omega_2 = 1$  rad/s. As joint 3 has no input, so there is no relative movement between link 2 and link 3. This means link 2 and link 3 is fixed together and can be regarded as one rigid body, and the length of this rigid body is 0.8m. The center of the circle is (0, 0.1, 0.8), that is where the joint 2 is located. Joint 2 rotates around X-axis, hence, the tip trajectory is a circle in Y-Z plane, and the radius is 0.7m.



**Figure 3.20:** Simulation results of serial links when  $\omega_2 = \omega_3 = 0$ ,  $\omega_1 = 1$  rad/s

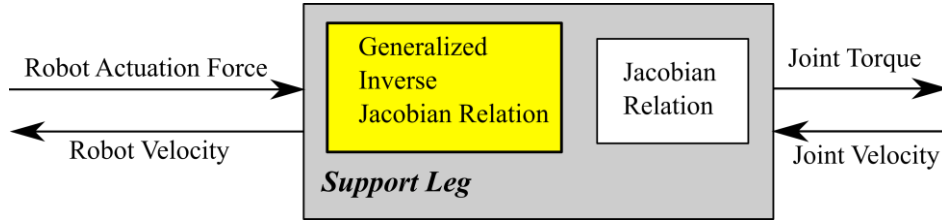
In figure 3.20, the joints have inputs with  $\omega_2 = \omega_3 = 0$ , and  $\omega_1 = 1$  rad/s. Since there is no input for joint 2 and joint 3. This means no relative motion between link 1 and link 2 nor between link 2 and link 3. This indicates that link 1, link 2 and link 3 are fixed with each other. The entire leg behaved like a rigid body. In the simulation result, it shows that the tip draws a circle on X-Y plane, the radius is 0.1m.

More tests have done with different joint velocities and link length, all the simulation results matches the behavior. Hence, we can conclude that according to the above simulation results, the Jacobian relations is giving the correct mapping with given joint velocities to foot position.

By applying the same Jacobian relations in equation 3.9, the relationship between the leg wrench and joint torques can be derived.

### 3.4 Generalized inverse Jacobian relation

In this section, the Generalized Inverse Jacobian relation is discussed.

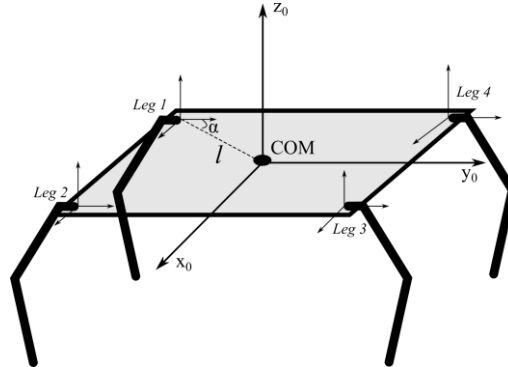


**Figure 3.21:** The subsystem of the support legs

The goal of the generalized inverse Jacobian relations is mapping wrench that applied on the robot body to the torques that are required in each leg's joints. For example, with a give wrench on the robot body, the torque that each joint of the support legs need to deliver should be calculated such that the body is supported or can be moved. By using the Generalized inverse of Jacobian matrix, this goal can be reached.

#### 3.4.1 Design

Figure 3.22 shows the coordinate system of the robot body and the connection points of its legs.



**Figure 3.22:** Coordinate system of the robot body and the legs

To achieve the goal and to calculate the generalized inverse of the Jacobian relations, the design is divided into three steps. The first step is to calculate how much wrench should each leg delivers when this it is applied to the robot body. Step two is to convert the wrench to be applied by the single leg to its joint torques. The final step is to combine the first two steps, such that the relation between the wrench applied on the robot body to torques that needed in all joints is obtained.

Step 1: Allocate the wrench on robot body to each leg

The transformation of a wrench between two frames is shown in equation 2.10. Hence, the wrench of each leg expressed at COM is:

$$(W_0)^T = Ad_{H_0^i}^T (W_{Li})^T \quad (3.26)$$

According to the coordinate systems in figure 3.22, the H-matrix that describes the position of these four legs with respect to the center of the robot body are:

$$H_1^0 = \begin{bmatrix} & I & -lsin\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.27)$$

$$H_2^0 = \begin{bmatrix} & I & lcos\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.28)$$

$$H_3^0 = \begin{bmatrix} & I & lcos\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.29)$$

$$H_4^0 = \begin{bmatrix} & I & -lcos\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.30)$$

The total wrench that exerted on the robot body is equal to the weighted sum of each leg. Then the total wrench at COM is:

$$(W_0)^T = Ad_{H_0^1}^T (W_{L1})^T + Ad_{H_0^2}^T (W_{L2})^T + Ad_{H_0^3}^T (W_{L3})^T + Ad_{H_0^4}^T (W_{L4})^T \quad (3.31)$$

This can also be written as:

$$(W_0)^T = A(\alpha) \begin{pmatrix} (W_{L1})^T \\ (W_{L2})^T \\ (W_{L3})^T \\ (W_{L4})^T \end{pmatrix} \quad (3.32)$$

where  $A(\alpha)$  is the Jacobian relation between body wrench and wrench of the legs:

$$A(\alpha) = [Ad_{H_0^1}^T \quad Ad_{H_0^2}^T \quad Ad_{H_0^3}^T \quad Ad_{H_0^4}^T] \quad (3.33)$$

And if we want to calculate how the body wrench allocates to each leg, the inverse of  $A(\alpha)$  is needed.

As can be seen from the above equations, each Adjoint matrix in  $A(\alpha)$  returns a  $6 \times 6$  matrix, so  $A(\alpha)$  is a  $6 \times 24$  matrix. This Jacobian matrix has more columns than rows. In robotic, this is the reflection of a redundant situation of the robot which has more actuated degrees-of-freedom than it needs to move the body by the legs. This also means that the inverse  $A^{-1}$  of the Jacobian  $A(\alpha)$  is not unique.

One widely used method in robotics to solve this inverse problem is called pseudo-inverse (Penrose. 1955). Mathematically, the pseudo-inverse  $A^+$  is the generalized inverse Jacobian matrix of A. In this case, it has the analytical form of:

$$A^+ = A^T (AA^T)^{-1} \quad (3.34)$$

As a result, the wrenches in legs when applying a wrench on the body is:

$$\begin{pmatrix} (W_{L1})^T \\ (W_{L2})^T \\ (W_{L3})^T \\ (W_{L4})^T \end{pmatrix} = A^+ (W_0)^T \quad (3.35)$$

One thing that needs to be noticed, the pseudo-inverse is just a mathematical solution, so it has no intrinsic geometrical or physical meaning.

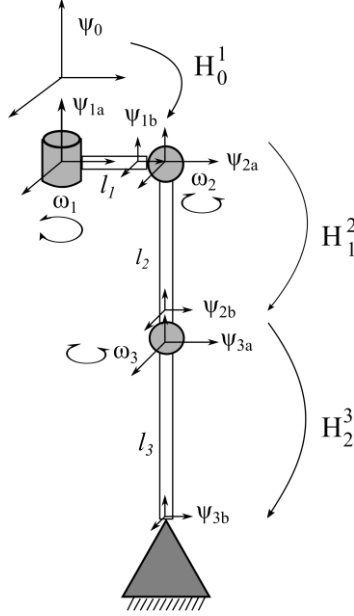
Step 2: Calculation of joint torques with given wrenches in a leg

Now, we have the relation between the leg wrench and total body wrench. But this leg wrench is just the wrench in the hip. The torques that need to be applied to the joints for the leg to exert the wrench also need to be calculated. The relation between joint torque and leg wrench for each leg is the transposed Jacobian:



$$\tau_{Li}^T = J_L^T(\theta) W_{Li}^T \quad (3.36)$$

The Jacobian is derived based on the configuration and coordinate systems showed in figure 3.23.



**Figure 3.23:** The configuration and the coordinate systems of the robot leg

In this case, the reference frame is fixed with the foot of the leg. That means the foot of the leg is standing still on the ground to maintain the stability while supporting and moving the robot body. When using the configuration and coordinate systems showed in figure 3.23, the Jacobian relation for each leg is

$$J_{Li} = [Ad_{H_{1a}^{3b}} \hat{T}_0^{1,1} \quad Ad_{H_{2a}^{3b}} \hat{T}_1^{2,2} \quad \hat{T}_2^{3,3}] \quad (3.37)$$

Where

$$\hat{T}_0^{1,1} = (0, 0, -1, 0, 0, 0)^T \quad (3.38)$$

$$\hat{T}_1^{2,2} = \hat{T}_2^{3,3} = (-1, 0, 0, 0, 0, 0)^T \quad (3.39)$$

And

$$H_{1a}^{3b} = (H_{3b}^{3a})^{-1} (H_{3a}^{2b})^{-1} (H_{2b}^{2a})^{-1} (H_{2a}^{1b})^{-1} (H_{1b}^{1a})^{-1}$$

$$H_{2a}^{3b} = (H_{3b}^{3a})^{-1} (H_{3a}^{2b})^{-1} (H_{2b}^{2a})^{-1}$$

See equation 3.16, 3.19, 3.20, 3.24 and 3.25.

Step 3: the relation between the body wrench to joint torques

With all the equations in previous steps, we can obtain the relation between joint torques of all legs and the body wrench:

$$\begin{pmatrix} \tau_{L1}^T \\ \tau_{L2}^T \\ \tau_{L3}^T \\ \tau_{L4}^T \end{pmatrix} = J_L^T A^+ W_0^T \quad (3.40)$$

where

$$J_L^T = \begin{bmatrix} J_{L1}^T & 0 & 0 & 0 \\ 0 & J_{L2}^T & 0 & 0 \\ 0 & 0 & J_{L3}^T & 0 \\ 0 & 0 & 0 & J_{L4}^T \end{bmatrix} \quad (3.41)$$

and

$$A^+ = [Ad_{H_0^1}^T \quad Ad_{H_0^2}^T \quad Ad_{H_0^3}^T \quad Ad_{H_0^4}^T]^+ \quad (3.42)$$

So the Generalized Inverse Jacobian is  $J_L^T A^+$ .

### 3.4.3 Simulation and evaluation

In this section, some 20sim models are built to test the above equations. The configuration of the robot body and its leg used in this section are showed in figure 3.22 and figure 3.23.

#### 3.3.4.1 Test methodology

Each step described above is tested separately to make sure that all the equations are correct.

For the simulation, the following values were used:

Leg Link 1	$l_1=0.1\text{m}$
Leg Link 2	$l_2=0.3\text{m}$
Leg Link 3	$l_3=0.4\text{m}$
Body Side length	$l=0.4\text{m}$

**Table 3.8:** The properties of the robot model used in this simulation

The first step is to check the transformation from the body wrench to leg wrench. Suppose all the four legs are standing on the ground to support the robot body. The hips, which are the connection points between the legs and the robot body, are fixed in the body. The distance between each hip and COM is constant and the same. This means the wrench in COM is equally distributed to all the legs and each of them delivers one quarter of the total wrench in COM. And if there are only three legs to support the body, which is normally the case as the forth leg is making a step to go to the new location, the body wrench would be allocated equally to these three legs.

Step two is designed to check if the leg wrench can be interpreted by joint torques. By applying the wrench of the leg to its hip, the joints of the leg show the corresponding force to support the leg wrench in a form of joint torque. Different leg configurations with the same wrenches show different combinations of joint torques.

After the first two steps, when all the parts are checked and functioning, the third step is to connect these two components. In this final step, with given body wrench, the joint torques of each leg will be calculated to support the body.

#### 3.4.4.2 Simulation results

During the simulation, all the joint velocities are set to 0. This means there is no relative motion between different links of the leg. It shows how the joint torque reacts with different wrench applied to robot body.

Test 1: Applying a wrench in the Z direction on the body

The robot body in this test is in a square shape with the side length of 0.4 meters, and the location of the hip, which is the location of the leg connected with the robot body, is in the corner of the robot body. See figure 3.22. The wrench applied on the body is  $W = (0, 0, 0, 0, 0, 40)^T$ .

Table 3.9 showed the simulation results.

Number of support legs		Leg Wrench
4	Leg 1	$(0, 0, 0, 0, 0, 10)^T$
	Leg 2	$(0, 0, 0, 0, 0, 10)^T$
	Leg 3	$(0, 0, 0, 0, 0, 10)^T$
	Leg 4	$(0, 0, 0, 0, 0, 10)^T$
3	Leg 1	$(0, 0, 0, 0, 0, 13.3)^T$
	Leg 2	$(0, 0, 0, 0, 0, 13.3)^T$
	Leg 3	$(0, 0, 0, 0, 0, 13.3)^T$
	Leg 4	$(0, 0, 0, 0, 0, 0)^T$

**Table 3.9:** Joint torque when applying a wrench acting like weight on robot with four legs are all supporting the robot body

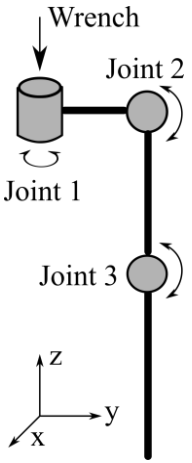
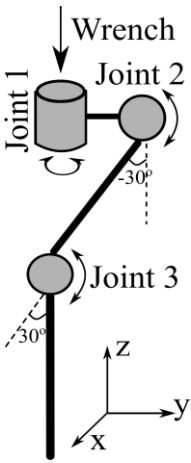
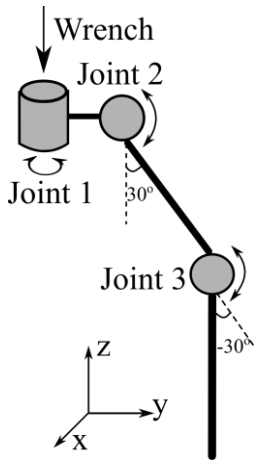
As expected in the last section, the wrench applied on the body is equally delivered on each leg. This means equation 3.35 gave the correct mapping between the body wrench and the wrenches in legs.

Test 2: Applying a wrench in the Z direction on a leg

In this part, the mapping between leg wrench and the joint torques is checked. By applying a wrench on the leg, different torques are calculated by the Jacobian matrix with different leg configurations. The wrench applied on the

leg is  $W = (0, 0, 0, 0, 0, 10)^T$ . Three leg configurations, which are A, B and C as shown in table 3.10, were used to verify the Jacobian relations.

Applying a wrench in Z direction on hip of the leg, that is also where the first joint is. As joint 1 rotates around Z-axis, it should not experience any torque. But for joint 2 and joint 3, as they rotate around X-axis, with different configuration, the torque should be different. As both joint 2 and joint 3 experience the same wrench, the torque in these two joints is defined by the lever arm length and rotation direction of this configuration.

	A	B	C
Joint Torque			
Joint 1	0	0	0
Joint 2	1	1	1
Joint 3	1	-0.5	2.5

**Table 3.10:** Joint torques when applying a linear force with different leg configuration

Table 3.10 verifies the comments above. For configuration A, the lever arm length are the same for joint 2 and joint 3, and the simulation result showed that they have the same torque. For configuration B and C, the lever arm length of joint 2 remained the same, hence the torque in joint 2 is the same. But for joint 3, these two configurations have different lever arm length, so the torque is different.

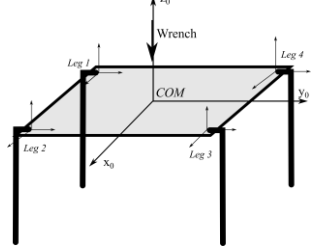
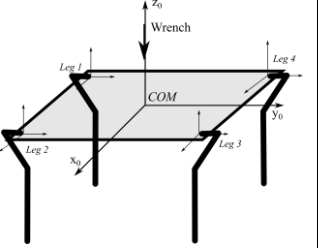
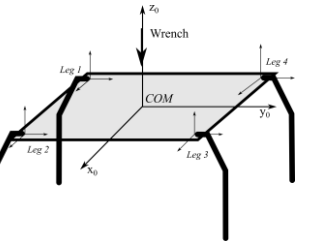
### Test 3: Applying a wrench in the Z direction on the body

In this test, the test 1 and test 2 are combined. A wrench is directly applied on the robot body, the joint torques of each leg is observed.

The first test is that all four legs are supporting the robot body in their initial configuration. And the simulation parameters are the same with the above two, the robot body in this test is square with the length of the side is 0.4 meters, and the location of the hip, which is the location of the leg connected with the robot

body, is in the corner of the robot body. And the wrench applied on the body is  $W = (0, 0, 0, 0, 0, 40)^T$ .

As can be seen in table 3.11, leg 1 and leg 2, leg 3 and leg 4 are identical, hence, the joint torques in leg 1 and leg2, leg 3 and leg 4 should be the same. And leg 1 and leg 2 are mirrored of leg 3 and leg 4. So the torque in leg 1 and leg 2 should be equal and opposite compare with the torque in leg 3 and leg 4.

	A			B			D		
									
	Joint 1	Joint 2	Joint 3	Joint 1	Joint 2	Joint 3	Joint 1	Joint 2	Joint 3
Leg 1	0	-1	-1	0	-1	0.5	0	-1	-2.5
Leg 2	0	-1	-1	0	-1	0.5	0	-1	-2.5
Leg 3	0	1	1	0	1	-0.5	0	1	2.5
Leg 4	0	1	1	0	1	-0.5	0	1	2.5

**Table 3.11:** Joint torques when applying a linear force directly on the COM of a robot body with different leg configuration

The results show in table 3.11 matched the expectation, and has consistency with those results showed in table 3.9 and table 3.10.

### 3.5 Conclusions

In this chapter, each unit component that needed in the control structure was designed and tested separately, all of them are functional. The integration of all these components is presenting in the next chapter.

## Chapter 4 Integration

In this chapter, the integration of the components that mentioned in chapter 1 is discussed.

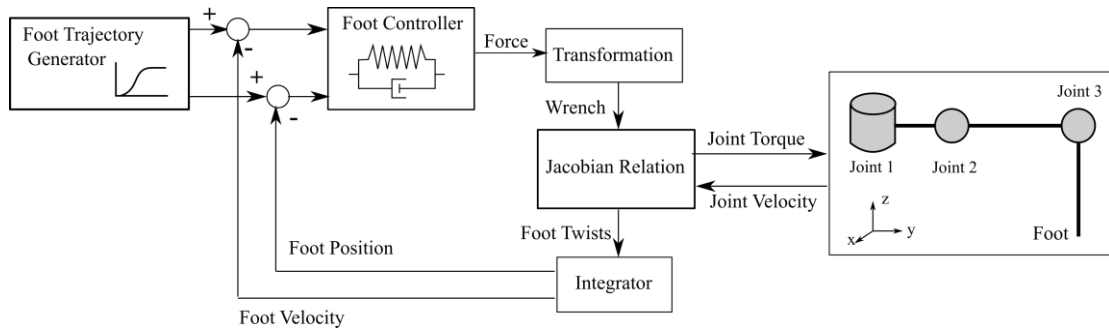
In order to walk with stability, the robot legs need to have two functions: transferring and supporting. In this project, one of the four legs is called the transfer leg. While the transfer leg is taking a step to move forward, the other three legs are cooperating to maintain the stability and motion of the robot body. These legs are called supportive legs.

As shown in figure 1.2, the foot trajectory generator, the foot controller and the Jacobian relations are the components cooperating with the transfer leg. The generalized inverse Jacobian relation is working with the supportive legs. Hence, the whole integration can be done in three steps:

- Moving the transfer leg to a target position with desired trajectory
- Keep the robot body stable with those supporting legs
- The integration of the two roles describe above.

### 4.1 Integration of the components for a transfer leg

The goal of this section is to implement the subsystem for the transfer leg. So the trajectory generator, the foot controller and the Jacobian relations are integrated on a single leg. This leg should be able to move to the target position with desired trajectory.



**Figure 4.1:** A schematic drawing of a controlled transfer leg

Figure 4.1 shows the interconnection of each component in transferring a leg to a new target position. The foot trajectory calculates the step profile based on the current position of the foot and the target position the foot needs to go with given velocities. By applying the foot controller in this feedback control system, during the transferring process the velocity and position of this foot closely follow the trajectory. To calculate the joint torque for each joint in this leg, the Jacobian relation is needed, but in its transposed matrix form.

Moreover, the output of the controller is force while the Jacobian is mapping wrench with joint torque, transformation box showed in figure 4.1 offered the converting from force to wrench by using the definition of wrench (see equation 2.9).

#### 4.1.1 Implementation

This implementation was done with an existing robot model. The purpose of this implementation is to test the subsystem of the transfer leg. In this case, only the relative velocity and position between the hip and foot of this leg is of interest. Hence the robot body is fixed.

##### 4.1.1.2 Simulation parameters

For the simulation, the values in table 4.1 were used for the length of each link and the position of the hip and the foot.

Name	Value
Link 1	$l_1=0.0\text{m}$
Link 2	$l_2=0.2\text{m}$
Link 3	$l_3=0.2\text{m}$
Hip Position (Joint 1 position)	(0.2, 0.2, 0.2)
Foot Position	(0.2, 0.4, 0 )

**Table 4.1:** The properties of this transferring leg

Based on the leg length and the foot position, the trajectory generator calculates the step profile with the parameters shown in table 4.2.

Speed	V1	0.1m/s
	V2	0.2m/s
	V3	0.05m/s
Ground clearance	0.1m	
Current foothold	(0.2, 0.4, 0)	
Target foothold	(0.287, 0.3, 0)	
Start time	0.1s	

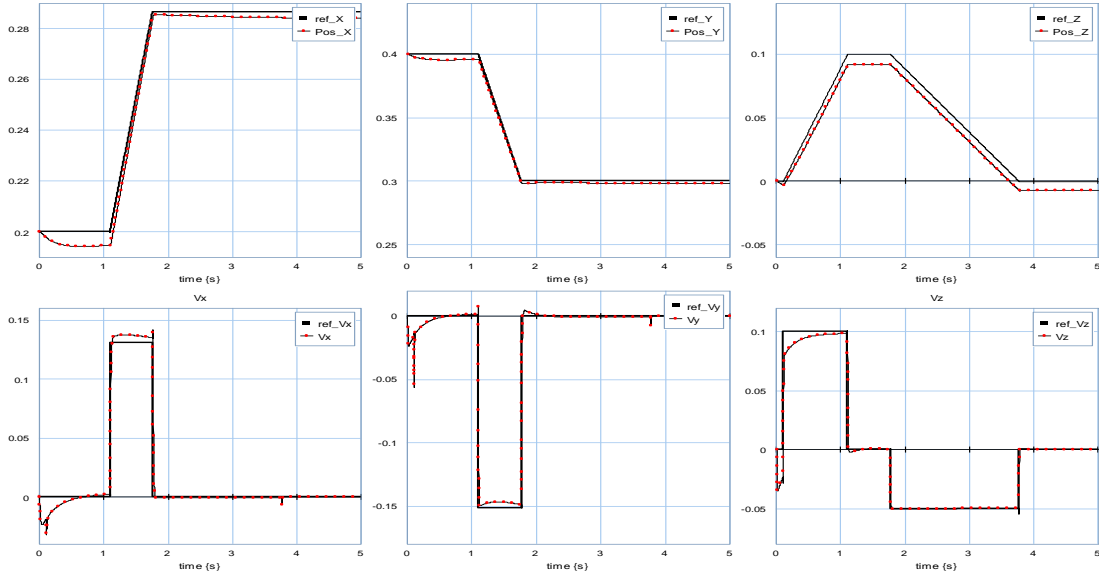
**Table 4.2:** The parameters of the trajectory

And the parameters in the controller are:

R (Proportional Gain)	$\begin{bmatrix} 100 & 0 & 0 \\ 0 & 150 & 0 \\ 0 & 0 & 200 \end{bmatrix}$
K (Derivative Gain)	$\begin{bmatrix} 25 & 0 & 0 \\ 0 & 30 & 0 \\ 0 & 0 & 40 \end{bmatrix}$

**Table 4.3:** parameters of the controller**4.1.1.3 Simulation results and evaluation**

The simulation results are shown in figure 4.2.

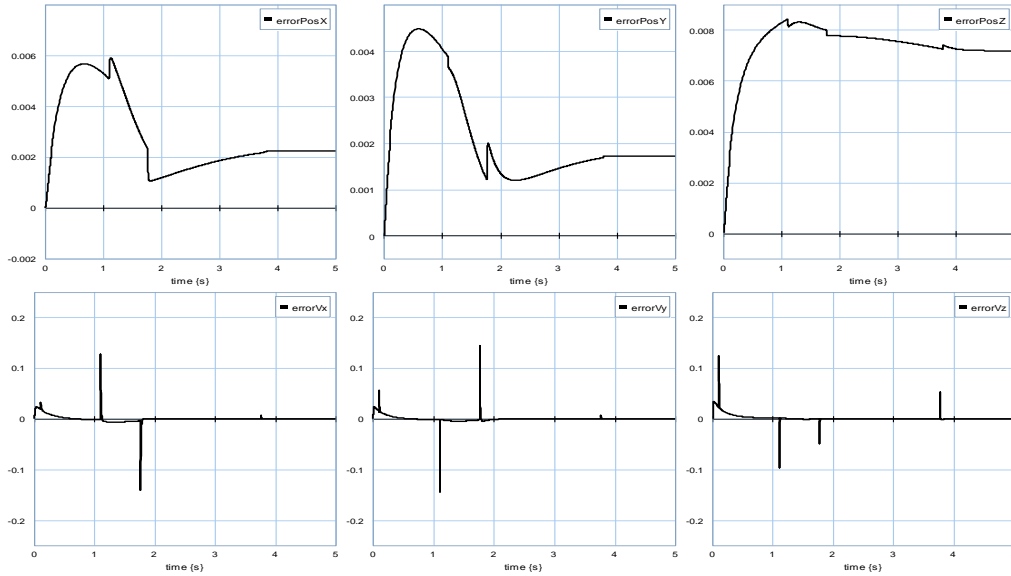
**Figure 4.2:** Simulation results of a transferring leg making a step

This figure shows that in X, Y and Z directions, the foot can follow the trajectory closely with the desired velocities.

Figure 4.3 shows the position and the velocity errors. For the position error, it is less than 0.005m while moves 0.087m in the X direction; the error is smaller than 6%. And in the Y direction, the error stays in 0.005m, which made the error within 0.5%. With Z direction, the foot lifting height is 0.1m and the maximum error is less than 0.008m. This means the error in Z direction is less than 8%.

And for the velocity, the errors in all the directions can converge within 0.1 second.



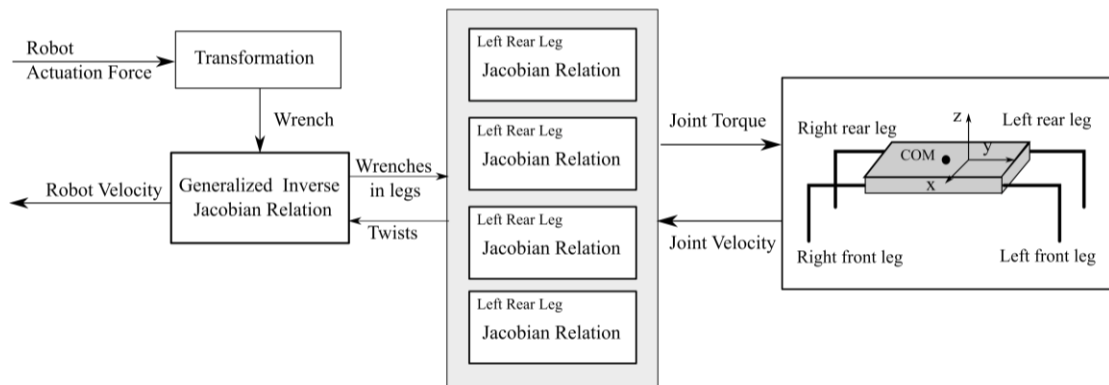


**Figure 4.3:** Simulation results of position and velocity errors

According to the simulation results, the transfer leg can move to the target position with the desired trajectory. Hence, it can be concluded that this subsystem for the transfer leg is functional.

## 4.2 Integration of the components for supporting the robot body

The second integration is to test if the Generalized Inverse of Jacobian relation is effective at supporting and moving the robot body.



**Figure 4.4:** The interconnection between the Generalized Inverse of Jacobian relation unit and the robot

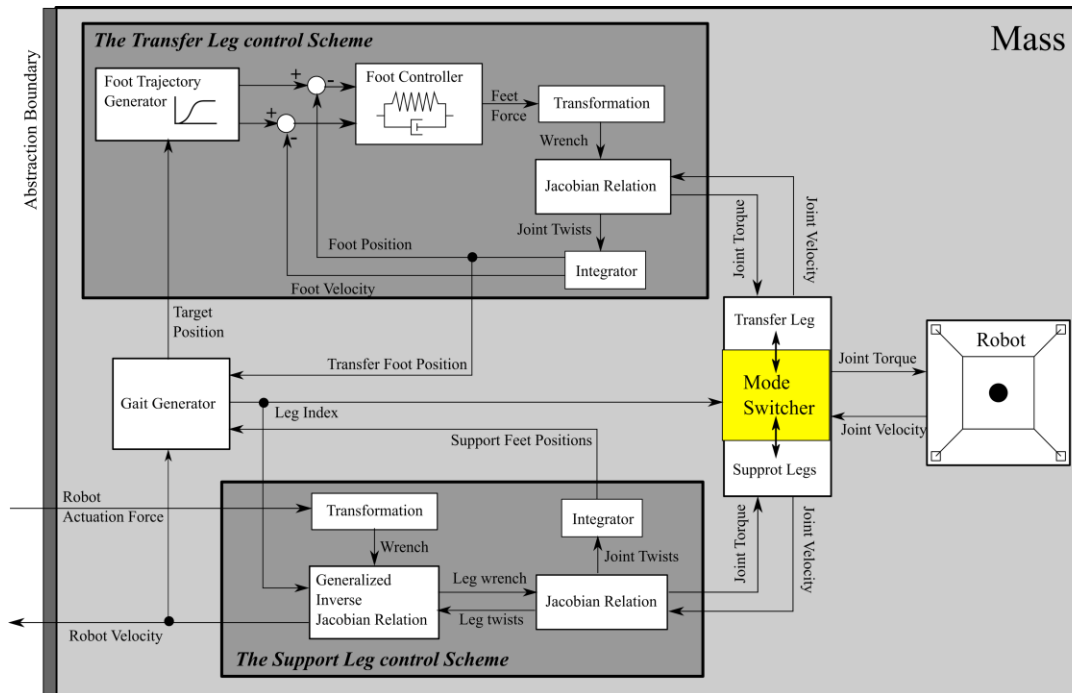
Figure 4.4 shows that when applying a force on the robot body, the generalized inverse of Jacobian will allocate the force to all of its supporting legs. The Jacobian relation converts this to the required joint torque, such that the robot body can be moved by the collaboration of the support legs.

## 4.3 Integration of the entire quadruped robot

The key point to integrate all the components with this quadruped robot is that the robot can get the correct signal of which leg is taking a step and which legs

are being supported. Therefore, a switcher is designed. It helps the legs of the robot to switch between being in transfer mode and being in supporting mode. Figure 4.5 shows the details of the entire interconnections in each component.

All the signals from the robot go to the mode switcher. Based on the leg index calculated by the gait generator, the mode switcher makes that a leg function as the transfer leg and the rest as supporting legs. The transfer leg receives the joint torque signal from all the components in the transfer leg subsystem. And the supporting legs are getting commands from the Generalized Inverse of Jacobian to support and moving the robot body while the transfer leg is making a step. When this transfer leg finished making its step, the Mode Switcher will get a new leg index signal from the Gait Generator.



**Figure 4.5:** A schematic drawing of a controlled quadruped robot

## Chapter 5 Conclusions and recommendation

This chapter presents the conclusion for the entire design as well as recommendations for further work.

### 5.1 Conclusions

Four components in the control structure for a quadruped robot have been designed. The simulation of each component has shown that they are able to function separately.

The partial integration of the components is also being implemented. The partial integration has two parts. One part is implementation of a transfer leg subsystem. This subsystem assured that the transfer leg is able to take a stable and accurate step to a target position in a desired trajectory. The simulation results shows that the leg can move to the desired position in its reachable area.

The other one is the implementation of supporting legs. Those supporting legs can support the robot body with its own body weight or extra loads. And when applying an external force, they can also move the robot body.

A mode switcher is designed to complete the entire integration of the quadruped robot. With this mode switcher, all the components in figure 4.5 can cooperate with each other, and the quadruped robot can walk without losing balance.

Due to lack of project time, the simulation results of section 4.2 and section 4.3 have not been presented in this report.

### 5.2 Recommendations

Due to the lack of project time, the full integration has not been simulated successfully. This could be the first recommendation to continue working on.

Also the profile of the trajectory can be improved. There are more possible trajectory shapes to move a robot leg. For example, the triangular motion, the sinusoidal motion or the semicircle motion. The trajectory profile can, for example, be optimized for energy consumption.

A weighted generalized-inverse (Doty et al. 1992) can be used to improve the calculation of the generalized inverse Jacobian.

## References

1. Arevalo, J.C., Garcia, E.: Impedance control for legged robots: An insight into the concepts involved (2012)
2. Douwe Dresscher, Michiel van der Coelen, Jan Broenink, and Stefano Stramigioli: Control and omni-directional locomotion of a crawling quadruped
3. McGhee, R. B. and A. A. Frank, On the stability properties of quadruped creeping gaits, *Mathematical Biosciences*, **vol. 3**, pp. 331–351. (1968)
4. M. van der Coelen: Design of a Dynamic Free Gait generator for a Quadruped Walking Robot (2013)
5. R. Penrose, “A generalized inverse for matrices,” *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 51, pp. 406–413, 1955Oct. 1955.