Invariance Control for Balance of Legged Robots in 3D

Scientific thesis for the procurance of the degree B.Sc.
from the Department of Electrical and Computer Engineering at the
Technical University of Munich.

Supervised by	Univ.-Prof. Dr.-Ing./Univ. Tokio habil. Martin Buss Dr.-Ing. Marion Leibold Chair of Automatic Control Engineering
Submitted by	cand. ing. Yaseen Sabry El-Zawahry Georg-Deschler-Platz 3 81245 Munich +4915257338745
Submitted on	Munich, 06.08.2018

Abstract

As early as the 1920's the future was imagined to be an exciting and marvelous time seeing the realization of man's imagination, from driving flying cars to working with humanoid robots. It is obvious that after all this time we are still not there, but we are getting there sooner than we imagine. Speaking of humanoid robots, research and advancements in technology in the last decade has resulted in rapid improvements in their effectiveness and reliability.

The problem with humanoid robots is that balancing is an exceedingly complex task, and robust controllers are needed for balancing during locomotion and unexpected disturbances. This thesis explores the utilization of the invariance controller in achieving the balance of a humanoid legged robot. This controller is based on the idea of the manipulation of the Zero Moment Point (ZMP) which is a measure of the robot's postural stability. The idea is to prevent the ZMP from violating certain constraints which guarantee the robot stays balanced. Invariance control sees the appropriate switching between a nominal controller, which is responsible for achieving the robot's nominal task (such as reaching for an object), and a corrective controller, which is responsible for preventing the violation of the ZMP constrains required to maintain balance.

This thesis explores how invariance control could be used to achieve postural stability for a legged robot without having to assume the robot to be planar as has been done before. Strategies are presented which overcome problems associated with the robot being not limited to be planar. These strategies are first demonstrated on a simple two degree of freedom inverted pendulum then finally on a more complex 7 DOF robot with more degrees of freedom.

Acknowledgements

For all of the good songs that kept me motivated the whole time ...

Lucky Chops - A/B Jam
Snarky Puppy - Shofukan
Snarky Puppy - Lingus
Fleetwood Mac - Don’t Stop
Kate Bush - Wurthering Heights
Blondie - Heart of Glass
Camila Cabello - Havana
Toto - Africa
Toto - Make Believe
B.J. Thomas - Raindrops Keep Fallin' On My Head
Brotherhood Of Man - Save Your Kisses For Me
Elton John - Rocket Man
The Five Stairsteps - O-o-h Child
Carpenters - I won’t Last a Day Without You
Carpenters - Only Yesterday
Steve Harley & Cockney Rebel - Make Me Smile

Contents

Chapter 1

Introduction

The research and effort that has been put into the field of humanoid robotics during the last two decades has been outstanding and the field has been advancing ever so significantly. It guarantees that humanoids will no longer be only for entertainment in Sci-Fi movies but will be involved in our everyday life in ways we thought not possible decades ago. An optimistic future sees humanoids taking over hazardous jobs like disaster response, damaged nuclear facility clean ups and space exploration; and in the medical field offering disabled people a chance to regain lost mobility and replace traditional prosthesis; or even being used for entertainment purposes such as toy robots like never seen before. A good example of this today being Boston Dynamics' Atlas robot which stands in the front line of cutting edge robotics showing a fantastic ability to balance, jog through the woods or even do back flips [atl].

Figure 1.1: Boston Dynamics' Atlas Robot [atl].

The ability to balance on legs; although very simple for us to do by the time we're 12 months old, is a difficult task for a legged robot. This difficulty arises because the dynamics involved with the walking of humanoid robots are non-linear, high dimensional and hybrid [PCDG06]. So in order to give a legged robot the ability to balance on its own, it requires an efficient and robust controller design.

One such controller that is both robust and based on simple ideas is the Invariance Controller. The invariance control is based on the simple idea of switching between two different controllers: the nominal controller and the corrective controller. The nominal controller being the one responsible for the fulfillment of the robot's nominal (main) task, which could be anything from walking to reaching for an object to hold. The corrective controller is responsible for keeping certain constraints from violation when needed. These constraints which are related to the so called Zero Moment Point need to be kept from violation as their violation implies that the robot will start to loose it's balance.

In this chapter we first explain the concept of a Zero Moment Point, or a ZMP for short. Then we explain the conditions that the ZMP must satisfy for our bipedal robot to remain balanced. Then related work which is also based on the idea of ZMP manipulation (as the invariance controller) is discussed before we demonstrate how a simple 3D robot can be modelled for the implementation of our invariance controller and for simulation later on.

1.1 Zero moment point

The concept of the Zero Moment Point was introduced by Vukobratović [VBv01] as a point on the ground with respect to which the sum of the moments of the dynamic reaction forces at the robot's feet and ground contact area has zero components in the horizontal direction. It is therefore also the point on the ground where the resultant contact force has to act to balance the robot.

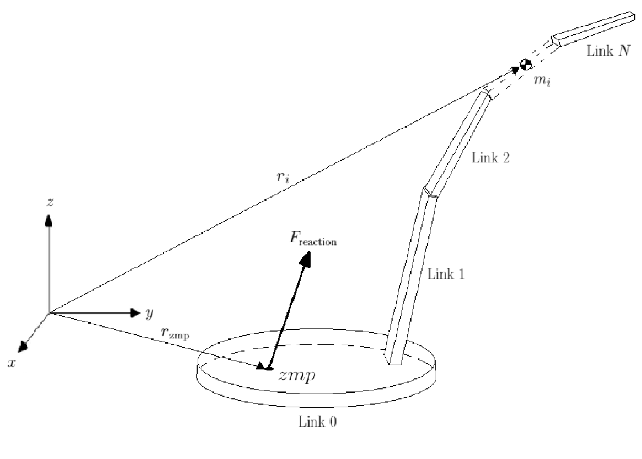


Figure 1.2: The ZMP Concept.

The concept of a ZMP relies on two major assumptions. First, that the contact area between the robot's feet and the ground is planar, and second, that the ground has sufficiently high friction to keep the feet from sliding.

In Fig. 1.2 one leg of a typical legged robot is shown. As seen, it consists of the foot which is linked to the rest of the robot through rigid links connected by revolute joints (not visualized here). Let the robot be formed of $(N + 1)$ links (links are numbered from 0 to N) with each i^{th} link having mass m_i and position of center of mass $\boldsymbol{r}_i = (x_i, y_i, z_i)^{\mathrm{T}}$. The robot is assumed to have flat feet which are flat on a flat ground. The frame of reference is chosen to be inertial and having its x and y axes parallel to the ground. The ZMP is then located somewhere on the ground with position specified by $\boldsymbol{r}_{\text{zmp}} = (r_{\text{zmp},x}, r_{\text{zmp},y}, r_{\text{zmp},z})^{\mathrm{T}}$, here $r_{\text{zmp},z}$ has a constant value to put the ZMP on the ground's surface and is irrelevant. For such a robot the coordinates of the ZMP are derived by K. Erbatur et al. [EOO+02] using D'Alambert's Principle [BJ90] yielding the following:

$$r_{\text{zmp},x} = \frac{\sum\limits_{i=0}^{N} m_i(\ddot{z}_i - g_z)x_i - \sum\limits_{i=0}^{N} m_i(\ddot{x}_i - g_x)z_i}{\sum\limits_{i=0}^{N} m_i(\ddot{z}_i - g_z)} \tag{1.1}$$

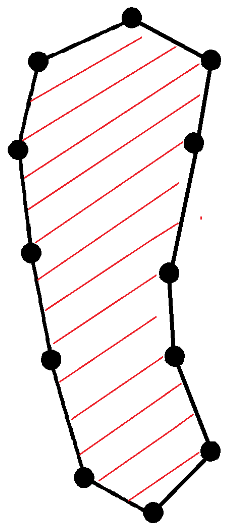
$$r_{\text{zmp,y}} = \frac{\sum\limits_{i=0}^{N} m_i(\ddot{z}_i - g_z)y_i - \sum\limits_{i=0}^{N} m_i(\ddot{y}_i - g_y)z_i}{\sum\limits_{i=0}^{N} m_i(\ddot{z}_i - g_z)} \tag{1.2}$$

Here g_x, g_y and g_z are the components of gravity resolved to the x, y and z directions respectively of our chosen reference frame. In the case that the ground is horizontal g_x and g_y are both zero and g_z is simply $-g$ for a z-axis pointing upwards; and in the case that the ground is inclined the values of g_x, g_y and g_z are obtained by resolving $\boldsymbol{g}$ in the x, y and z directions respectively.

Here it is assumed that the joint angles have small velocities and accelerations; and that the robot's links have small enough rotational inertia. This enables us to treat links as point masses at the link's center of mass, this in turn greatly simplifies the expressions for the ZMP's coordinates as well as the robot's equations of motion. This assumption helps reduce the complexity associated with having a robot with several degrees of freedom and helps reduce simulation times.

1.2 Balancing conditions

As mentioned in the previous section, the ZMP is the point on the ground where the resultant contact force *has* to act to balance the robot. So as long as the contact forces between the robot's feet and the ground have a distribution that places their resultant at the ZMP's location the robot stays balanced. This is possible only if the ZMP remains inside the area formed by the convex hull of the contact points between the robot's feet and the ground [con], because the resultant force can only act inside this area. In the case of contact with a single foot, this area is simply the contact area between the robot's foot and the ground as shown in Fig. 1.3(a); and for the case of contact with two feet Fig. 1.3(b) shows the area the ZMP is allowed to remain in for the robot to stay balanced. Let us call this area the safe area.


(a) Ground contact with one foot.

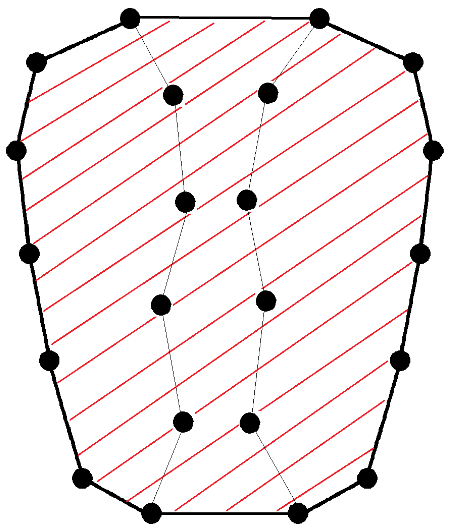

(b) Ground contact with two feet.

Figure 1.3: Safe areas for different contact situations

As soon as the ZMP drifts outside the safe area, the robot will start to tip over the edges of its feet and fall in the direction of the ZMP's exit, and here lies our main problem. The ZMP's location depends on the positions and the accelerations of the links as seen in (1.1) and (1.2), which in turn depend on the input torques provided by the robot's actuators. The torques provided by the robot's actuators are determined by the nominal controller which is not concerned at all with the ZMP's location but with some nominal task such as having the joint angles follow a certain trajectory. So it is not unlikely that at some point our nominal controller results in actuator torques which cause the ZMP to exit the safe area. Therefore, any controller which successfully keeps the robot balanced should either directly or indirectly be manipulating the ZMP to stay in the safe area.

1.3 Related Work

As was discussed in Sec. 1.2, the Zero Moment Point which was introduced in Sec. 1.1 has constraints which must be satisfied for the robot to remain balanced and avoid tipping over the foot edges and start falling. To maintain balance control we require a balance controller which has influence over the ZMP's dynamics so as to limit where the ZMP can be and avoid it exiting the safe area. Such a controller which has influence over the ZMP is said to be one based on ZMP manipulation. Several approaches to achieve balance control are based on ZMP manipulation.

The ZMP was first introduced by Vukobratović [VBv01] as an indicator of the robot's mechanism behaviour as well as a measure of the robustness of the robot's dynamic equilibrium. Experimental studies exist such as in [HNI01] where dynamic patterns are generated so as to satisfy ZMP constraints. This is achieved by the modulation of the ankle's joint trajectory based on sensory reflexes. This modulation is a function of the difference between the actual ZMP trajectory obtained using sensory data and the desired ZMP trajectory. This method proved experimental success; however, a difficulty in simulation arises due to the existence of an algebraic loop due to the dependency of the ZMP on the input torque existing along with a dependency of the input torque on the ZMP. It is pointed out by [SWB07] that the accounting of actuator dynamics as well as link flexibilities would negate this ZMP's dependence on the input torque.

In [KH03] a ZMP-plane which changes with respect to time is used which includes all the allowable accelerations that imply dynamics that guarantee that the ZMP is kept at a certain value. First, the desired acceleration is evaluated. If this desired acceleration implies the violation of the ZMP constraints then it is projected onto the ZMP-plane so as to fix the ZMP at its admissible boundary.

In [SWB07] the invariance controller is used for ZMP manipulation so as to achieve balance control for legged robots. Due to the work being based on planar legged robots, the ZMP's position only had to be manipulated so to restrict it's position to be on a line between two points representing the two extremes of the robot's foot, making it a *safe line* rather than a safe area.

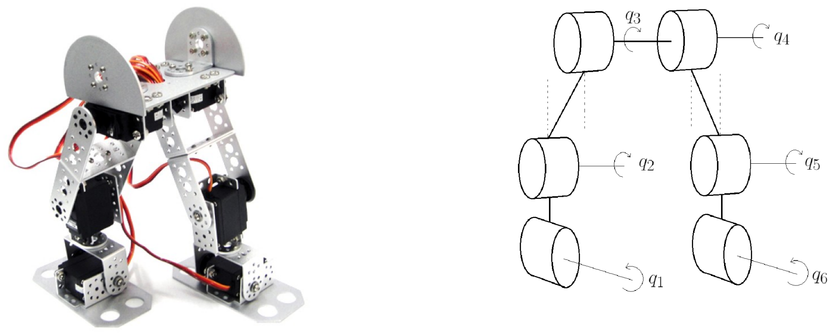
The bounds of stability for the invariance controller which was introduced in [SWB07] are explored in [EB16] to determine the region in state space where balance control is still possible. This is to be able to determine when balance control by invariance control is successful and to be used, otherwise stepping control could be used which requires the robot to change the foot and ground contact points by stepping.

Lacking from [SWB07] and [EB16] was the implementation of invariance control for 3D legged robots, which is essential for the possibility of experimental testing of invariance control in future work. This thesis explores the techniques and methods used for applying invariance control for a 3D legged robot in a single support phase. A simple algorithm is presented which can handle the motion of the ZMP in two dimensions instead of only one, the algorithm is responsible for constraining the ZMP in a safe area bounded by a polygon with a non-trivial shape.

This thesis is organized as follows. The next section explains the techniques used to model a legged robot in 3D. Chapter 2 shows how the nominal and corrective controllers which are the underlying controllers of the invariance controller are formulated. Chapter 3 shows how the nominal and corrective controllers are combined so as to form the invariance controller. Then finally in Chap. 4 the application of the invariance controller is demonstrated for a 2 DOF and a 7 DOF robot to demonstrate their effectiveness.

1.4 Robot Modelling

The typical legged robot is consisted of a number of rigid links connected by revolute joints. Each of these revolute joints will have a joint angle q which contributes to the overall robot form. For a robot with flat feet which are flat on a flat ground like in Fig. 1.4(a) the positions of all the links are functions of the joint angles and only the joint angles. In this case we have joint angles from q_1 till q_6, so we can formulate a mathematical model of the robot which is fully described by $\boldsymbol{q} = (q_1, \cdots, q_6)^{\mathrm{T}}$ and its time derivatives.



(a) A simple 6 DOF biped robot [rob]. (b) Simplified model of the 6 DOF robot.

Figure 1.4: Example of a 6 DOF robot and its mathematical model

We can generalize this for the typical legged robot with n joints which has flat feet on a flat ground (or one flat foot flat on the flat ground) and have a mathematical model formed by a set of minimal coordinates chosen to be the joint angles:

$$\boldsymbol{q} = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{pmatrix} \in \mathbb{R}^n$$

If the i^{th} joint is actuated by a torque τ_i we can let the vector $\boldsymbol{\tau} = (\tau_1, \cdots, \tau_n)^{\mathrm{T}}$ represent all of our actuator torques. These two vectors $\boldsymbol{\tau}$ and $\boldsymbol{q}$ are part of the robot's equations of motion expressed in the general form:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau} \tag{1.3}$$

Here $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix which represents the inertias of the robot's links and depends on the robot's configuration, and $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ represents the gravitational, Coriolis and centrifugal forces and torques. The determination of $\boldsymbol{M}(\boldsymbol{q})$ and $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ can be done using Lagrangian formulation as discussed in Sec. 1.4.2, but first we need to determine the positions of the links centers of mass as functions of $\boldsymbol{q}$ as will be done in the next section.

1.4.1 Transformations Matrices

Imagine having a point P in space and two different reference frames $O_1 - x_1y_1z_1$ and $O_2 - x_2y_2z_2$ as shown in Fig. 1.5. The position of P can be described by the vector $\boldsymbol{p}^1$ relative to the first frame and by $\boldsymbol{p}^2$ relative to the second frame. Although both $\boldsymbol{p}^1$ and $\boldsymbol{p}^2$ describe the same point in space $\boldsymbol{p}^1 \neq \boldsymbol{p}^2$ because both of them are defined relative to different reference frames. However, we can obtain $\boldsymbol{p}^1$ from $\boldsymbol{p}^2$ (or the other way round) using transformation matrices.

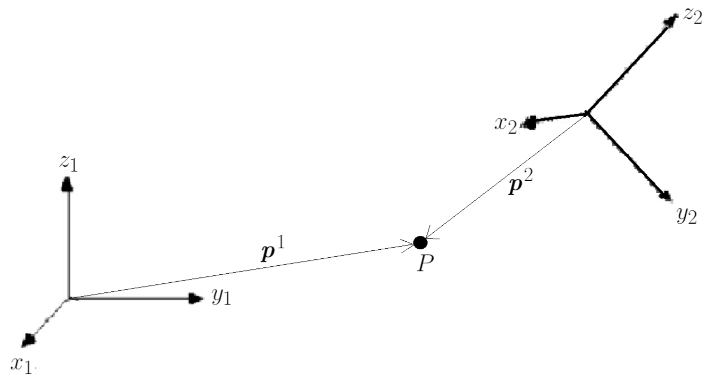


Figure 1.5: A point P in space viewed from different frames.

First, if we take a look at Fig. 1.5 we can see that the second frame is related to the first frame by geometric transformations (translations and rotations); if we first translate the first frame such that its origin is the same as the second frame's origin, then rotate this new frame about its x-axis by some angle θ_x, then do the same with the new y-axis with angle θ_y and then do the same with the new z-axis with an angle θ_z we can get a frame that coincides with the second frame.

The transformations we used to transform the first frame to the second frame can be represented by 4×4 matrices as follows:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad Trans(\delta x, \delta y, \delta z) = \begin{pmatrix} 1 & 0 & 0 & \delta x \\ 0 & 1 & 0 & \delta y \\ 0 & 0 & 1 & \delta z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Now we can represent the transformation from the first frame to the second frame with a matrix $\boldsymbol{A}_2^1$ defined by:

$$\boldsymbol{A}_2^1 = Trans(\delta x, \delta y, \delta z) R_x(\theta_x) R_y(\theta_y) R_z(\theta_z)$$

with the appropriate values of δx, δy, δz, θ_x, θ_y and θ_z. So now we can obtain $\boldsymbol{p}^1$ from $\boldsymbol{p}^2$ using:

$$\begin{pmatrix} \boldsymbol{p}^1 \\ 1 \end{pmatrix} = \boldsymbol{A}_2^1 \begin{pmatrix} \boldsymbol{p}^2 \\ 1 \end{pmatrix}$$

Now using this concept of transformation matrices it is now desired to find the center of mass positions for the $(N+1)$ links of our robot as functions of $\boldsymbol{q}$. To do this we apply the same concept as from the previous example, the only difference is that in this case the transformation matrices will be functions of $\boldsymbol{q}$. It will be assumed the robot is in a single support phase (standing on one foot) and that this one foot is flat on a flat ground; this will be the assumed throughout this thesis.

First a base frame must be defined, the only requirement is that it must be inertial, for instance we can attach it to the supporting foot's center of mass. Then using the Denavit–Hartenberg Convention [1] we define a coordinate frame attached to each link, from link 0 to link N [SSVO10]. Then for each i^{th} joint we calculate the DH parameters that are associated with the transformation from link $(i-1)$'s frame to link i's frame, so that for each i^{th} joint we have values for a_i, d_i, α_i and θ_i (For revolute joints θ_i will depend on the joint angle q_i). These values are then plugged into the DH matrix:

$$\boldsymbol{A}_i^{i-1}(q_i) = DH(a_i, \alpha_i, d_i, \theta_i) = \begin{pmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

[1]The Denavit–Hartenberg Convention is an easy but long topic and won't be discussed here.

Here $\boldsymbol{A}_i^{i-1}(q_i)$ is a transformation matrix associated with the i^{th} joint which connects link $(i-1)$ to link i as seen in Fig.1.6.

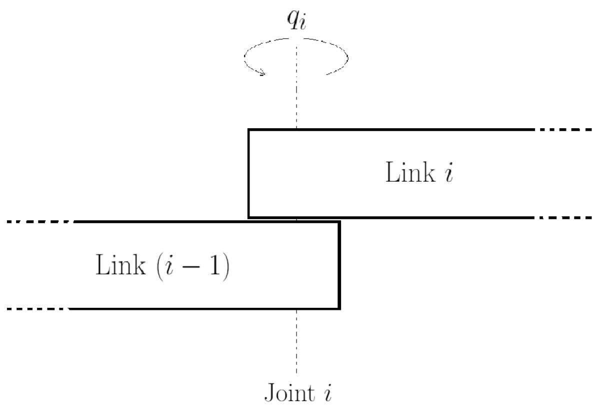


Figure 1.6: Link $(i-1)$ joined to link i by jonit i.

Now if want the transformation matrix $\boldsymbol{T}_j^0(\boldsymbol{q})$, which transforms from link 0's frame to link j's frame, we can obtain it simply by:

$$\boldsymbol{T}_j^0(\boldsymbol{q}) = \boldsymbol{A}_1^0(q_1)\boldsymbol{A}_2^1(q_2)\cdots\boldsymbol{A}_j^{j-1}(q_j)$$

It is also possible that some link $(i-1)$ connects to two other links $i^{'}$ and $i^{''}$ by two different joints with joint angles $q_{i^{'}}$ and $q_{i^{''}}$ respectively, an example of this branching of joints is shown in Fig. 1.7.

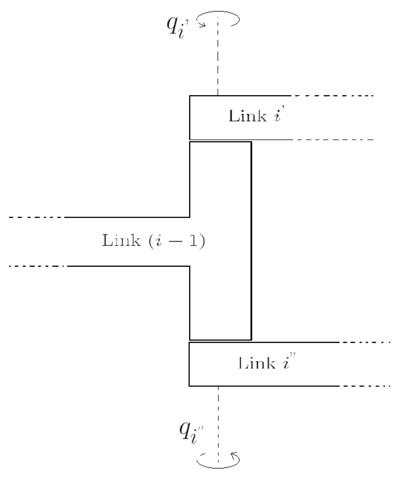


Figure 1.7: Example of joints branching.

The concept here is still the same but instead of having $\boldsymbol{A}_i^{i-1}(q_i)$ as before we now would have $\boldsymbol{A}_{i'}^{i-1}(q_{i'})$ and $\boldsymbol{A}_{i''}^{i-1}(q_{i''})$, and based on which path is taken along the chain of links to reach the link of interest j' (or j'') we can have:

$$\boldsymbol{T}_{j'}^0(\boldsymbol{q}) = \boldsymbol{A}_1^0(q_1)\boldsymbol{A}_2^1(q_2)\cdots\boldsymbol{A}_{i'}^{i-1}(q_{i'})\cdots\boldsymbol{A}_{j'}^{j'-1}(q_{j'})$$

or,

$$\boldsymbol{T}_{j''}^0(\boldsymbol{q}) = \boldsymbol{A}_1^0(q_1)\boldsymbol{A}_2^1(q_2)\cdots\boldsymbol{A}_{i''}^{i-1}(q_{i''})\cdots\boldsymbol{A}_{j''}^{j''-1}(q_{j''})$$

Note here that $q_{i'}$, $q_{i''}$, $q_{j'}$ and $q_{j''}$ are elements from the vector $\boldsymbol{q}$.

We then define the transformation matrix $\boldsymbol{T}_0^b$ (typically constant) to be one which transforms from our base frame to link 0's frame. It can be obtained using the transformation matrices for translation and rotation. Now we can calculate transformation $\boldsymbol{T}_j^b$ that transforms from the base frame to link j's frame:

$$\boldsymbol{T}_j^b(\boldsymbol{q}) = \boldsymbol{T}_0^b\boldsymbol{T}_j^0(\boldsymbol{q})$$

Links j's frame; however, does not necessarily coincide with link j's center of mass. So we need to define one more transformation matrix $\boldsymbol{T}_{\mathrm{CoM}_j}^j$ which transforms from link j's frame (which was chosen according to the DH convention) to any (chosen) frame which has it's origin coinciding the link j's center of mass, this transformation is typically constant and is obtained similarly as $\boldsymbol{T}_0^b$. So now we can obtain the transformation matrix $\boldsymbol{T}_{\mathrm{CoM}_j}^b(\boldsymbol{q})$ which transforms from the base frame to link j's center of mass using:

$$\boldsymbol{T}_{\mathrm{CoM}_j}^b(\boldsymbol{q}) = \boldsymbol{T}_0^b\boldsymbol{T}_j^0(\boldsymbol{q})\boldsymbol{T}_{\mathrm{CoM}_j}^j$$

Note that for link 0, $\boldsymbol{T}_{\mathrm{CoM}_0}^b$ is typically constant and found from:

$$\boldsymbol{T}_{\mathrm{CoM}_0}^b = \boldsymbol{T}_0^b\boldsymbol{T}_{\mathrm{CoM}_0}^0$$

The position vector $\boldsymbol{p}_j = (x_j, y_j, z_j)^{\mathrm{T}}$ for link j's center of mass relative to the base frame is simply the origin of link j's center of mass frame viewed relative to the base frame so:

$$\begin{pmatrix} x_j \\ y_j \\ z_j \\ 1 \end{pmatrix} = \boldsymbol{T}_{\mathrm{CoM}_j}^b(\boldsymbol{q}) \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

and so to obtain $\boldsymbol{p}_j$ directly:

$$\boldsymbol{p}_j = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \boldsymbol{T}_{\mathrm{CoM}_j}^b(\boldsymbol{q}) \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

The velocity $\boldsymbol{v}_j$ and acceleration $\boldsymbol{a}_j$ of the center of mass of link j can be found by differentiating $\boldsymbol{p}_j$ and $\boldsymbol{v}_j$ respectively with respect to time:

$$\boldsymbol{v}_j = \frac{d\boldsymbol{p}_j}{dt} = (\dot{x}_j, \dot{y}_j, \dot{z}_j)^{\mathrm{T}}$$

$$\boldsymbol{a}_j = \frac{d\boldsymbol{v}_j}{dt} = (\ddot{x}_j, \ddot{y}_j, \ddot{z}_j)^{\mathrm{T}}$$

1.4.2 Lagrange formulation

Lagrange formulation enables us to derive the equations of motion of a system systematically and independently from a reference frame [SSVO10]. This makes the derivation of the equations of motion a lot easier than calculating the resultant force and torque for each member of the system and then using Newton's second law of motion (which is good enough for simpler systems). First we have to define a set of variables known as *generalized coordinates* that completely define the configuration of our system, in this case it is the vector $\boldsymbol{q} = (q_1, \cdots, q_n)^{\mathrm{T}}$ which effectively describes the locations of all the links of our n-DOF robot as shown in the previous section, which is assumed to be in a single support phase with one flat foot flat on the ground.

First, we define the *Lagrangian* of our system as a function of the generalized coordinates:

$$\mathcal{L} = \mathcal{T} - \mathcal{U}$$

Here $\mathcal{T}$ denotes the total kinetic energy of the system and $\mathcal{U}$ the total potential energy of the system. These quantities can be determined from the summations of the kinetic energy and potential energy for links 0 to N:

$$\mathcal{T} = \sum_{i=0}^{N} \frac{1}{2} m_i \|\boldsymbol{v}_i\|^2 = \sum_{i=0}^{N} \frac{1}{2} m_i (\dot{x}_j^2 + \dot{y}_j^2 + \dot{z}_j^2)$$

$$\mathcal{U} = \sum_{i=0}^{N} -m_i (\boldsymbol{g} \cdot \boldsymbol{p}_i) = \sum_{i=0}^{N} -m_i (g_x x_i + g_y y_i + g_z z_i)$$

Here m_i is the i^{th} link's mass, $\boldsymbol{p}_i$ and $\boldsymbol{v}_i$ are respectively the position and velocity of its center of mass. The vector $\boldsymbol{g} = (g_x, g_y, g_z)^{\mathrm{T}}$ is the gravity vector. Here $\boldsymbol{p}_i$, $\boldsymbol{v}_i$ and $\boldsymbol{g}$ are all resolved and relative to the chosen (inertial) base frame.

Now assuming the joints with joint angles $\boldsymbol{q} = (q_1, \cdots, q_n)^{\mathrm{T}}$ are actuated by the torques $\boldsymbol{\tau} = (\tau_1, \cdots, \tau_n)^{\mathrm{T}}$ and there are no dissipating forces throughout our system (such as friction at the joints), the Lagrange equations are expressed as:

$$\begin{pmatrix} \dfrac{d}{dt}\dfrac{\partial \mathcal{L}}{\partial \dot{q}_1} - \dfrac{\partial \mathcal{L}}{\partial q_1} \\ \vdots \\ \dfrac{d}{dt}\dfrac{\partial \mathcal{L}}{\partial \dot{q}_n} - \dfrac{\partial \mathcal{L}}{\partial q_n} \end{pmatrix} = \boldsymbol{\tau}$$

This can be expressed in the compact form:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}}\right)^{\mathrm{T}} - \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{q}}\right)^{\mathrm{T}} = \boldsymbol{\tau} \tag{1.4}$$

These equations establish the relations existing between $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ (the joint angle positions, velocities and accelerations); and the torques applied at the joints. Hence they allow us to derive the dynamic model of our system and their rearrangement gives us (1.3), so then the expressions of $\boldsymbol{M}(\boldsymbol{q})$ and $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ can be obtained as required.

Chapter 2

Nominal and Corrective Controllers

Any functional robot is designed so that it performs one or more nominal (main) tasks. A biped robot for instance can have nominal tasks such as walking, reaching for an object to grab or even standing on one leg to display the ability to balance. The controller responsible for outputting the correct torques from the robot's actuators to see the performance of these nominal tasks is termed the nominal controller. In this thesis a corrective controller is one which has the sole task of outputting the correct torques from the robot's actuators that result in link positions and accelerations which result in the ZMP to be restricted at certain point or to move on a certain line. This chapter will first demonstrate how a simple nominal controller can be derived and then how different corrective controllers for different ZMP constraining situations can be derived.

2.1 Control System

To define our control system, we first choose a convenient state vector. From (1.3) a good choice would be:

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{pmatrix} \in \mathbb{R}^{2n}$$

with n being the number of joints. Taking $\boldsymbol{\tau}$ to be our control input $\boldsymbol{u}$, our system is therefore:

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{pmatrix} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}(\boldsymbol{x})[\boldsymbol{u} - \boldsymbol{n}(\boldsymbol{x})] \end{pmatrix} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ -\boldsymbol{M}^{-1}(\boldsymbol{x})\boldsymbol{n}(\boldsymbol{x}) \end{pmatrix} + \begin{pmatrix} 0 \\ \boldsymbol{M}^{-1}(\boldsymbol{x}) \end{pmatrix} \boldsymbol{u}$$

which can be expressed in the form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u} \tag{2.1}$$

A system such as this is termed a *nonlinear control-affine system*, meaning that it is linear with respect to the input and non-linear with respect to the state.

2.2 Nominal Controller

Although several possibilities for nominal controllers exist for a legged robot, we will choose a simple one to be used throughout this thesis for demonstration purposes. Such a simple nominal controller can be one responsible for having the joints track desired joint angles in the form:

$$q_{\text{des},i} = A_i \sin(\omega_i t) + c_i$$

which is simply having each i^{th} joint angle oscillate around some angle c_i with angular frequency ω_i and amplitude A_i. At first thought, it may be suggested to use a simple PD controller as shown:

$$\boldsymbol{u} = \boldsymbol{\tau} = \boldsymbol{K}_D(\dot{\boldsymbol{q}}_{\text{des}} - \dot{\boldsymbol{q}}) + \boldsymbol{K}_P(\boldsymbol{q}_{\text{des}} - \boldsymbol{q})$$

with $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$ being the parameters of our PD controller and defined as the following diagonal matrices:

$$\boldsymbol{K}_P = \begin{pmatrix} K_{P,1} & 0 & \dots & 0 \\ 0 & K_{P,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & K_{P,n} \end{pmatrix} \qquad \boldsymbol{K}_D = \begin{pmatrix} K_{D,1} & 0 & \dots & 0 \\ 0 & K_{D,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & K_{D,n} \end{pmatrix}$$

The problem with this; however, is that this PD controller is a linear controller and our system is a non-linear one as shown in (2.1). A linear system would be one where (2.1) would be in the form $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$ instead. The use of this linear PD control law on our nonlinear system can give a bad response where the tracking error is not guaranteed to converge to zero and when it does it is a relatively slow convergence [SK08]. To get over this we use a computed-torque controller which is based on the feedback linearization principle [Kha02]. This principle allows us to map a nonlinear model such as ours to an equivalent linear one, consequently we would be able to use a linear controller such as a this PD controller. We begin first by rewriting (1.3) as:

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})[\boldsymbol{\tau} - \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})] \tag{2.2}$$

If we can somehow cancel the non-linear terms $\boldsymbol{M}(\boldsymbol{q})$ and $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, then our system is effectively converted to a linear one. Using our prior knowledge of the expressions for $\boldsymbol{M}(\boldsymbol{q})$ and $\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, we can smartly choose $\boldsymbol{\tau}$ to be:

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})[\boldsymbol{u}] + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{2.3}$$

where $\boldsymbol{u}$ is an auxiliary (helping) control signal now and not the same as $\boldsymbol{\tau}$. If we substitute (2.3) into (2.2) we get:

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})[\boldsymbol{M}(\boldsymbol{q})[\boldsymbol{u}] + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})] \tag{2.4}$$

so we have the equivalent linear system:

$$\ddot{\boldsymbol{q}} = \boldsymbol{u} \tag{2.5}$$

Our system now can be expressed as:

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \boldsymbol{u}$$

which is in effect a linear system and can be controlled effectively using our PD controller. We see from (2.5) that when in steady state where the tacking error $\boldsymbol{e} = (\boldsymbol{q}_{des} - \boldsymbol{q})$ and all of its derivatives are approximately zero, so we should have $\ddot{\boldsymbol{q}} = \boldsymbol{u} \approx \ddot{\boldsymbol{q}}_{des}$. Therefore we choose $\boldsymbol{u}$ to be:

$$\boldsymbol{u} = \ddot{\boldsymbol{q}}_{des} + \boldsymbol{K}_D(\dot{\boldsymbol{q}}_{\mathrm{des}} - \dot{\boldsymbol{q}}) + \boldsymbol{K}_P(\boldsymbol{q}_{\mathrm{des}} - \boldsymbol{q})$$

instead of only

$$\boldsymbol{u} = \boldsymbol{K}_D(\dot{\boldsymbol{q}}_{\mathrm{des}} - \dot{\boldsymbol{q}}) + \boldsymbol{K}_P(\boldsymbol{q}_{\mathrm{des}} - \boldsymbol{q})$$

and now to obtain $\boldsymbol{\tau}$ all we have to do is to substitute $\boldsymbol{u}$ in to (2.3).

2.3 Corrective Controller

The corrective controller is responsible for outputting the correct torques from the robot's actuators that result in link positions and accelerations which result in the ZMP to be restricted at certain point or to move on a certain line. In this section we will show how we can formulate the corrective controller that can restrict the ZMP to move on a certain line, as well as the corrective controller that can restrict the ZMP at a certain point. In the next chapter it will be shown how the corrective controllers along with the nominal controller discussed in Sec. 2.2 can be combined to form the invariance controller for controlling our robot.

2.3.1 ZMP Equations

It was shown in (1.1) and (1.2) that the coordinates of the ZMP are functions of the positions and accelerations of the links' centers of mass and in Sec. 1.4.1 its was shown how these positions and accelerations could be found using transformation matrices. The position of the i^{th} link's center of mass is a function of $\boldsymbol{q}$:

$$\boldsymbol{p}_i(\boldsymbol{q}) = \begin{pmatrix} x_j(\boldsymbol{q}) \\ y_j(\boldsymbol{q}) \\ z_j(\boldsymbol{q}) \end{pmatrix}$$

and its acceleration is a function of $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$:

$$\boldsymbol{a}_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \begin{pmatrix} \ddot{x}_j(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \\ \ddot{y}_j(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \\ \ddot{z}_j(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \end{pmatrix}$$

and because we're using computed torque control and $\ddot{\boldsymbol{q}}$ is related to $\boldsymbol{u}$ by (2.5) we can restate the previous statement as:

$$\boldsymbol{a}_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) = \begin{pmatrix} \ddot{x}_j(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \\ \ddot{y}_j(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \\ \ddot{z}_j(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \end{pmatrix}$$

and so assuming our base frame was chosen to have its x and y axes parallel to the ground, and knowing the fact that the ZMP lies somewhere on the surface of the ground, we could define the 2D position vector representing the ZMP's location relative to the base frame's x and y directions (the z coordinate is irrelevant) as functions of $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$:

$$\boldsymbol{r}_{\text{zmp}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) = \begin{pmatrix} r_{\text{zmp},x}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \\ r_{\text{zmp},y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \end{pmatrix}$$

This expression can be found by substituting the expressions for the positions and the accelerations of the links' centers of mass from Sec. 1.4.1 into (1.1) and (1.2).

2.3.2 Line Constraining

Assume that we would like to restrict the ZMP's motion such that it can only move on a line l_i on the xy-plane of our base frame projected on to the ground described by the two parameters α_i and $d_{0,i}$ as shown in

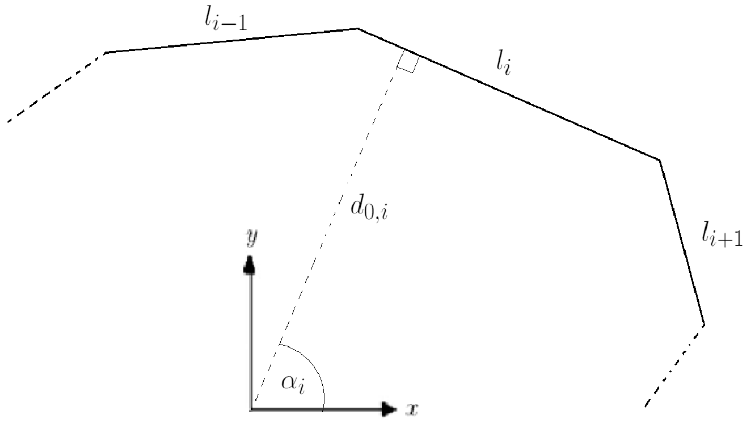


Figure 2.1: A general line l_i described by the parameters α_i and $d_{0,i}$.

Here $d_{0,i}$ is the length of the shortest line joining between our origin and the line l_i; and α_i is the angle this shortest line makes with the positive x-axis. We then define the vector $\boldsymbol{r}_{\mathrm{zmp}_\alpha}$ to be the position vector of the ZMP but viewed from a reference frame rotated by α about the positive z-axis from our base frame as shown in Fig. 2.2.

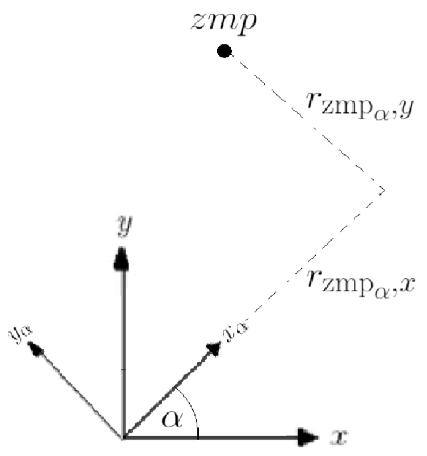


Figure 2.2: The ZMP viewed relative to the $x_\alpha y_\alpha$ axes.

We can obtain $\boldsymbol{r}_{\mathrm{zmp}_\alpha}$ by observing the equivalence between it and simply rotating the ZMP (visually) by $-\alpha$ about the z-axis and viewing it from the base frame, so we have:

$$\boldsymbol{r}_{\mathrm{zmp}_\alpha}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) = \begin{pmatrix} \cos(-\alpha) & -\sin(-\alpha) \\ \sin(-\alpha) & \cos(-\alpha) \end{pmatrix} \boldsymbol{r}_{\mathrm{zmp}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) = \begin{pmatrix} r_{\mathrm{zmp}_\alpha,x}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \\ r_{\mathrm{zmp}_\alpha,y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \end{pmatrix}$$

Now our problem of wanting to constrain the motion of the ZMP to be on the line l_i is simply requiring:

$$r_{\mathrm{zmp}_{\alpha_i},x}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) = d_{0,i} \tag{2.6}$$

we only need to solve (2.6) for the control signal $\boldsymbol{u}$ required. Fortunately, this equation is linear in $\boldsymbol{u}$ and can be rewritten as:

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}$$

or more detailed:

$$\begin{pmatrix} a_1 & \cdots & a_n \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} b \end{pmatrix}$$

The problem here is that unless $n = 1$, this is an undetermined system and no unique solution exists. However, if we specify that we want our solution $\boldsymbol{u}$ to be one having minimal l_2-norm $\|\boldsymbol{u}\|_2 = \sqrt{u_1^2 + \cdots + u_n^2}$, we can use the pseudo inverse of $\boldsymbol{A}$ to find the a unique solution [SWB07]:

$$\boldsymbol{u}_{\mathrm{cor}} = \boldsymbol{A}^+\boldsymbol{b}$$

For reasons that will become more apparent later on, we may want a solution for $\boldsymbol{u}_{\mathrm{cor}}$ that is as close as possible to the nominal controller signal $\boldsymbol{u}_{\mathrm{nom}}$, so we want to have a solution with minimal l_2-norm $\|\boldsymbol{u} - \boldsymbol{u}_{\mathrm{nom}}\|_2 = \sqrt{(u_1 - u_{\mathrm{nom}_1})^2 + \cdots + (u_n - u_{\mathrm{nom}_n})^2}$, this solution can be obtained using [SWB07]:

$$\boldsymbol{u}_{\mathrm{cor}} = \boldsymbol{A}^+\boldsymbol{b} + (\boldsymbol{I} - \boldsymbol{A}^+\boldsymbol{A})\boldsymbol{u}_{\mathrm{nom}}$$

where $\boldsymbol{I}$ is the identity matrix. We can even go one step further and define a diagonal weights matrix $\boldsymbol{W}$ which based on its i^{th} element along the diagonal $w_{i,i}$ compared with all the other diagonal elements determines how much the i^{th} joint should be participating constraining the ZMP. This requires a solution which has a minimal l_2-norm $\|\boldsymbol{W}^{-1}(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{nom}})\|_2$. This solution can be obtained from [SWB07]:

$$\boldsymbol{u}_{\mathrm{cor}} = \boldsymbol{W}(\boldsymbol{A}\boldsymbol{W})^+\boldsymbol{b} + (\boldsymbol{I} - \boldsymbol{W}(\boldsymbol{A}\boldsymbol{W})^+\boldsymbol{A})\boldsymbol{u}_{\mathrm{nom}}$$

For later referencing, a corrective control signal for a line defined by α and d_0 will be denoted by $\boldsymbol{u}_{\mathrm{cor,line},\alpha,d_0}$.

2.3.3 Point Constraining

It may be desired to constrain our ZMP to a point defined by the coordinates x_c and y_c on the xy-plane of our base frame projected on to the ground. Obtaining the corrective control signal for this requires the solution of the following:

$$\boldsymbol{r}_{\text{zmp}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) = \begin{pmatrix} r_{\text{zmp},x}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \\ r_{\text{zmp},y}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \end{pmatrix}$$

and as was the case with line constraining, these equations are also linear in $\boldsymbol{u}$ and can be rewritten as:

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}$$

but when written with more detail:

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Here similar to line constraining a unique solution will only exist for $n = 2$, for cases where $n > 2$ we use the same techniques used to to obtain a unique solution by requiring the solution to have minimal l_2-norm $\|\boldsymbol{u}\|_2$ or $\|\boldsymbol{u} - \boldsymbol{u}_{\text{nom}}\|_2$ or even $\|\,\boldsymbol{W}^{-1}(\boldsymbol{u} - \boldsymbol{u}_{\text{nom}})\|_2$ as required.

For later referencing the a corrective control signal for a point defined by x_c and y_c will be denoted by $\boldsymbol{u}_{\text{cor,point},x_c,y_c}$.

In this chapter it was demonstrated how we can formulate a simple nominal controller as well as two different corrective controllers, one of which designed to restrict the ZMP to move on a certain line defined by the parameters α_i and $d_{0,i}$, the other designed to restrict the ZMP at a certain point with coordinates x_c and y_c. In the next chapter, it will be shown how invariance control combines the nominal controller and the two corrective controllers to achieve balance control for the legged robot.

Chapter 3

Invariance Controller

In controlling our legged robot using solely the nominal controller we risk an important condition required for the robot to remain balanced being violated, that condition is the remaining of the ZMP inside the safe area. Invariance control solves this by appropriate switching between using the nominal controller and corrective controllers which are responsible for constraining the motion or the position of the ZMP. Chapter 2 showed how we can formulate a simple nominal controller designed to track sinusoidal joint angle trajectories, and two different corrective controllers each designed for a different constraint of the ZMP's motion, being constraining the ZMP's motion (or position) to be on a line or on a point. In this chapter we demonstrate how we can formulate an invariance controller for a safe area bounded by a polygon with a nontrivial shape.

Assume we have a safe area bounded by a convex polygon formed of N line segments as shown in Fig. 3.1. Where each of the lines from l_1 to l_N is described by the parameters α and d_0, such that the i^{th} line is described by α_i and $d_{0,i}$.

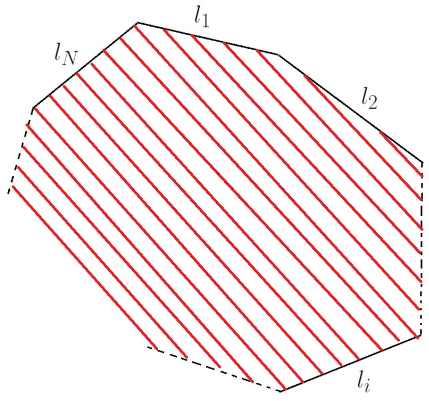


Figure 3.1: Example of a safe area bounded by a polygon.

For each i^{th} line we define a variable h_i as follows:

$$h_i = r_{\text{zmp}_{\alpha_i},x} - d_{0,i}$$

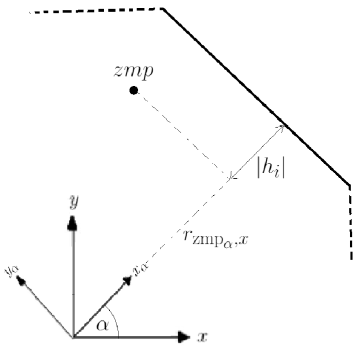


Figure 3.2: Interpretation of the quantity h_i.

The variable h_i is a measure of the perpendicular distance between the ZMP and the line l_i. The sign on h_i determines which side of the line l_i our ZMP lies on; a negative h_i implies that the ZMP is on the safe side of l_i as shown in Fig. 3.2 and a positive ZMP implies a position on the unsafe side of the line l_i and hence its position outside the safe area. So if we want the ZMP to remain inside the safe area then then $h_i < 0$ for all $1 \leqslant i \leqslant N$. So for a ZMP inside the safe area we need:

$$\max\{h_1, \cdots, h_N\} < 0$$

We define $h_{\text{cur},i}$ and $h_{\text{nom},i}$ for $1 \leqslant i \leqslant N$:

$$h_{\text{cur},i} = r_{\text{zmp}_{\alpha_i},x}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}_{\text{cur}}) - d_{0,i}$$

$$h_{\text{nom},i} = r_{\text{zmp}_{\alpha_i},x}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}_{\text{nom}}) - d_{0,i}$$

where $\boldsymbol{u}_{\text{cur}}$ is the output of the current controller, nominal or corrective; and $\boldsymbol{u}_{\text{nom}}$ is the output of the nominal controller. So $h_{\text{cur},i}$ is the h-value for the i^{th} line for the ZMP location implied by the current controller and $h_{\text{nom},i}$ is the h-value for the i^{th} line for the ZMP location implied by the nominal controller which may be fictive.

The idea that our invariance controller should be based upon is that if

$$\max\{h_{\text{nom},1}, \cdots, h_{\text{nom},N}\} < 0$$

then $\boldsymbol{u}_{\text{cur}} = \boldsymbol{u}_{\text{nom}}$; otherwise, we would need to take some corrective action to hold the ZMP at the boundary. One may initially suggest an simple algorithm as follows:

```
Data: $\{h_{\text{nom},1}, \cdots, h_{\text{nom},N}\}$ and $\{h_{\text{cur},1}, \cdots, h_{\text{cur},N}\}$
Result: $\boldsymbol{u}_{\text{cur}}$
if $\max\{h_{nom,1}, \cdots, h_{nom,N}\} < 0$ then
    $\boldsymbol{u}_{\text{cur}} = \boldsymbol{u}_{\text{nom}}$;
else
    $I = \max_i\{h_{\text{cur},1}, \cdots, h_{\text{cur},N}\}$ ;            // Index of the maximum element
    $\boldsymbol{u}_{\text{cur}} = \boldsymbol{u}_{\text{cor,line},\alpha_I,d_{0,I}}$
end
```

Algorithm 1: Basic control strategy for polygon boundary.

The problem with this; however, is that line corrective controllers restrict the ZMP's location on a line without constraint for *where* on the line. For example it may happen that the nominal controller would suggest a ZMP position that would violate line l_i, so according to the algorithm we would switch to the corrective controller for line l_i. However, the corrective controller for line l_i suggests a ZMP location that violates line l_{i+1} so we switch to the corrective controller for line l_{i+1}, but this controller violates l_i so we switch back and very rapid switching between the corrective controllers for l_i and l_{i+1} occurs, this is illustrated in Fig. 3.3. This leaves the ZMP outside the boundary for the whole time that $\max\{h_{\text{nom},1}, \cdots, h_{\text{nom},N}\} \geqslant 0$, needless to mention that this rapid switching implies rapid switching of actuator torques which can cause damage to the actuators.

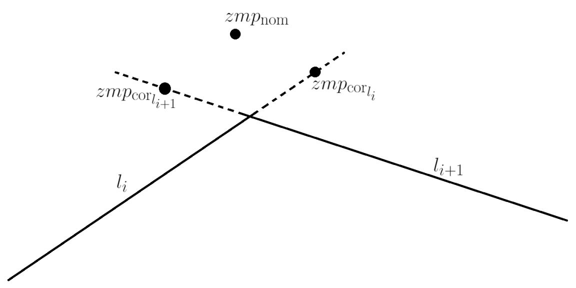


Figure 3.3: The problem associated with the basic control strategy.

A simple and effective way to get around this is to use point corrective constraining during this time to constrain the ZMP to the corner formed by the lines l_i and l_{i+1}. The algorithm for this is as follows:

```
Data: $\{h_{\text{nom},1}, \cdots, h_{\text{nom},N}\}$ and $\{h_{\text{cur},1}, \cdots, h_{\text{cur},N}\}$
Result: $\boldsymbol{u}_{\text{cur}}$
if $\max\{h_{nom,1}, \cdots, h_{nom,N}\} < 0$ then
    $\boldsymbol{u}_{\text{cur}} = \boldsymbol{u}_{\text{nom}}$;
else
    $I1 = \max_i\{h_{\text{cur},1}, \cdots, h_{\text{cur},N}\}$ ;                // Index of the maximum element
    $\boldsymbol{u}_{\text{cur1}} = \boldsymbol{u}_{\text{cor,line},\alpha_{I1},d_{0,I1}}$;

    calculate $\{h_{\text{cur1},1}, \cdots, h_{\text{cur1},N}\}$ associated with $\boldsymbol{u}_{\text{cur1}}$;

    if $\max\{h_{cur1,1}, \cdots, h_{cur1,N}\} < 0$ then
        $\boldsymbol{u}_{\text{cur}} = \boldsymbol{u}_{\text{cur1}}$;
    else
        $I2 = \max_i\{h_{\text{cur1},1}, \cdots, h_{\text{cur1},N}\}$ ;
        $\boldsymbol{u}_{\text{cur2}} = \boldsymbol{u}_{\text{cor,line},\alpha_{I2},d_{0,I2}}$;

        calculate $\{h_{\text{cur2},1}, \cdots, h_{\text{cur2},N}\}$ associated with $\boldsymbol{u}_{\text{cur2}}$;

        if $\max\{h_{cur2,1}, \cdots, h_{cur2,N}\} < 0$ then
            $\boldsymbol{u}_{\text{cur}} = \boldsymbol{u}_{\text{cur2}}$;
        else
            calculate the intersection point $(x_p, y_p)$ of $l_{I1}$ and $l_{I2}$ ;
            $\boldsymbol{u}_{\text{cur}} = \boldsymbol{u}_{\text{cor,point},x_p,y_p}$;
        end
    end
end
```

Algorithm 2: Improved control strategy for polygon boundary.

So it was shown how we could combine the nominal controller with the corrective controllers to form our invariance controller for a safe area formed by a convex nontrivial polygon boundary. This algorithm is simple and effective as will be shown in the next chapter discussing simulation results.

Chapter 4

Simulation Results

Chapter 3 showed us how we can formulate the invariance controller, intended to constrain our ZMP inside a boundary shaped by a convex nontrivial polygon to achieve balance for the legged robot. In this chapter we will demonstrate this controller for a simple 2 DOF legged robot and then for a more complicated 7 DOF legged robot. For the 2 DOF we demonstrate the difference between using the different solution options for $\boldsymbol{u}_{\text{cor}}$ as shown in Sec. 2.3 and then we discuss the limitations of invariance controllers briefly. Then we apply the invariance controller for the 7 DOF robot which resembles an actual legged robot to show its effectiveness.

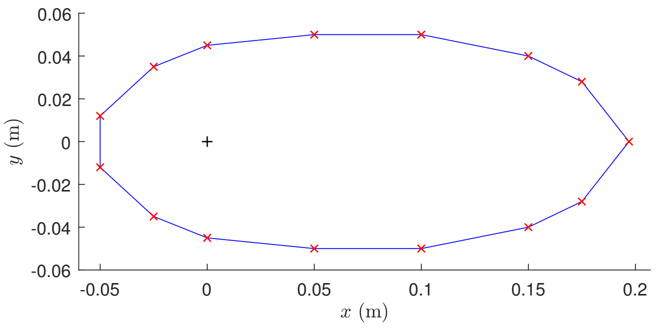


Figure 4.1: Polygon to describe the robot's foot base.

For both robot models the polygon in Fig. 4.1 will be used to describe the robot's foot base which is the same as the safe area because the robot is assumed to be in single support. The black + sign is where the ankle is above by h_{ankle} from the ground. Also, for the sake of simplicity any collision between a link and another link or a link and the ground is ingnored.

4.1 2 DOF Robot

The 2 DOF robot to be used is shown in Fig. 4.2. It consists of an inverted pendulum with two revolute joints at its base perpendicular to each other; one in the x-direction with a joint angle q_1 and another in the y-direction with a joint angle q_2.

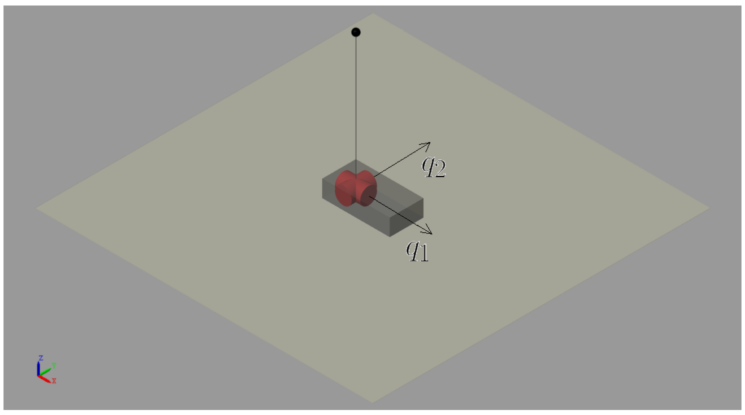


Figure 4.2: Two degree of freedom inverted pendulum robot.

Although technically possible to do by hand, the generation of the equations of motion and the controllers was done using the MATLAB® Symbolic Math Toolbox™ which provides a quick, correct and efficient method for such a task. This model is then simulated using MATLAB® along with Simscape™ within the Simulink® environment. This was done for both the 2 DOF model and the 7 DOF model. The model parameters used can be found in Sec. A.2.

4.1.1 Different Solution Options

As was shown in Sec. 2.3 it was possible to obtain three different solutions for $\boldsymbol{u}_{\text{cor}}$, each being based on obtaining a solution which minimizes the l_2-norm $\|\boldsymbol{u}\|_2$ or $\|\boldsymbol{u} - \boldsymbol{u}_{\text{nom}}\|_2$ or even $\|\boldsymbol{W}^{-1}(\boldsymbol{u} - \boldsymbol{u}_{\text{nom}})\|_2$ as required. In this section we explore the results of each solution and hence show how minimizing $\|\boldsymbol{W}^{-1}(\boldsymbol{u} - \boldsymbol{u}_{\text{nom}})\|_2$ is the most suitable of the three. First we choose our nominal controller to track the following joint angle trajectories:

$$\begin{pmatrix} q_{\text{des},1}(t) \\ q_{\text{des},2}(t) \end{pmatrix} = \begin{pmatrix} 0.50\sin((2)(2\pi)t)^\circ \\ 2.50\sin((1)(2\pi)t)^\circ \end{pmatrix}$$

using the PD controller with computed-torque control as in Sec. 2.2. The PD controller has the following parameters:

Parameter	Value
$\boldsymbol{K}_P$	$5\boldsymbol{I}$
$\boldsymbol{K}_D$	$2.5\boldsymbol{I}$

Table 4.1: PD controller parameters used for simulation.

and starting from rest with initial joint angles:

$$\begin{pmatrix} q_1(0) \\ q_2(0) \end{pmatrix} = \begin{pmatrix} 0^\circ \\ 0^\circ \end{pmatrix}$$

We use the invariance controller for the boundary shown in Fig. 4.1. We first simulate; however, without using the invariance controller and relying solely on the nominal controller alone. Figure. 4.3 shows the ZMP's motion visualized from roughly $t = 0.60$s till from $t = 1.00$s. As shown, the ZMP at some point exits the safe area, meaning that if we relied solely only on the nominal controller our robot will have started to loose balance from $t = 0.60$s. So the use of the invariance is essential for maintaining the balance of the robot.

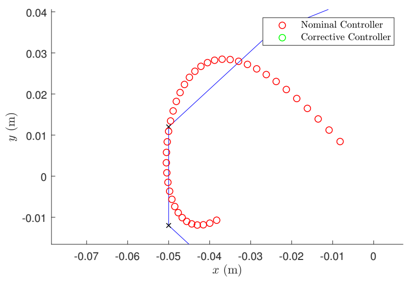


Figure 4.3: The ZMP violating the boundary without invariance control.

Invariance Control Using the First Solution Option

Using invariance control with the first solution option of $\boldsymbol{u}_{\text{cor}}$ which minimizes the l_2-norm $\|\boldsymbol{u}\|_2$ we simulate for the chosen nominal controller. Figure 4.4 shows the ZMP's motion visualized from roughly $t = 0.60$s till from $t = 1.00$s.

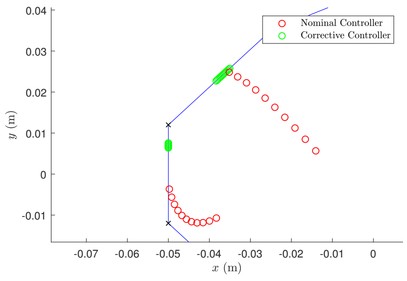
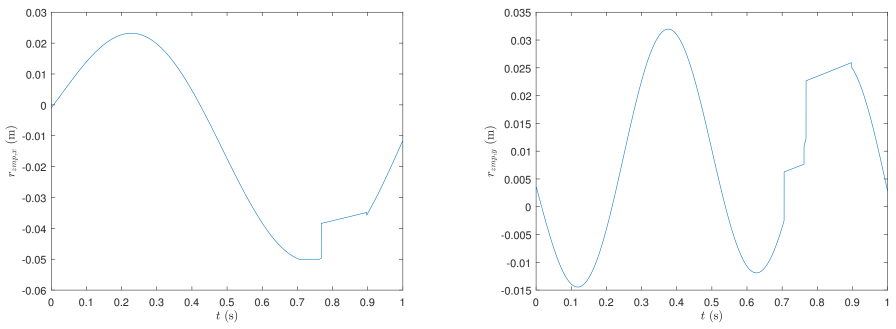


Figure 4.4: The ZMP being constrained at the boundary using the first solution option for $\boldsymbol{u}_{\text{cor}}$.

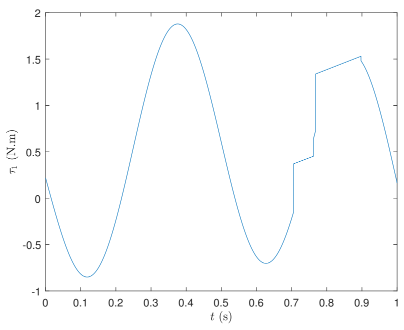
As seen, the path taken by the ZMP is not continuous as it jumps due to the switching of our controller from first the nominal controller to the corrective controller of the line which was about to be violated first then to the corrective controller of the adjacent line and then to the nominal controller again. This jumping of the ZMP is clearer in Fig. 4.5 where the ZMP changes position instantaneously at times.



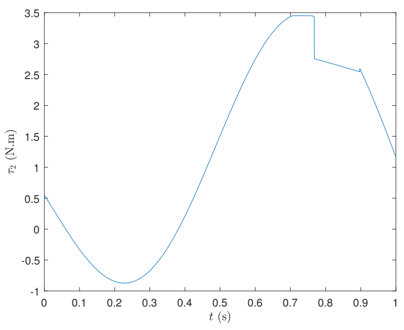
(a) $r_{\text{zmp},x}$ vs. t. (b) $r_{\text{zmp},y}$ vs. t.

Figure 4.5: The ZMP's sudden changes of position are clear in the plots of its x and y coordinates against time.

In theory this is not a problem since the ZMP remains constrained within the boundary as required for the robot's balance. The problem lies in practicality. As seen in Fig. 4.6, these instantaneous changes in the ZMP's position go back to instantaneous changes in the robot's actuator torques. In practical terms, there is always a limit to how rapidly an actuator can change its torque. Even then, it is usually recommended by the actuator's manufacturer to avoid such sudden changes in torque because they can have a damaging effect on the actuator along with its control circuit.



(a) τ_1 vs. t.



(b) τ_2 vs. t.

Figure 4.6: Sudden changes in the actuator torques resulting from the first solution option for $\boldsymbol{u}_{\text{cor}}$.

It is therefore not recommended to use the first solution option of $\boldsymbol{u}_{\text{cor}}$ due to lack of its practicality.

Invariance Control Using the Second Solution Option

Using the second solution option for $\boldsymbol{u}_{\text{cor}}$ instead allows for a minimization of the l_2-norm $\|\boldsymbol{u} - \boldsymbol{u}_{\text{nom}}\|_2$. This makes $\boldsymbol{u}_{\text{cor}}$ resemble $\boldsymbol{u}_{\text{nom}}$ as much as possible while maintaining the required constraints, so on switching controllers there is no sudden change of $\boldsymbol{u}$. This corresponds to continuos actuator torques which corresponds to a continuous path taken by the ZMP as shown in Fig. 4.7.

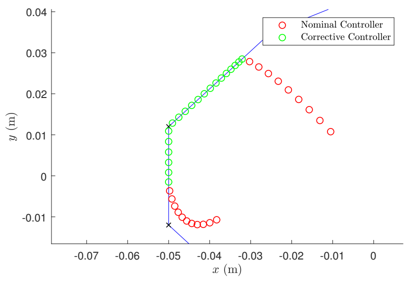
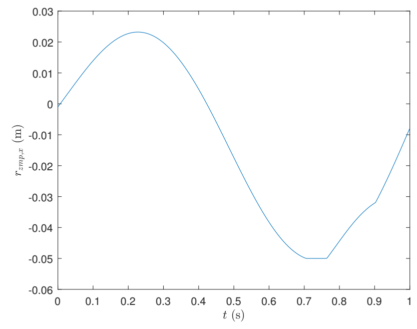
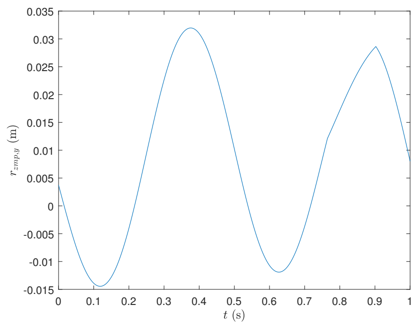


Figure 4.7: The continuos ZMP path resulting from using the second solution option.

Shown in Fig. 4.8 are the ZMP's coordinates plotted against time. As opposed to Fig. 4.5, no sudden changes in the ZMP's position are present. This corresponds to continuous actuator torques as shown in Fig. 4.9 which can be easily and safely fulfilled by the typical actuator.

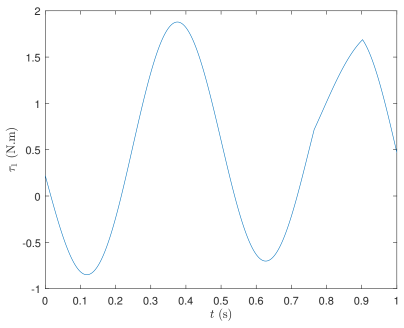


(a) $r_{\text{zmp},x}$ vs. t.

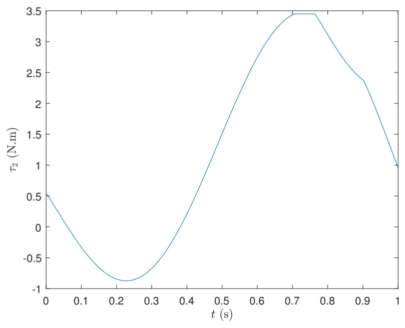


(b) $r_{\text{zmp},y}$ vs. t.

Figure 4.8: No sudden changes in the ZMP's x and y coordinates are present when using the second solution option for $\boldsymbol{u}_{\text{cor}}$.



(a) τ_1 vs. t.



(b) τ_2 vs. t.

Figure 4.9: No sudden changes in the actuator torques are present when using the second solution option for $\boldsymbol{u}_{\text{cor}}$.

So due to its practicality the second solution option is clearly better than the first solution option. We can still; however, go one more step further than this.

Invariance Control Using the Third Solution Option

When performing the corrective action required to hold the ZMP at the boundary it may be desired to have each joint participate in this corrective action to a certain extent. For example we may have a legged robot balancing on one foot while holding an object in its hands. If when balancing a corrective action has to be taken, we would not want the finger joints to participate in the corrective action as this can make them deviate enough from the nominal task (gripping the object) and cause the object to fall. We may want the elbow joint to participate in the corrective action, but not to a great extent as this can also cause the object to fall if deviant enough from the nominal task. The hip joints could be chosen to participate fully in the corrective action.

This can be achieved by assigning each i^{th} joint a weight $w_{i,i}$, such that if $w_{i,i}$ has a high value relative to the other joints' weights then the i^{th} joint will deviate more from the nominal task to contribute towards the corrective action, and for a low value of $w_{i,i}$ compared to the other joints' weights then the i^{th} joint will deviate less from the nominal task. To achieve this we can use the third solution option for $\boldsymbol{u}_{\text{cor}}$ which minimizes the l_2-norm $\| \boldsymbol{W}^{-1}(\boldsymbol{u} - \boldsymbol{u}_{\text{nom}})\|_2$. Taking the same case from Sec. 4.1.1 we can specify the weighing matrix $\boldsymbol{W}$ as:

$$\boldsymbol{W} = \begin{pmatrix} w_{1,1} & 0 \\ 0 & w_{2,2} \end{pmatrix}$$

We can increase $w_{2,2}$ relative to $w_{1,1}$ and see the effect of this as shown in Fig. 4.11, increasing the value of $w_{2,2}$ relative to $w_{1,1}$ decreases the contribution of τ_1 towards the corrective action as it makes it resemble the nominal controller torque $\tau_{\text{nom},1}$ more, and due to this τ_2 will have to compensate for the reduction of τ_1's corrective action and will have to deviate more from the nominal controller $\tau_{\text{nom},2}$ more to keep the ZMP at the boundary.

Also as shown in Fig. 4.10 the path taken by the ZMP is continuous and the boundary constraints are still satisfied while assigning different weights for the joints.

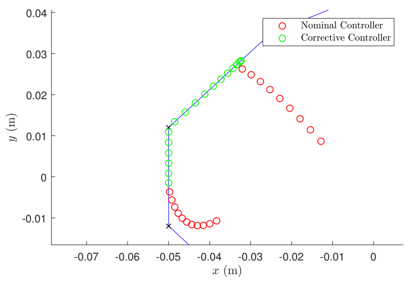
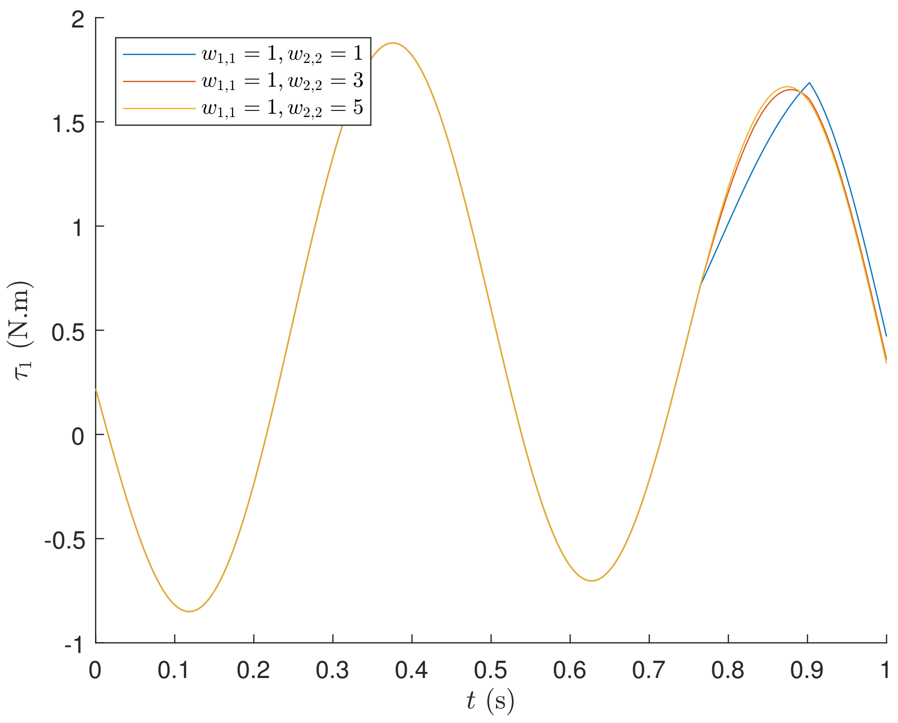
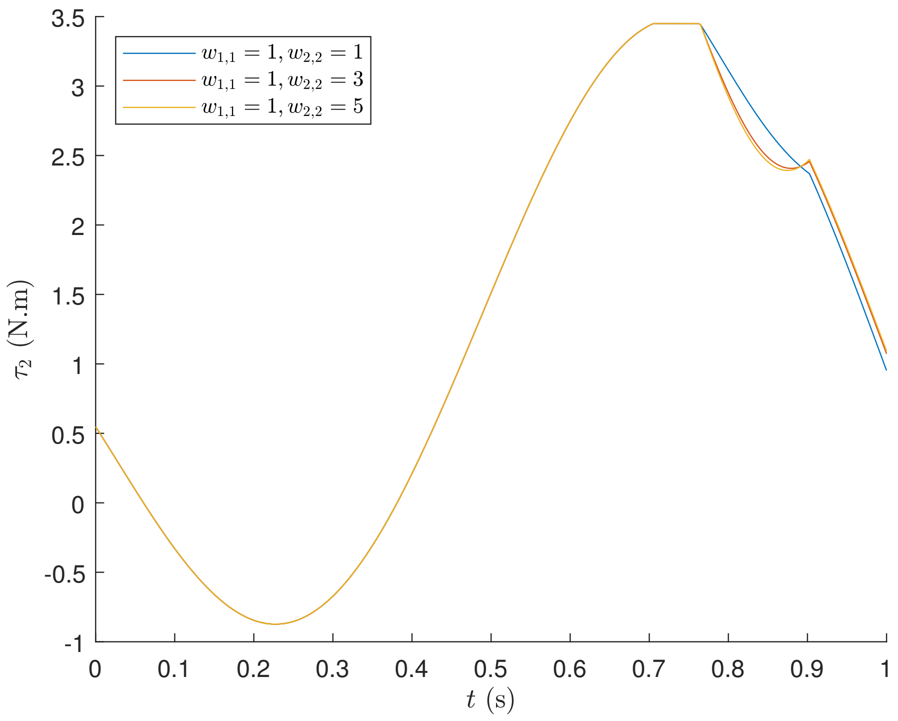


Figure 4.10: A continuos ZMP path also resulting from using the third solution option.

It is therefore clear that the third solution option is the most convenient of the three as not only can it be fulfilled by typical actuator in terms of torque, it can also give us the freedom of specifying how much each joint should contribute towards the corrective action taken to constrain the ZMP at the boundary. So we can choose an appropriate weights matrix $\boldsymbol{W}$ that results in a satisfactory corrective action as required. The third solution option will be used for all of the upcoming simulations.



(a) Plot of τ_1 vs. t.



(b) Plot of τ_2 vs. t.

Figure 4.11: The different robot actuator torques for different values of $w_{2,2}$. Note that for equal values of $w_{1,1}$ and $w_{2,2}$ the result is the same as the second solution option.

4.1.2 Limitations of the Invariance Controller

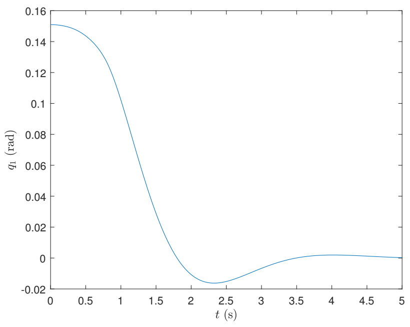
Although the upgrade from a sole acting nominal controller to an invariance controller does indeed increase the balancing capabilities of a legged robot, this increase in fact limited. In this section we will demonstrate an example of how much invariance control can increase the balance capabilities of a legged robot using the 2 DOF robot from the previous sections. Assume the robot has the same PD computed-torque controller with the parameters as in Tab. 4.1 and a joint weights matrix equal to the identity matrix for simplicity. However, this time our target is to reach and maintain an upright posture such that

$$\begin{pmatrix} q_{\text{des},1}(t) \\ q_{\text{des},2}(t) \end{pmatrix} = \begin{pmatrix} 0^\circ \\ 0^\circ \end{pmatrix}$$

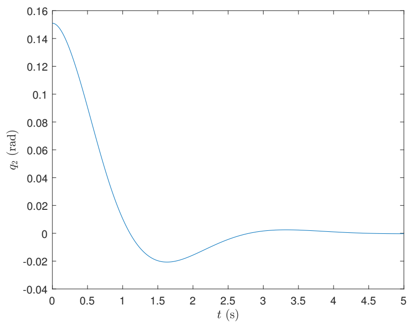
and start from rest at different starting positions defined by $\boldsymbol{q}(0)$. If we first try this with the starting position

$$\boldsymbol{q}(0) = \begin{pmatrix} 8.65^\circ \\ 8.65^\circ \end{pmatrix}$$

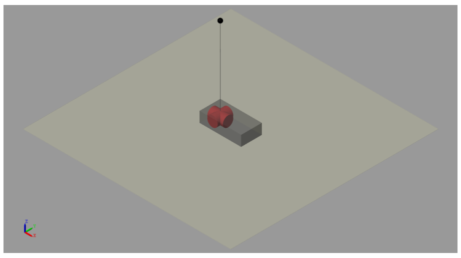
the robot does indeed reach the desired upright position after some settling time like expected as shown in Fig. 4.12.



(a) q_1 vs. t.



(b) q_2 vs. t.



(c) The final upright position reached by the robot as required.

Figure 4.12: The joint angles settle at the desired position as required.

This settling at the upright position even occurs in spite of the fact that at some point the corrective controller had to be used to prevent the ZMP from exiting the safe area as shown in Fig. 4.13.

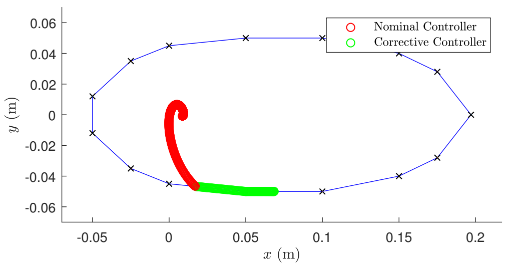
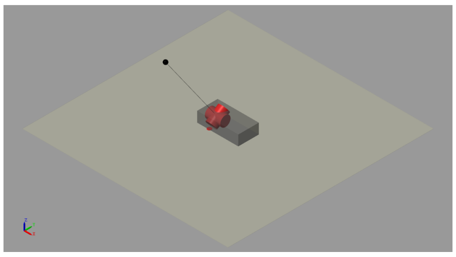


Figure 4.13: The settling at the upright position is achieved in spite of the fact that at some point the corrective controller was used.

However, if we try the same but with a starting position

$$\boldsymbol{q}(0) = \begin{pmatrix} 8.75^\circ \\ 8.75^\circ \end{pmatrix}$$

the robot collapses and reaches an unacceptable position as shown in Fig. 4.14


(a) Robot at $t = 1.35$s.

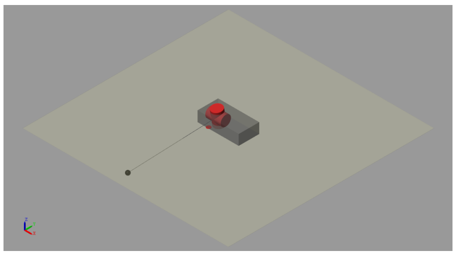

(b) Robot at $t = 1.60$s.

Figure 4.14: The robot collapses and doesn't reach the upright position.

and on examining the path taken by the ZMP we see that although an erratic path is taken, at no point does the ZMP exit the safe area.

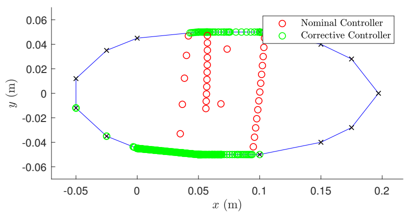
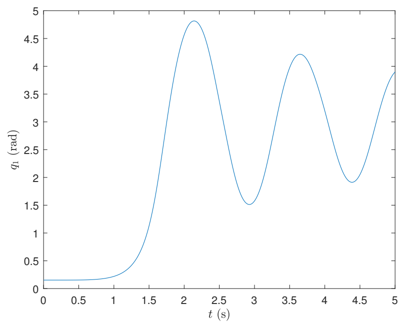
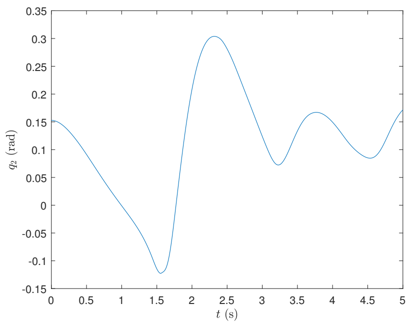


Figure 4.15: Erratic path taken by the ZMP without exiting the safe area.

Why did the robot collapse then when the ZMP never exited the safe area? This is because the remaining of the ZMP inside the safe area is not actually a complete determinant of the robot staying in an upright position as we want for any legged robot. Its remaining inside the safe area only guaranteed that the robot's foot will not rotate about one of its edges and cause the robot to topple over. However, although the robot's foot did not rotate about any of its edges, the corrective controller did cause q_1 to exceed $90°$ as shown in Fig. 4.16(a), and the robot did technically fall down in the end. The corrective controller has one and only one task and that is to constrain the ZMP on the boundary regardless of the consequent joint positions, velocities and accelerations.



(a) q_1 vs. t.



(b) q_2 vs. t.

Figure 4.16: The joint angles for a failed case of invariance control.

The problem with the corrective controller as pointed out by [SWB07] is that the necessary torques to keep the ZMP constrained at the boundary increase the distance between the ZMP and the robot's center of mass (unless the center of mass is at rest above the ZMP), so persistent constraining of the ZMP at the boundary will always eventually lead to the robot falling down (without the foot ever rotating about one of the edges). The success of invariance control relies on a brief utilization of the corrective controller to hold the ZMP at the boundary to prevent the foot from rotating about one of its edges, followed by a quick enough switch back to nominal control mode. If this switch back to nominal controller mode happens quickly enough then we get a successful case of invariance control as was shown in the first example of this section, otherwise we get a failed case of invariance control as in the second example. So it is clear that invariance control has limitations.

We could demonstrate such limitations by trying to do the same as in the first two examples but with different starting positions of q_1 and q_2. We could regard that a case is failed if at any time any of the joint angles exceed $90°$ in any direction. This is shown in Fig. 4.17 where starting positions which lead to failure are marked by a red cross, and those which require corrective action some point and do not fail are marked with a green circle and others which do not fail and do not require any corrective action are marked with a blue circle.

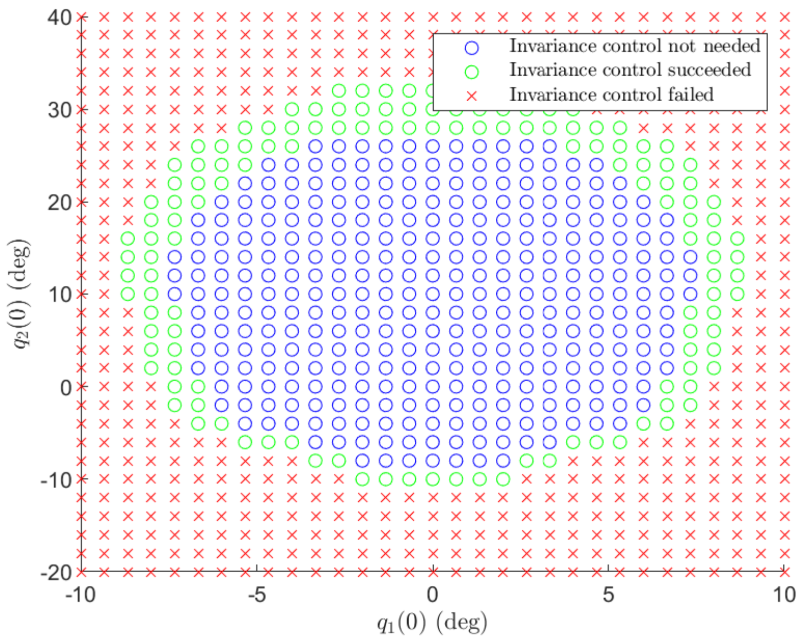


Figure 4.17: The simulation runs for different starting positions show the improvement of the robot's balancing capabilities brought by the invariance controller, but this improvement is limited.

In Fig. 4.17 the blue area represents the starting positions that did not require invariance control, so the robot could have stayed balanced using a sole acting nominal controller. The red area represents the starting positions that failed using the invariance controller and would have also failed with a sole acting nominal controller. It can be deduced then that the green area is a measure of the added balance capability by the invariance controller to the robot, as without the invariance controller the robot would have failed in this region too. The red area is where we can rely on invariance control no more and be forced to use another strategy to avoid loosing balance such as stepping control. It is worth mentioning that the reason for the symmetry of the plot about the $q_2(0)$ axis and the asymmetry about the $q_1(0)$ axis is due to the symmetry of the safe area boundary about the x-axis and its asymmetry about the y-axis respectively.

Although it was pointed out that the corrective controller can cause the robot to fail due to the consequent unbounded trajectories of the joint angles that will cause the robot to hit the ground, there are other ways in which it can cause the robot to fail. For a more realistic model we would have limitations for joint angle positions, velocities and accelerations; and also links that should not collide with each other and a ground that can not be penetrated (as was ignored in the simulation). It is necessary to understand the limitations of the invariance controller well as only this way can we be able to utilize it to its full potential without failure.

4.2 7 DOF Robot

The 7 DOF robot to be used is shown in Fig. 4.18. It consists of a legged robot which resembles a humanoid robot. The robot has the same foot base as that of the 2 DOF robot used in the previous section. The model parameters used for this robot can be found in Sec. A.3.

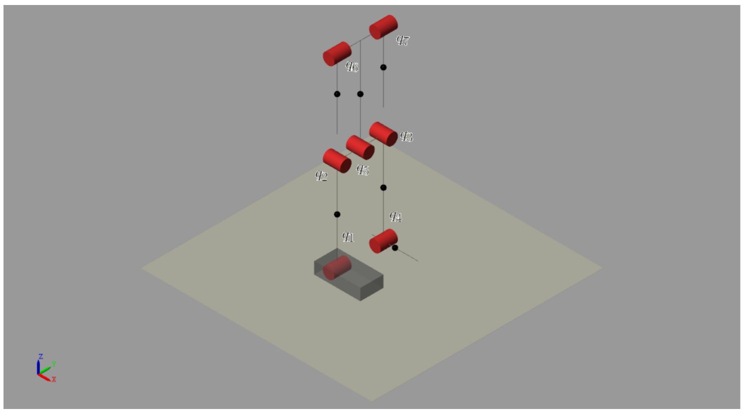


Figure 4.18: Seven degree of freedom legged robot.

Assume we want to assign this robot the nominal task of dancing while standing on one foot. For this we choose to have the joints track the following joint angle trajectories:

$$\begin{pmatrix} q_{\text{des},1}(t) \\ q_{\text{des},2}(t) \\ q_{\text{des},3}(t) \\ q_{\text{des},4}(t) \\ q_{\text{des},5}(t) \\ q_{\text{des},6}(t) \\ q_{\text{des},7}(t) \end{pmatrix} = \begin{pmatrix} (2.5\sin((1)(2\pi)t) + 5)^\circ \\ (10\sin((1)(2\pi)t) + 20)^\circ \\ (-10\sin((1)(2\pi)t) - 20)^\circ \\ 0^\circ \\ (-10\sin((1)(2\pi)t) + 20)^\circ \\ (20\sin((1)(2\pi)t) - 90)^\circ \\ (-20\sin((1)(2\pi)t) - 90)^\circ \end{pmatrix}$$

with the initial joint angle positions and velocities:

$$\boldsymbol{q}(0) = \begin{pmatrix} 5^\circ \\ 20^\circ \\ -20^\circ \\ 0^\circ \\ 20^\circ \\ -90^\circ \\ -90^\circ \end{pmatrix} \qquad \dot{\boldsymbol{q}}(0) = \begin{pmatrix} ((2.5)(2\pi))^\circ/s \\ ((10)(2\pi))^\circ/s \\ ((-10)(2\pi))^\circ/s \\ 0^\circ/s \\ ((-10)(2\pi))^\circ/s \\ ((20)(2\pi))^\circ/s \\ ((-20)(2\pi))^\circ/s \end{pmatrix}$$

The tracking will be done using a PD controller with computed-torque control having parameters as in Tab. 4.1.

Normally, this nominal task does not cause any ZMP boundary violations and results in ZMP motion consisting of oscillation along the arc shown in Fig. 4.19.

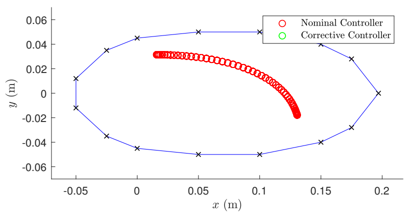


Figure 4.19: .

However, if the robot is exposed to external disturbances, this motion pattern of the ZMP will change and is likely to exit the safe area and cause the robot to loose balance without corrective action. In the next section we will demonstrate how the invariance controller can help the robot recover from external disturbances.

4.2.1 Disturbance Rejection Demonstration

During the robot's performance of the nominal task we expose it to two different impulsive forces at two different times at the mid-torso as shown in Fig. 4.20, one in the x direction with impulse J_x and the other in the y direction with impulse J_y.

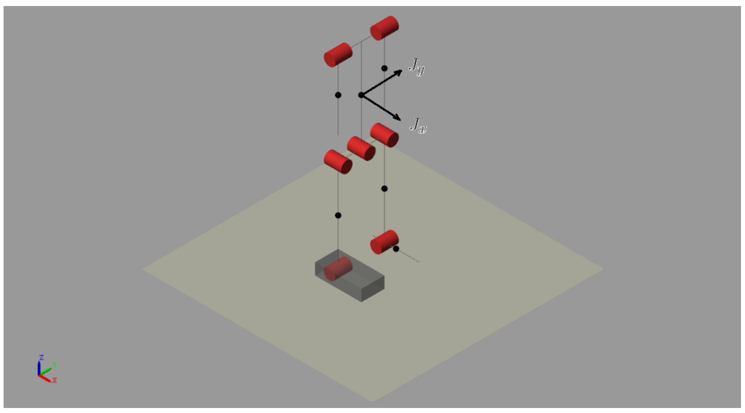


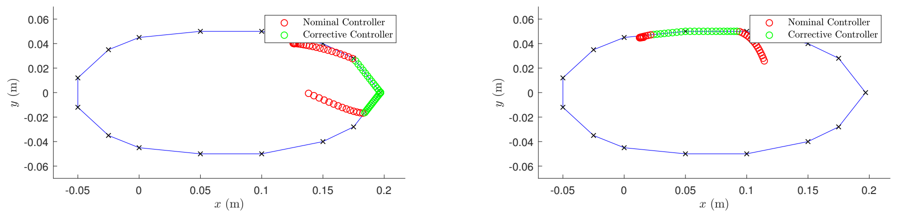
Figure 4.20: The impulses exerted at the robot's mid-torso.

The invariance controller is used with the third solution option for $\boldsymbol{u}_{\text{cor}}$ with a weights matrix $\boldsymbol{W}$ chosen to be the diagonal matrix:

$$\boldsymbol{W} = \text{diag}(5, 1, 1, 5, 15, 10, 10)$$

Successful Case

For this example we choose J_x to be 2.6Ns acting at $t = 1$s and J_y to be 2.3Ns acting at $t = 3$s. Both of the impulses result in two occasions when the ZMP is about to exit the safe area and gets constrained at the boundary by the corrective controller as shown in Fig. 4.21



(a) From $t = 1.1$s to $t = 1.8$s. (b) From $t = 3.3$s to $t = 3.8$s.

Figure 4.21: Two occasions where the ZMP is about to exit the safe area.

The success of the invariance controller in this example is confirmed by the disappearance of the joint angle tracking errors after both hits as shown in Fig. 4.22.

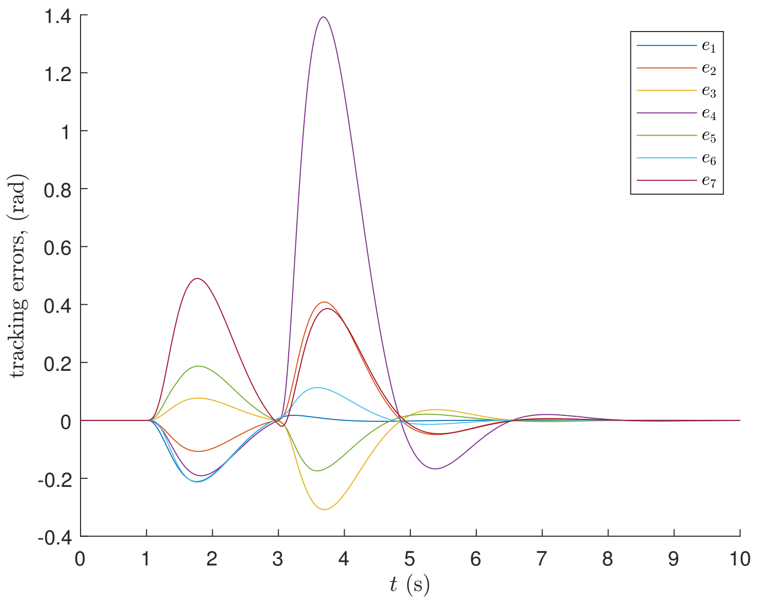


Figure 4.22: Disappearance of the joint agle tracking errors after both hits.

Failed Case

If we choose instead to give the robot only one rather strong push characterized by the impulse J_x with magnitude 3.0Ns acting at $t = 1$s we get a failed case of invariance control as shown in Fig. 4.23 where the robot collapses and hits the ground at $t = 2.7$s.

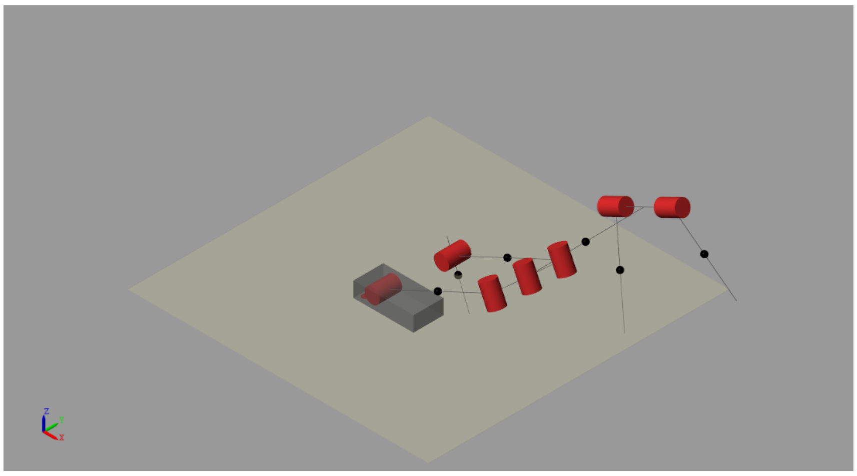


Figure 4.23: The robot collapses and hits the ground at $t = 2.7$s.

This is in spite of the fact that the ZMP never exits the safe area as shown in Fig. 4.24.

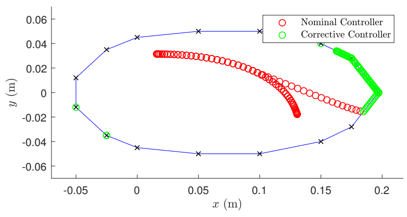


Figure 4.24: The robot hits the ground even though the ZMP never exits the safe area.

As discussed in Sec. 4.1.2 the remaining of the ZMP inside the safe area is not a complete determinant of the robot staying balanced, it is only one requirement. The true determinant of the failure of invariance control in this case is the robot configuration which has links colliding with the ground at $t = 2.7$s.

The unbounded tracking errors that result after the hit as shown in Fig. 4.25 also confirm that invariance control has failed in this case.

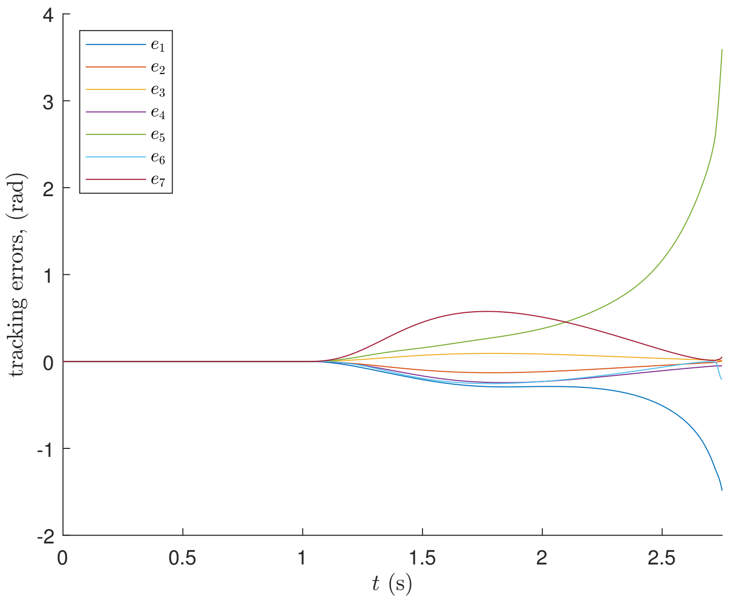


Figure 4.25: Unbounded tracking errors that result after the hit at $t = 1$s.

So it was shown how in some cases invariance control can be successful and help prevent the robot from loosing balance but in others it is not enough and the robot falls anyway. With a good understanding of the limitations of invariance control we can use stepping control in the cases where invariance control is to fail to effectively prevent all cases of failure.

Chapter 5

Conclusion

In this thesis we first introduced the Zero Moment Point as the point on the ground where the ground reaction must act to balance a legged robot. A legged robot will stay balanced only if the ZMP remains inside the convex hull of the contact points between the robot's feet and the ground, which we termed in this thesis the safe area. Our main problem was that the nominal controller which is responsible for the execution of the robot's main task may cause the ZMP to exit the safe area, and hence cause the robot to loose balance. Therefore we needed some ZMP manipulative control law that would always restrict the ZMP inside the safe area to keep the robot balanced.

An intuitive and simple idea to restrict the ZMP inside the safe area is to switch from the nominal controller to a corrective controller whenever the nominal controller is about to cause a violation of the ZMP's constraint. The corrective controller's task is then to restrict the ZMP to move on the boundary of the safe area. As soon as the nominal controller is about to result in a ZMP which does not violate the constraint we switch back to it. We assumed to have a safe area boundary formed of a convex polygon with any number of sides. Constraining the ZMP to move on such a boundary requires at times to constrain it to move on the sides of the polygon boundary and at times to constrain its position to one of the polygon's corners. To achieve this two different corrective controllers were formulated; the line constraining corrective controller which constrains the ZMP to move on a line described by the two parameters α and d_0 and the point constraining corrective controller which can constrain the ZMP's position to a point with coordinates (x_c, y_c). Then using a simple algorithm that switches between the nominal, line constraining corrective and point constraining corrective controllers we are able to constrain the ZMP inside the safe area as required, this switching strategy forms the invariance controller.

First, we simulated the use of the invariance controller for a 2 DOF robot to demonstrate the differences between the three different solution options we had for each of the corrective controllers where each solution was based minimizing the l_2-norm

$\|\boldsymbol{u}\|_2$ or $\|\boldsymbol{u}-\boldsymbol{u}_{\text{nom}}\|_2$ or even $\|\boldsymbol{W}^{-1}(\boldsymbol{u}-\boldsymbol{u}_{\text{nom}})\|_2$ and in the same time resulting in the required corrective action. It was then shown that the third solution was the most convenient of the three as it offered practicality, resemblance to the nominal controller as well as the freedom of choosing how much each of the robot's joints should participate in the corrective action. Then an example was shown to demonstrate both how the invariance controller can add balance capability to the robot but in the same time this added balance capability is limited because of the instabilities that can occur due to the corrective controller. Then for a 7 degree of freedom legged robot we showed how invariance control can help prevent a legged robot from loosing balance due to unexpected pushes, but also how it can fail to do so if such pushes are strong enough. All in all it was shown that invariance control is effective and does indeed improve a robot's balance capabilities.

The main limitation for this thesis' work was the fact that the robot's links were modelled as point masses without rotational inertia as a way of simplifying simulation. This of course is not applicable for large robots; and cases where the angular velocities and accelerations are high such as running. Also, it was assumed that the controller has direct control over the actuators torques which is not usually the case. A more realistic model would be one where the controller has direct control over the actuator input currents instead. Another assumption used was that there is zero friction in the joints. Although usually negligible, frictional forces can sometimes have influence over the robot's performance.

It is recommended for future work to apply and simulate invariance control for a robot with much more degrees of freedom such as the atlas robot. One can also explore the influence of parameters such as $\boldsymbol{K}_p$, $\boldsymbol{K}_d$ and $\boldsymbol{W}$ on the effectiveness of the invariance controller and maybe find parameter values which give optimal added balance capability. Developing an understanding of the limitations of invariance control is recommended so it can be determined when invariacne control will fail and another strategy to maintain balance should be used, because only this way can we utilize invariance control to its maximum potential.

Appendix A

Robot Parameters

For all simulations the gravity vector was chosen to be $\boldsymbol{g} = (0, 0, -9.80665)^{\mathrm{T}}\mathrm{m/s}^2$

A.1 Safe Area Boundary

The coordinates of the points which describe the boundary of the safe area shown in Fig. 4.1:

x(m)	-0.05	-0.025	0	0.05	0.1	0.15	0.175	0.197
y(m)	0.012	0.035	0.045	0.05	0.05	0.04	0.028	0
x(m)	0.175	0.15	0.1	0.05	0	-0.025	-0.05	-0.05
y(m)	-0.028	-0.04	-0.05	-0.05	-0.045	-0.035	-0.012	0.012

Table A.1: Coordinates of the points which describe the safe area boundary.

The i^{th} line is formed by joining the points (x_i, y_i) and (x_{i+1}, y_{i+1}). If this line has equation $y = mx + c$ then we can find the parameters α_i and $d_{0,i}$ using:

$$\alpha_i = \tan^{-1}(m) + \pi/2$$
$$d_{0,i} = c\cos(\tan^{-1}(m))$$

A.2 2 DOF Robot

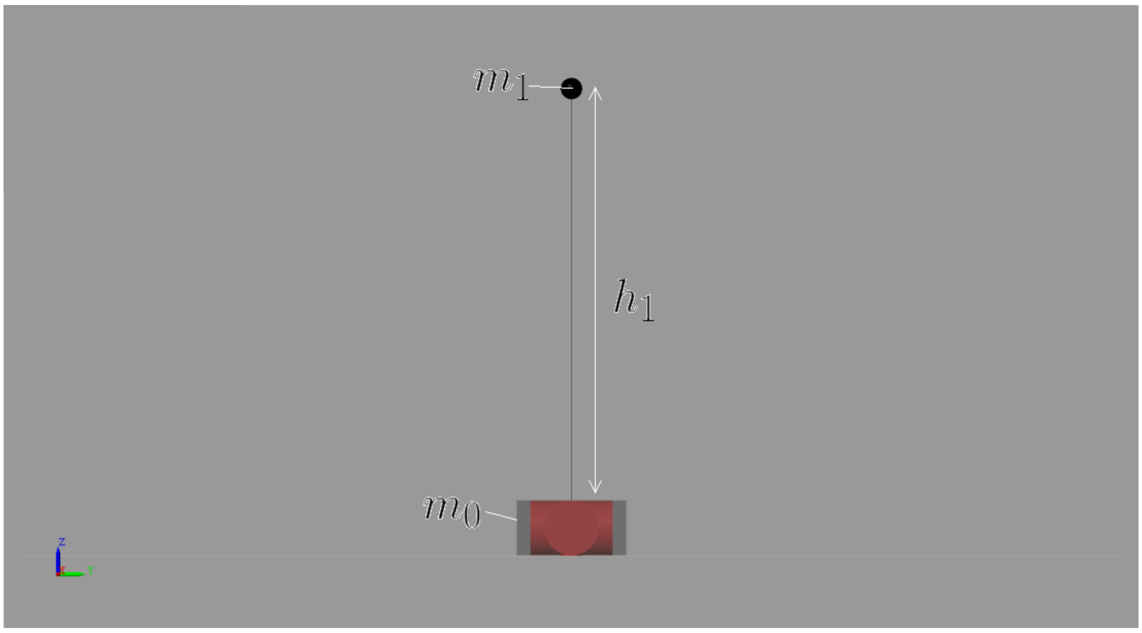


Figure A.1: The 2 DOF robot.

Parameter	Value	Unit
m_0	1	Kg
m_1	5	Kg
h_1	0.4	m
h_{ankle}	0.025	m

Table A.2: Model parameters for the 2 DOF robot.

A.3 7 DOF Robot

A.3.1 Inertial Parameters

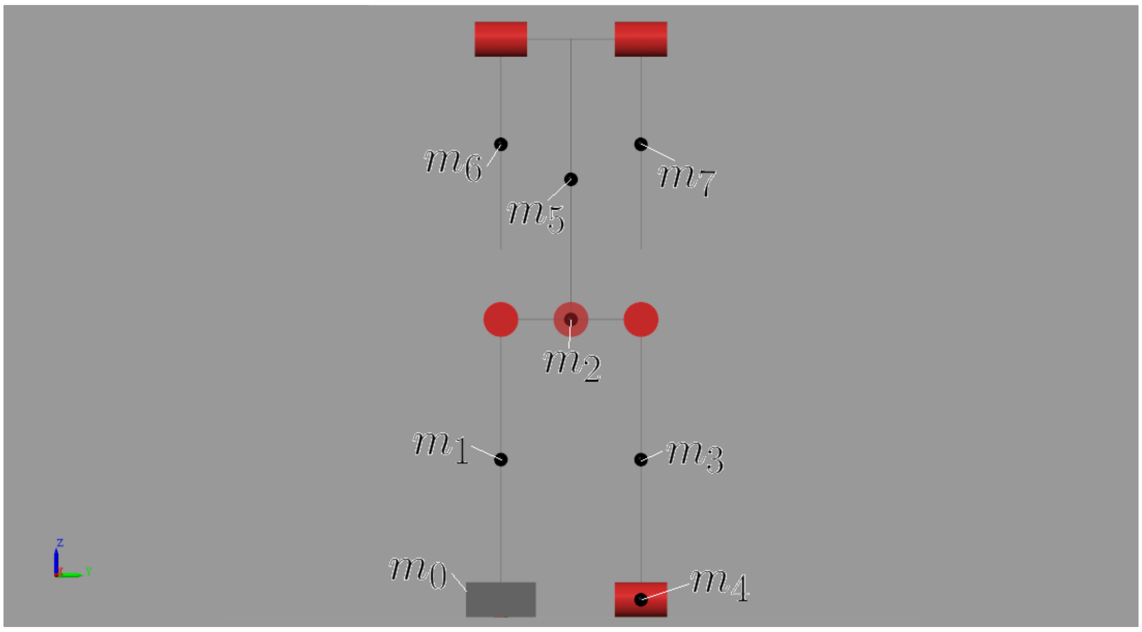


Figure A.2: Inertial parameters.

Parameter	Value	Unit
m_0	1	Kg
m_1	1	Kg
m_2	1	Kg
m_3	1	Kg
m_4	1	Kg
m_5	1	Kg
m_6	1	Kg
m_7	1	Kg

Table A.3: Inertial parameters for the 7 DOF robot.

A.3.2 Dimensional Parameters

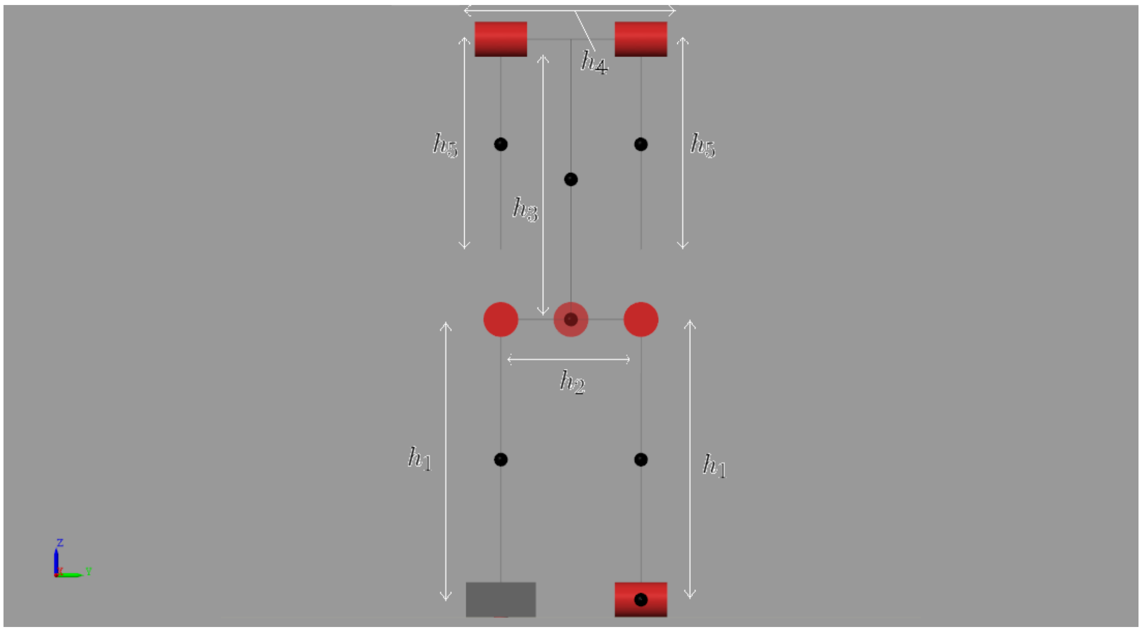


Figure A.3: Dimensional parameters.

Parameter	Value	Unit
h_1	0.4	m
h_2	0.2	m
h_3	0.4	m
h_4	0.2	m
h_5	0.3	m
h_{ankle}	0.025	m

Table A.4: Dimensional parameters for the 7 DOF robot.

List of Figures

Bibliography

[atl] Atlas robot official page. `https://www.bostondynamics.com/atlas`. Accessed: 2018-07-30.

[BJ90] F. Beer and E. R. Johnston. *Vector Mechanics for Engineers: Dynamics.* McGraw Hill, 1990.

[con] Convex hull explained. `https://en.wikipedia.org/wiki/Convex_hull`. Accessed: 2018-07-25.

[EB16] S. El-Baklish. Stability bounds in constrained control of legged robots. *Bachelor Thesis at The Technical University Of Munich*, 2016.

[EOO$^+$02] K. Erbatur, A. Okazaki, K. Obiya, T. Takahashi, and A. Kawamura. A study on the zero moment point measurement for biped walking robots. In *7th International Workshop on Advanced Motion Control. Proceedings (Cat. No.02TH8623)*, pages 431–436, July 2002. `doi:10.1109/AMC.2002.1026959`.

[HNI01] Qiang Huang, Y. Nakamura, and T. Inamura. Humanoids walk with feedforward dynamic pattern and feedback sensory reflection. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 4, pages 4220–4225 vol.4, May 2001. `doi:10.1109/ROBOT.2001.933277`.

[KH03] K. Kondak and G. Hommel. Control algorithm for stable walking of biped robots. *Proceedings of the International Conference on Climbing and Walking Robots (CLAWAR)*, pages 119–126, 2003.

[Kha02] H. K. Khalil. *Nonlinear Systems.* Pearson, 2002.

[PCDG06] J. Pratt, J. Carff, S. Drakunov, and A. Goswami. Capture point: A step toward humanoid push recovery. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 200–207, Dec 2006. `doi:10.1109/ICHR.2006.321385`.

[rob] Example of a simple robot. `http://www.alsrobot.com/index.php?route=product/product&product_id=516`. Accessed: 2018-07-31.

[SK08] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer, 2008.

[SSVO10] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 2010.

[SWB07] M. Sobotka, J. Wolff, and M. Buss. Invariance controlled balance of legged robots. In *2007 European Control Conference (ECC)*, pages 3179–3186, July 2007. `doi:10.23919/ECC.2007.7068567`.

[VBv01] M. Vukobratović, B. Borovac, and D. Šurdilović. Zero-moment point - proper interpretation and new applications. *Proceedings of the IEEE/RAS International Conference on Humanoid Robots (HUMANOIDS)*, pages 237–244, 2001.

License