MASTER THESIS

DESIGN OF A MOMENTUM-BASED OPTIMAL CONTROLLER FOR A LOWER LIMB HUMANOID

Joep T.J. van de Rijt
s1703528

FACULTY OF ENGINEERING TECHNOLOGY
DEPARTMENT OF BIOMECHANICAL ENGINEERING

EXAMINATION COMMITTEE
dr. Edwin H.F. van Asseldonk
dr.ir. Arvid Q.L. Keemink
dr.ir. Wouter B.J. Hakvoort
Ander Vallinas Prieto MSc

DOCUMENT NUMBER
BE - 851

UNIVERSITY OF TWENTE.

MAART 2022

Abstract

Between 250.000 and 500.000 suffer a spinal cord injury (SCI) per year worldwide. Demographics show that young people are among the groups most at risk. Trials suggest that early treatment may improve neurological recovery, however, full recovery is often not possible. In the majority of cases, the resulting immobility will lead to a significant reduction in quality of life. Exoskeletons could be used to combat immobility and improve quality of life. The Biomechanical Engineering group at the University of Twente has been developing a lower limp exoskeleton designed to enable individuals with incomplete or complete SCI. While the exoskeleton is mechanically functioning and a trajectory generator is being developed, a controller that adequately executes the trajectory, while maintaining balance is still absent. This thesis explores the option of using a Momentum Based Controller.

The problem is approached by a two-dimensional lower limb humanoid model with point feet. For this model, a simple trajectory generator is created that translates a CoM trajectory into momentum rate and feet acceleration trajectories. A quadratic program is used to reconcile the trajectory goals with the dynamics and find the optimal joint torque while minimizing the energy expenditure. Using this system, the control parameters are analyzed during two situations: walking at a constant velocity and recovering for a push. The results show that all tracking goals can be used as decision variables and none have to be fully constrained. For the cost, a high priority should be given to feet and chest angle tracking. If these are adequate, the momentum tracking can be done with a lower cost and balanced with operational costs.

While the model used is relatively simple compared to the exoskeleton, the recommendations made can still be applied to more complex situations and give useful insight into the control priorities of human-like walking.

Contents

List of acronyms

Abbreviation	Name	Description
BoS	Base of Support	The convex hull around all contact point on the supporting surface.
CMM	Centroidal momentum matrix	the matrix that describes the linear relationship between joint. velocities and momentum.
CoM	Center of Mass	The location of the system's effective total mass determined by the average position of all parts of the system.
CoP	Center of Pressure	The net reaction force of surfaces on an object.
CWC	Contact wrench cone	The cone of all possible wrenches that could be applied to a surface without slipping.
EoM	Equation of motion	The equation that captures the full dynamics of a system based on the configuration and the forces applied to the system.
FWP	Feasible wrench polytope	A convex shape that include all possible wrenches that can be executed by a system.
ICP	Instantaneous capture point	Point on the ground that enables the system to come to a stop when the CoP is placed there instantly.
LIP	Linear Inverterted Pendulum	Model that consists of a point mass at set height and a massless rod in contact with the ground.
MBC	Momentum-based control	A control method that focuses on controller the momentum of the system.
MRP	Motion reference planner	A subsystem of the high-level controller that translates the reference into tracking goals
RG	Reference Generator	A subsystem of the high-level controller that generates reference trajectories.
SCI	Spinal cord injury	damage to any part of the spinal cord or nerves at the end of the spinal canal.
XcoM	Extrapolated center of mass	Projection of the CoM on the supporting surface that includes a velocity factor. A CoP generated at the XcoM will cause a LIP to come to a standstill above the XcoM.
ZMP	Zero moment Point	Ghe point where the effects of all forces acting on the device can be replaced by a single point

Table 1: Caption

Chapter 1

Introduction

1.1 Problem statement

According to the World Health Organization, worldwide between 250.000 and 500.000 people suffer from a spinal cord injury (SCI) each year. The demographics show that young male adults (ages 20-29) are most at risk of suffering an SCI, followed by older males (age 70+) and young females (ages 15-19) [1]. Although preclinical studies and small trials suggest that early treatment may improve neurological recovery of sci patients, full recovery is often not possible [2]. This results in the majority of patients dealing with devastating long-term effects, such as loss of sensory and motor function and an increased likelihood of premature death [1].In the majority of cases, the resulting immobility will lead to a significant reduction in quality of life [3].

Exoskeletons could be used to combat immobility and help improve quality of life [4]. Exoskeletons have been in development since the 1960s when the US military started working on exoskeletons to enhance the performance of soldiers [5]. While early development was slow, a lot of progress has been made in the last decades. This has led to a great number of new exoskeletons [6]. These exoskeletons can be divided into three categories: human power augmentation, haptic interactions, and rehabilitation [5, 7].

For an overview of the state of the art of exoskeletons, Aliman et al. [8] gives a broad overview of the current exoskeletons on the market and in development. Anam et al. [5] and Young et al. [7] give overviews on the most used control systems for exoskeletons. Jatsun et al. [9] gives an overview of the current industrial exoskeletons.

The biomechanical engineering group of the University of Twente is developing an 18 degrees of freedom lower limb exoskeleton called the Symbitron Exoskeleton [6]. The exoskeleton has 8 actuated degrees of freedom, creating an under-actuated system. A trajectory generator is in development that will be capable of providing a motion trajectory that can be executed without the need for crutches. Multiple self-balancing controller are being considered.

In this thesis, the possibility of using a Momentum Based Controller (MBC) is explored. Furthermore, the control parameters are analyzed to give more insight into the influence of individual parameters on the performance.

1.2 Goals

This thesis has the following goals:

- Design a simplified model of the exoskeleton problem
- Design a quadratic program controller capable of walking and rejecting disturbance
- Analyse the influence and importance of parameters

- Give a recommendation on parameter prioritization

1.3 Problem approach

The exoskeleton is approached by a 2D lower limb humanoid. For this model, an MBC has been developed that is capable of following a given center of mass (CoM) trajectory and rejecting disturbances up to its total weight. The controller is built up of a high-level and mid-level controller. The high-level controller translates the CoM trajectory into momentum rate goals and generates the feet trajectories required to maintain balanced. The mid-level consists of a quadratic program that finds the optimal joint torques that execute the generated trajectories while minimizing the energy expenditure.

Using this model, the control parameters are experimentally analyzed. From the results of the parameter analysis, a final controller is designed. This controller is then used to discuss the feasibility of using an MBC as a controller for an exoskeleton and the recommended parameter prioritization.

1.4 Outline of this thesis

Chapter 2 will cover the background information. This includes background information on momentum-based control, the Symbitron exoskeleton for which the controller will be explored, a brief description of the most used high-level controllers, and a detailed description of the state of the art of optimal mid-level controllers. Chapter 3 covers the dynamic model used to explore the controller, including the contact model. Chapter 4 will go into the design of the controller. In chapter 5 the analysis method is explained and the results are shown and briefly discussed in chapter 6. In chapter 7 the final results of the analysis are discussed as well as the limitations of the design and analysis and applicability to the exoskeleton. Finally, chapter 8 will cover the recommendations for the control parameters as well as recommendations for the focus of further research.

Chapter 2

Background

This chapter will cover the background knowledge of controllers for humanoid walking systems. While exoskeletons and (lower limb) humanoids are different systems, in this chapter they are assumed to be similar when it comes to gait control. Both fields are developing rapidly and the strategies employed are often the same. For this reason, knowledge from both fields is integrated into this chapter.

Section 2.1 explains how free moving bodies are modelled. Section 2.2 covers the fundamentals of Momentum Based Control. Section 2.3 explains the exoskeleton developed by the Biomechanical Engineering department. Section 2.4 covers an overview of various control approaches. Sections 2.5, 2.6 and 2.7 cover the controllers used as high-level, mid-level and low-level controllers.

2.1 Floating Base Model

Considering both humans and walking robots can move freely in space, the motions of the system cannot be described by only joint positions, unlike a system that is fixed to the world. As such, a method is needed to model the position of the object in space.

The position of a body in space can be modeled by creating virtual joints between the world and the body, illustrated in figure 2.1. These joints represent the degrees of freedom of the body with respect to the world. For a 2D model, there are three virtual joints, two translations in the x and y direction, and one rotational around the z-axis. For a three 3D model, there are six virtual joints, three translation joints, and three rotation joints around the translation axis [10, 11].

Because these are virtual joints, they cannot be directly controlled, i.e. no force can be applied on these joints directly. The motion of these joints is determined by the motion of the joints of the body and external forces applied to the body.

2.2 Momentum Based Control

The control of free moving robotic systems is generally realized by controlling the position of the center of mass (CoM) relative to the world. The CoM is the location of the system's effective total mass determined by the average position of all parts of the system, weighted by their masses [12]. Momentum-based control (MBC) aims to control the CoM by manipulating its momentum.

Full dynamic models of a multi-body system are often complex. MBC simplifies the dynamics by looking only at the dynamics projected at the CoM, instead of the whole system. These dynamics are called the centroidal dynamics [12, 13].

Each free-moving system has the same dynamical behavior as a certain solid shape, depending on the configuration. Figures 2.2a and 2.2b illustrate this shape for a humanoid. This means that

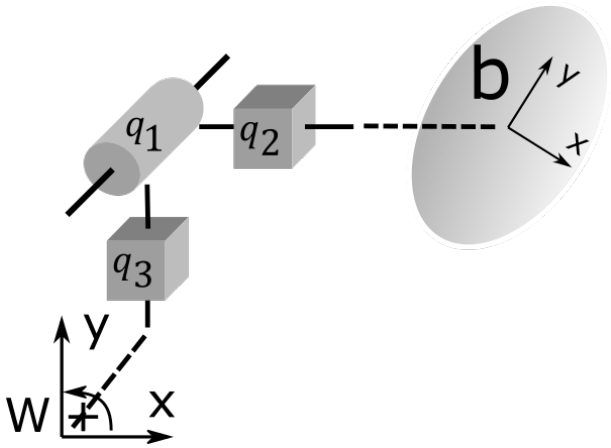
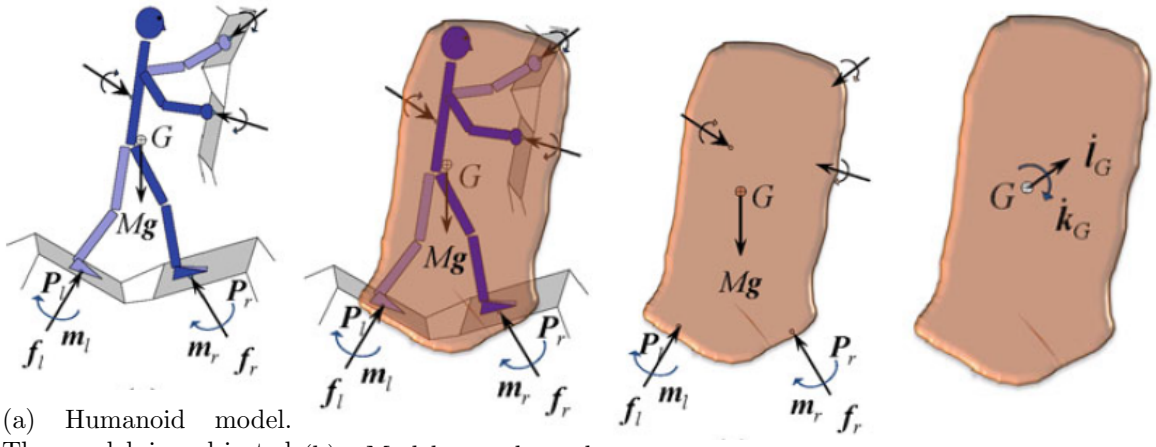


Figure 2.1: The 2D floating base model. Virtual joints are created between the world coordinates, W, and the object,b. The blocks represent linear joints and the cylinder a rotational joint.



(a) Humanoid model. The model is subjected to both joint torques and external forces in the form of gravity, GRF and interaction forces.

(b) Model overlapped by the centroidal shape. Only subjected to external forces.

(c) Only the centroidal form, subjected to external forces.

(d) Centroid form with resultant momentum from external forces.

Figure 2.2: Visual breakdown of centroidal dynamics made by Orin et al.[12].

the system, irrespective of complexity, will have an equal reaction to external force as this solid shape, centered around the CoM [12, 14]. This property can be used to calculate what forces need to be applied to a system to create the desired linear and rotational accelerations [12, 13].

The momentum equation for a multi body system is given by

$$h = A\dot{q}. \tag{2.1}$$

The momentum is equal to the centroidal momentum matrix (CMM), A, multiplied by the joint velocities $\dot{q}$. The CMM thus gives the linear relationship between the momentum and the joint velocities [14].

Change in momentum can be calculated by taking the time derivative of equation 2.1, resulting in:

$$\dot{h} = \dot{A}\dot{q} + A\ddot{q}. \tag{2.2}$$

Since, by Newton's law of motion, a change in momentum is equal to the forces applied to the system, the same equation can be used to determine what forces should be applied to

generate the desired momentum change. This gives the full equation

$$\dot{h} = \dot{A}\dot{q} + A\ddot{q} = \sum W, \tag{2.3}$$

where W are the wrenches, the effects of forces on a certain point, applied to the CoM.

MBC uses the relation between momentum rate, force and CoM acceleration to generate the desired movement [14]. The main benefit of this approach is that the reduction of the dynamical behavior of a complex system to the centroidal dynamics gives a direct relationship between the movement of the system and the forces applied anywhere on the system, as illustrated by figures 2.2c and 2.2d. Furthermore, it reduces the number of equations to be solved from all the degrees of freedom of the system to just the degrees of freedom of the floating body. Thus creating a relatively simple problem to solve.

2.3 Symbitron Exoskeleton

The exoskeleton for which the controller will be explored is the Symbitron Exoskeleton, designed by the biomechanical engineering department at the University of Twente [15]. The lower limb exoskeleton is designed to enable individuals with incomplete or complete spinal cord injury to walk again. The exoskeleton uses a modular design with four configurations for each leg. These configurations make it possible to only actuate certain joints, making it adaptable to the needs of a specific individual.

Using the full exoskeleton will result in 18 degrees of freedom, of which 8 are actuated. An overview of the exoskeleton and the actuators is shown in figure 2.3. The exoskeleton has a backpack, which contains the batteries and the computers responsible for calculations. Each leg has two passive joints: the hip inversion and eversion joint(HIE) and the ankle inversion and eversion joint(AIE). The other four joints, hip abduction and adduction(HAA), hip flexion and extension(HFE), knee flexion and extension(KFE), and ankle dorsi- and plantarflexion(ADP), are actuated.

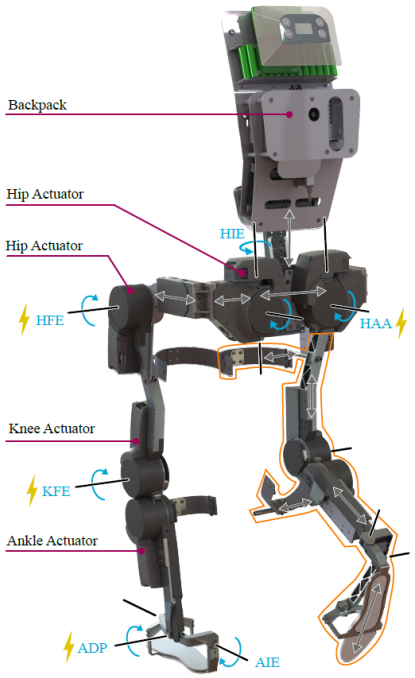


Figure 2.3: The Symbritron Exoskeleton designed by the biomechanical engineering deparment of the University of Twente [15].

2.4 General control schemes

When it comes to the control of exoskeletons and humanoid robots there are two main methods: online and offline control [5]. With offline control, the motion trajectory is calculated beforehand and then executed by the device. This allows for complex calculations because computation time is not an issue [16]. The trajectory can be planned using a detailed dynamic model of the device, resulting in an accurate and precise trajectory. The downsides of this method however are quite apparent. Because the calculations are done beforehand, the models used for the computation need to be very accurate because the system is not able to adjust to any deviations or unexpected disturbances.

With online control, the reference trajectories are calculated in real-time. The trajectories are usually determined for the next couple of steps. This trajectory is adjusted and updated regularly, allowing it to adapt to deviations and disturbances. The main limitation of this method is the limited computation time allowed. While using a detailed model of the system would result in an accurate trajectory, the computation times are often too long for online use.

One method of realizing online control is using multi-layered control. [1] This control method cuts down the computation time by splitting the control task into different layers. Each layer uses a different model that can be simplified according to the requirements of the layer-specific tasks. Most multi-layered controllers in exoskeletons are split up into three layers: high-level, mid-level, and low-level control [8].

Multi-layered control systems rely on different layers of control, using different models. The function of each layer depends on the goal of the controller. The high-level controller is responsible for intention detection. It determines what task needs to be executed by the device, because of this it is sometimes called the task-level controller. For augmenting exoskeletons this usually involves measurements on the wearer's legs to determine walking intention [8]. For autonomous walking exoskeletons and humanoid robots this controller uses a simplified model of the system, often a linear inverted pendulum (LIP), to calculate a motion trajectory. This trajectory usually includes at least a CoM trajectory and footholds. Due to the simplicity of the model used in this layer, the computation time is low [18, 19, 20].

The mid-level controller is used to transform the task given by the high-level controller into joint trajectories. Its output can either be desired joint forces or joint positions. In some cases, the mid-level controller's output is only a desired configuration. In that case, the low-level controller is responsible for translating this configuration to joint trajectories and controller joints.

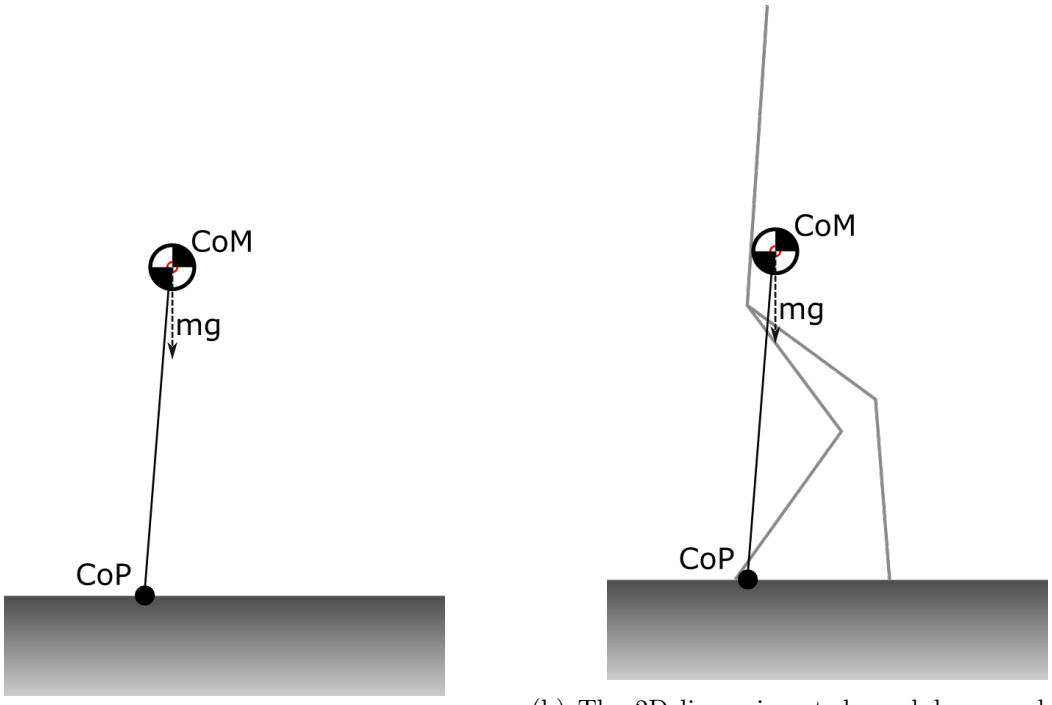
The low-level controller is usually responsible for executing the joint trajectories. It will receive the desired configuration for each joint.

2.5 High Level Control

The high-level controller is responsible for the trajectory and motion planning [20]. The high-controller often uses a simplified model for gait planning [17, 21, 22]. These models capture the overall dynamics of the system, without having the complexity of modeling each separate joint or movement.

This section first covers the model used to approximate bipedal motion. Then it goes into various balance strategies used for exoskeletons and humanoid robots. Lastly it covers the preferred walking motion.

[1] This type of control system is sometimes also called hierarchical control, for example in [5]. However, this term is also used for systems that stack quadratic programs, for example in [17]. The term multi-layered is used in this report to prevent confusion.



(a) The 2D linear inverted pendulum.

(b) The 2D linear inverted pendulum overlapping with a humanoid model.

Figure 2.4: Linear inverted pendulum model. The motion is determined by the horizontal position of the CoP relative to the CoM.

2.5.1 Linear Inverted Pendulum

The most widely used model to approximate the motion of bipedal robots is the 2D or 3D linear inverted pendulum model [21]. This model is comprised of a floating mass, kept at constant height, and a massless leg link of variable length, which allows for force to be applied to the floating mass, illustrated in figure 2.4. There are multiple variations of this model. The most simple is the so called point-foot, whereby the link ends in a point on the floor. The center of pressure is thus always at this specific point of contact. Some models do include an ankle toque at the point foot to add more stabilizing mechanisms. This model can be expanded by replacing the point foot with a finite sized foot. This, combined with an ankle torque, does allow for the center of pressure to vary within a the limited area of the foot [23]. This model has shown to be sufficient for creating motion trajectories for humanoid robots [21, 24]. The model, however, could be expanded by adding another torque between the link and the floating body. This addition allows for a bit more momentum control. However, this extra momentum seems only to be applicable to a single step [23].

2.5.2 Balance strategies

The main focus of almost all gait generators is stability. Humans and humanoids are highly unstable when it comes to balance [21]. This can easily be seen when looking at the LIP approximation of human dynamics. If the stick or foot is not precisely underneath the CoM it supports, the system will topple.

While stationary, balance can be achieved by having the center of mass be above the base of support (BoS), which is the convex hull around the contact points with the ground. This criterion for stationary stability is based on the principle that for each CoM position above the BoS a CoP can be generated to keep the position of the CoM constant. This stabilizing CoP lies directly underneath the CoM. During movement, however, this is not deemed sufficient since

this criterion does not take the momentum of the center of mass into account. Another criterion thus has to be defined to take this movement into account [23, 25].

To combat the stability problem, various stability criteria are used. Three prevalent used criteria will be covered.

Extrapolated Center of Mass

To deal with the stability problem, Hof et al. introduced the extrapolated center of mass(XcoM) in 2005 [25]. The XcoM is a projection of the CoM on the supporting surface with an added factor of the velocity of the center of mass. A CoP generated at the XcoM is able to let the CoM come to a standstill above the XcoM. So to always be able to come to a standstill, the XcoM should lie within the base of support [25, 26].

The concept is created from the linear inverted pendulum model. When the CoP is not directly underneath the CoM, the pendulum falls over. This acceleration is based on the horizontal distance between the CoP and the projection of the CoM on the supporting surface and the length of the pendulum. Thus the position XcoM is based on the current position of the CoM plus its velocity multiplied a factor of $\sqrt{l/g}$, with g being the acceleration of gravity and l the leg length. The full formula is shown is given by

$$\xi = x_C + \frac{1}{\omega_0}\dot{x}_C, \tag{2.4}$$

where ω_0 is the eigenfrequency of the LIP defined as $\sqrt{g/l}$ [25, 27].

The stability theory of the XcoM is based on the instantaneous placement of the CoP on the XcoM. However, these criteria cannot be met in practical settings, especially during walking where a step has to be taken to keep the XcoM within the base of support. Thus it is important to look at the development of the XcoM over time.

Looking at the LIP model, the acceleration of the CoM depends on the distance between the CoP and the projection of the CoM. So when the CoP does not move, the position of the XcoM moves away from the CoP exponentially. Given this, an equation for the position of the XcoM at a certain point is given by

$$\xi(t) = (\xi_n - x_{P,n})e^{\omega_0 t} + x_{P,n}, \tag{2.5}$$

where the subscript n indicates the position of the current time-step [27].

ZMP

Another stability criterion is to keep the Zero Moment Point (ZMP) within the BoS. This criterion has been extensively explored and applied by, among others, L.Lanari and S.Hutchinson [16, 21, 24]. The ZMP is defined as the point where the effects of all forces acting on the device can be replaced by a single force [26]. It is important to note that this point can only be within the BoS. Translating this to the LIP model means that if the end of the stick is placed in the ZMP and the right force is applied, the CoM will come to a standstill exactly above this point. This point must lie within reach of the stick. If the ZMP would be outside the BoS, it is labeled as the fictitious ZMP. This means that the device cannot come to a standstill with the current foot placement and thus another step would be needed to bring the ZMP within the BoS. Generating a trajectory where the ZMP always exists results in the system being able to come to a standstill on every point along the trajectory, thus ensuring a measure of stability.

N-step capturability

The ZMP based stability margin is based on being able to come to a standstill without the need to take extra steps. While this method has been applied successfully to multiple mechanisms,

there are some limitations. These limitations arise due to circumstances that cause the ZMP to leave the BoS, thus becoming a fictitious ZMP. This can for example happen due to disturbance or moving at higher velocity [23, 26]. The N-step capturability, proposed by Koolen et al. [23], is defined by a system being able to come to a stop in N steps or less. This stability analysis is thus not limited by its current BoS. The method is based on the instantaneous capture point (ICP). This point is defined as the point on the ground that enables the system to come to a stop if it were to instantaneously place and maintain its CoP pressure there. This point is not fixed and moves over time. The ICP moves on the line from the foot through the ICP, away from the foot. The velocity is proportional to the distance to the foot. It is important to notice that this causes the velocity to increase exponentially when the foot remains stationary. Taking into account the step size and step time, N-step regions can be determined. The size of the regions is dependent on the number of balance strategies involved and the current velocity and acceleration of the center of mass. When designing a trajectory that is N capturable, the system must always stay within the N or lower capture region around the ICP. Due to the exponential velocity of the ICP, the ∞ and 4-step capture regions are almost the same size. As such, capture regions of higher N than four are not relevant.

2.5.3 Walking motion

While the high-level controller provides preferred footholds, the simplified model used for this is not capable of modeling leg swing. The leg swing is determined by the position of the chest and a foot trajectory.

Humanoid robot designs have shown that a simple polynomial-shaped foot trajectory is sufficient to achieve decent walking motion [19, 28]. Assuming the normal human gait is optimized when it comes to energy consumption, a similar movement would likely make be optimal for exoskeleton gait as well. Research performed by Wu et al. [29] looked at the energy consumption of the human gait for different levels of ground clearance and scuffing. The measurements showed that both larger ground clearance and scuffing significantly increase energy consumption. It concluded that the likely minimized cost is achieved when ground clearance is kept at a minimum, without actually touching the ground to prevent scuffing or even stumbling.

2.6 Mid-level controller

There are many ways to design a mid level controller. The simplest method is to use pre-programmed motions. With this method the joint trajectories for certain tasks are determined beforehand. While fixed trajectories allow for precise and smooth joint trajectories, it is not very robust to disturbances and is limited to only the pre-programmed motion. Therefore the amount of different tasks the exoskeleton can be used for are limited.

While developments in battery technology have significantly increased the energy density of batteries, the amount of energy that can be carried is still a major limitation of exoskeletons [4]. To limit the energy consumption it is preferred to generate the most energy efficient gait. The method that lends itself the best for this is the quadratic program controller.

2.6.1 Quadratic program controllers

Quadratic program (QP) controllers consist of two parts: a cost function it has to minimize and one or multiple constraints [30]. The cost function is a summation of multiple factors and their cost scaling. Most optimal controllers minimize a certain balance of accuracy and energy input. These controller have thus at least the error variable and input in their cost. The input/energy cost can be inserted in multiple ways. Preferably the actuation energy cost is included, however, to cut computation time effects that cause more energy consumption can be included instead, such a join torque or acceleration.

Other factors in the cost function are usually based on design choices. These are factor the designers wants to be as close to zero as possible, however, they don't necessary have to be zero.

The constraints contain requirements that have to be met. Examples of this are laws of physics and configuration. A common constraints in humanoid robots are range of motion of the joints, the relationship between joint motion and body motion and the equation of motion.

Which factors are included in the cost function and which are included in the constraints depend on the design and controller design choices. In some cases the computation time can be lowered by moving factors from the constraints to the cost function. This way the controller will try to adhere to the constraint as much as possible, but doesn't have to make it exactly zero. Whether to do this or not is dependant on the computation time and necessary accuracy of the constraints.

2.6.2 Force limitations

There are a couple of force limitations that need to be taken into account. One clear force limitation comes from the limited actuator torque. This can be considered a constant limit, which simplifies this limit. However, a more detailed model might be necessary which includes back EMF, where the torque limit is dependent on the joint velocity. Whether or not this is necessary thus depends on the joint velocities reached during the walking motion and has to be determined during the design process.

Another limitation comes from interaction with the environment, mainly making sure slipping does not occur. One way to implement this is the contact wrench cone (CWC). A CWC is built of all possible wrenches that could be applied to a surface at a specific point without slipping. While this cone would in reality be a complicated shape, it seems to be sufficient to approximate it as a square-based pyramid. The CWC usually assumes that the surface is capable of handling an infinite normal force. Taking the Minkowki sum of the friction cones gives all the wrenches that the exoskeleton can apply to the environment without slipping. [31]

By taking the Minkowski sum of the intersections of all the friction cones and the possible actuation wrenches the Feasible Wrench Polytope (FWP) can be created. This is a convex three-dimensional shape that contains all possible wrenches that can be applied by the exoskeleton without slipping. [31] One downside of using the FWP as a factor in the quadratic program is the expensive computation. This may result in the controller being too slow to be effective.

2.7 Low Level Control

Low-level control encompasses the control of individual, actuated, joints. Each joint needs a controller to execute the tasks given by the higher-level controllers. The individual controllers have two main objectives: follow the joint-specific trajectory, either force or position trajectory, and keep the motor position within the desired limits.

2.7.1 Joint controllers

There are two main types of controllers used, based on the input given by higher-level controllers: torque controllers and position controllers [5].

With torque control, the input of the low-level controller consists of a desired torque. The goal of the low-level controller is to generate the input torque. One method of achieving this is by controlling the energy input with a PD controller. A PD controller calculates an input based on the proportional and derivative error.

With position control, the input of a the low-level controller consists of a desired position or acceleration. The low-level controllers goal is to generate a torque that brings the joint in the desired position or achieves the desired joint acceleration. This torque will be regulated by an internal torque controller.

2.7.2 Joint position limitations

One method to adhere to the desired joint position limits while trying to minimize the required torque is to limit the joint velocity near these limits. Meijneke et al. [15] implemented a formula that determined the maximum velocity based on the distance to the limit and the maximal deceleration,

$$v_{max}(\phi) = \sqrt{2a_{max}(\phi - \phi_{limit})}. \tag{2.6}$$

The method was successful in creating a strong limit while staying within the torque limits. The downside is that tracking is poor near the joint limits.

Another method is to create a virtual spring-damper near the limits. When nearing the edge, this model will create a virtual counter force that pushes the joint away from the limit. The main downside of this method is that this virtual force is applied regardless of the input. This will cause the joint to be pushed away from the limits when the input is lower than the virtual spring damper.

2.8 Related momentum-based controllers

In the last decades, there have been a lot of developments regarding the control of humanoid robotics and exoskeletons. For these systems, a wide variety of control systems are explored and researched. To remain within the scope of this report, only the momentum-based controllers are discussed.

Machietto et al. [32] used momentum-control to create a controller that is capable of controlling the balance of a humanoid model. They used the relationship between the momentum rate and the CoM acceleration to keep the CoM close to the center of the BoS. The angular momentum rate was used to determine the position of the CoP. The goal was to maintain balance while the model was executing a perpetual leg swing with one leg and being disturbed. The controller consisted of a single QP that used tracking goals in the form of joint accelerations for the swing leg and momentum rate with a constraint to keep the stance foot at a constant location.

Koolen et al. [28] implemented a momentum-based controller on the humanoid robot Atlas for the DARPA robotics challenge in 2015. The controller consisted of a high-level controller that translated the human operator's input into motion tasks for the mid-level controller. The mid-level controller consisted of a quadratic program and an inverse dynamics calculator. The QP used a momentum rate tracking cost and constraint the feet acceleration tracking. It used the momentum-rate equation as its overall dynamics constraint, which required an inverse dynamics calculator for force calculation. During the challenges, the team was able to make the robot climb into vehicles, walk various terrain and handle objects [28, 33].

Herzog et al. [17] designed a hierarchical momentum-based controller for a lower limb humanoid. The controller consists of multiple QP which are solved in sequence. The sequence of QPs is based on a task priority. Tasks with the highest priority are solved by the first QP in the sequence. Lower priority tasks and subsequently solved in other QPs in the sequence. The QPs must operate in the null space of the tasks solved by QP higher in the sequence. With this controller, the group is able to robust balance control, in both double and single support. Stepping and walking have not been tested.

A recent contribution was made by Soliman and Ugurlu. [34]. The duo designed a task prioritization algorithm for an lower limb underactuated exoskeleton. The exoskeleton is similar in design as the Symbitron, section 2.3. The task prioritization algorithm is hierarchical and functions in a similar fashion as the hierarchical controller of Herzog et al. ([17]). With Soliman's design, QPs lower in the sequence used the results of higher QPs as equality constraints, thus ensuring that lower priority tasks do not disrupt the higher priority tasks. Balanced was achieved by keeping the CoM with a virtual spring-damper in the middle of the BoS. They implemented

this system with three different controller types: ZMP impedance feedback, basic admittance and momentum-based control. Using this system they achieved stable unperturbed and perturbed walking at velocities of 0.3 and 0.2 m/s.

Chapter 3

Model and Dynamics

This chapter covers the model for which a controller is designed. Section 3.1 contains an overview of the model used. Section 3.2 goes into the dynamics model and section 3.3 covers the contact model used.

3.1 Model overview

The goal of the model is to approach an exoskeleton as a 2D humanoid. The exoskeleton covers the lower limbs of the wearer. This is approached by creating a humanoid consisting of a torso and two legs. The upper limb movement of the wearer can be simulated as a disturbance to the exoskeleton. The model does not include feet. The feet were excluded to simplify the model as adding feet would double the number of contact points and increase the degrees of freedom. The model has a total of seven degrees of freedom, three from the floating base attached to the CoM of one of the legs and four actuated joints. An overview of the model is shown in figure 3.1.

3.2 Dynamic Model

The total dynamics of the 2D humanoid is given by:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = S_\tau \tau + J^T(q)\rho,$$

The procedure to derive the dynamic model is largely taken from [35]. The theory is implemented by Arvid Keemink and Ander Vallinas-Prieto. This section is written by Arvid Keemink and adjusted to fit the report.

The kinematics of the humanoid structure are shown in Fig. 3.1. The locations of the center of mass for each body i ($p_{c,i}(q)$) can be determined from the model's kinematic parameters and q and be expressed in an inertial world frame. Each body has mass m_i and rotational inertia I_i. We can then determine the following (T describing the total kinetic energy and ω_i the absolute

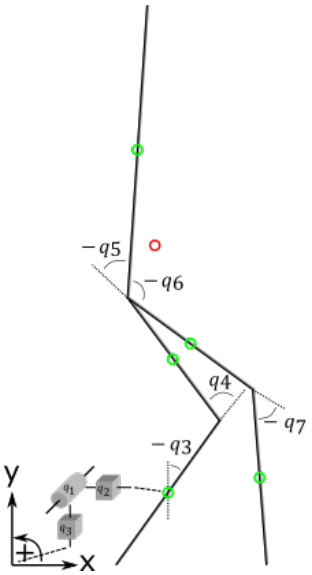


Figure 3.1: Overview of the 2D humanoid model. The model has 7 degrees of freedom. The first three are the floating body DoF, given by the coordinates and rotation of the back leg with respect to the world coordinates. The CoM is indicated with the red circle and the green circles are the CoM of each body segment.

rotational velocity of a body):

$$
\begin{aligned}
\dot{p}_{\mathrm{c},i}(q,\dot{q}) &= J_{\mathrm{c},i}\dot{q} = \frac{\partial p_{\mathrm{c},i}}{\partial q}\dot{q} \\
\omega_i &= J_{\omega,i}\dot{q} = \frac{\partial \omega_i}{\partial q}\dot{q} \\
T &= \sum_{i=1}^{6} \frac{1}{2}\left\{\dot{p}_{\mathrm{c},i}^T m_i \dot{p}_{\mathrm{c},i} + \omega_i^T I_i \omega_i\right\} \\
&= \frac{1}{2}\sum_{i=1}^{6}\left\{\dot{q}J_{\mathrm{c},i}^T m_i J_{\mathrm{c},i}\dot{q} + \dot{q}J_{\omega_i}^T I_i J_{\omega_i}\dot{q}\right\} \\
&= \frac{1}{2}\dot{q}\left(\sum_{i=1}^{6}\left\{J_{\mathrm{c},i}^T m_i J_{\mathrm{c},i} + J_{\omega_i}^T I_i J_{\omega_i}\right\}\right)\dot{q} \\
&= \frac{1}{2}\dot{q}M(q)\dot{q} \\
M(q) &= \sum_{i=1}^{6}\left\{J_{\mathrm{c},i}^T m_i J_{\mathrm{c},i} + J_{\omega_i}^T I_i J_{\omega_i}\right\}.
\end{aligned}
$$

The Coriolis and Centrifugal contributions is given by $C(q,\dot{q})\dot{q}$. The elements of $C(q,\dot{q})$ can be determined as follows (for $q \in \mathbb{R}^7$):

$$
\begin{aligned}
C_{ij}(q,\dot{q}) &= \sum_{k=1}^{7} c_{ijk}(q)\dot{q}_k \\
c_{ijk}(q) &= \frac{1}{2}\left(\frac{\partial M_{ij}(q)}{\partial q_k} + \frac{\partial M_{ik}(q)}{\partial q_j} - \frac{\partial M_{jk}(q)}{\partial q_j}\right)
\end{aligned}
$$

The gravitational contribution can be found because we already have the center of mass of each

body i ($p_{\mathrm{c},i}(q)$) and determine the generalized force from the potential energy $V(q)$:

$$V(q) = g \sum_{i=1}^{6} \begin{bmatrix} 0 & 1 \end{bmatrix} p_{\mathrm{c},i}(q) m_i$$
$$G(q) = \frac{\partial V(q)}{\partial q}.$$

The Jacobian $J(q) \in \mathbb{R}^{4\times 7}$ of the interaction locations (the point-feet) is determined by:

$$J(q) = \begin{bmatrix} \frac{\partial p_{\mathrm{foot1}}(q)}{\partial q} \\ \frac{\partial p_{\mathrm{foot2}}(q)}{\partial q} \end{bmatrix},$$

where $p_{\mathrm{foot1}}(q)$ and $p_{\mathrm{foot2}}(q)$ are the point feet locations, determined from the kinematic model. Joint-level damping is trivially added as an extra generalized force $\tau_d = -B\dot{q}$ with B being positive semi-definite and typically diagonal.

3.3 Contact Model

The contact model is based on the work of [36], but is adapted from the 3D case to the 2D sagittal case. The code was implemented by Ander Vallinas-Prieto and Arvid Keemink.

The model determines the force impulse required to prevent a contact point from moving into the ground once contact is made. Furthermore, it determines the forces generated by friction according to the Coulomb friction model.

This model is chosen over a spring-damper contact model because it does not create a bounce effect. With the spring-damper model, the ground is modelled as a dampened spring. Upon contact, the spring generates a force to push the object out of the ground. This force is relative to the displacement of the ground by the contact. This pushing force can cause the object to lose contact with the ground and then fall back. The force impulse model only prevents the object from acceleration into the ground and thus does not create a bounce effect.

The model is described in more detail in appendix A.

Chapter 4

Controller design

This chapter covers the design of the controller. Section 4.1 covers the design goals. Section 4.2 goes into the overall design of the controller and sections 4.3 and 4.4 cover the high- and mid-level of the controller.

4.1 Design Goals

The controller will be a real-time momentum-based controller. The goal of the controller is to track a given CoM trajectory while minimizing the amount of movement and energy required. The high-level controller should be capable of generating a momentum trajectory given a CoM path. It then has to generate a momentum rate reference that, if executed correctly, is capable of tracking the momentum trajectory. Furthermore, it has to generate feet trajectories and feet accelerations capable of tracking these trajectories. The mid-level controller has to find the optimal joint torque that executes the momentum rate and feet acceleration references while minimizing the energy expenditure.

The controller should be capable of handling two situations: walking at a constant pace and recovering after a push. The walking will be simulated by giving the controller a CoM trajectory with constant velocity. The push recovery will be simulated by applying a force impulse on the CoM.

4.2 Controller Overview

Figure 4.1 shows a high-level overview of the control framework. The controller consists of two layers: a high-level and a mid-level controller. Low-level control is assumed to be perfect and thus is not implemented in the framework. The high-level controller consists of two systems that are together responsible for creating tracking goals for the mid-level controllers. The mid-level controller consists of a quadratic program that reconciles the tracking goals with the dynamics and determines the optimal joint torques for the robot model. The robot model gives state feedback to the control systems.

4.3 High-level control

The high-level controller consists of two systems: the reference generator and the motion reference planner. The reference generator creates reference trajectories for the CoM and feet. The CoM trajectory is used as input and can be seen as the general task for the robot. The feet trajectory is reactionary and generates stepping trajectories only when required to maintain balance. These trajectories are then sent to the motion reference planner. The motion reference planner consists of PD controllers that generate momentum rate and acceleration goals for the

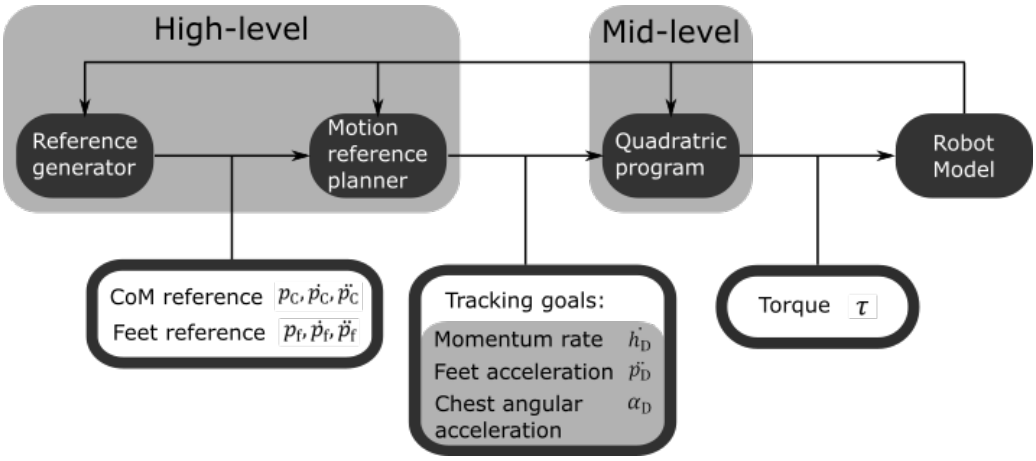


Figure 4.1: High level overview of the information flow in controller framework.

mid-level controller. The CoM trajectory is transformed into a momentum rate that pulls the CoM towards the trajectory, creating a momentum controlled system. The feet trajectories are transformed into feet acceleration goals.

4.3.1 Reference Generator

The reference generator (RG) creates a desired reference trajectory for the CoM and feet positions, velocities, and accelerations at each time step. With a two-dimensional model, each object has three configuration variables, one angular and two linear. Furthermore, for each variable, the first two derivatives will be used. For the total system, this results in 27 trajectory variables, nine for the chest and nine for each foot. These values are then used by the Motion Reference Planner to create desired momentum rates and accelerations for the CoM and the feet.

The purpose of the CoM trajectory is to give an overall desired motion plan. As such the CoM trajectory is predetermined and does not depend on the current state of the model. If a selected trajectory consists of a stationary point, the robot will try to reach that point or return to it when disturbed. If, on the other hand, a moving trajectory is made, the robot will follow that trajectory and thus start walking. This design approach allows for a simple way to give the robot tasks, such as walking to a certain point, executing a specific movement, or remaining as stationary as possible.

Contrary to the CoM trajectory, the feet trajectory is reactionary and thus is dependent on the state and dynamics of the model. The feet trajectory generation is controlled by a state machine, shown in figure 4.2. The feet reference will remain in constant positions on the supporting surface until a step is required to maintain stability. Once an appropriate step is taken, the feet will remain stationary again.

The main benefit of this system lies in that no specific feet trajectories are required as input beforehand. The robot will automatically step when necessary to follow the desired CoM trajectory or to remain upright when disturbed. The main downside of the current implementation of the reference generation is that the height of the CoM trajectory has to be limited to ensure feasibility. Since the stepping method is purely based on maintaining stability, it does not consider the range of motion. This can result in the desired feet position moving outside of the maximum range of the legs. If this occurs, the controller will attempt to match the desired trajectory as much as possible, however, balance cannot be guaranteed. These situations can be prevented by limiting the CoM height. This crouched gait requires higher torques compared to a more upright gait, which is a significant downside.

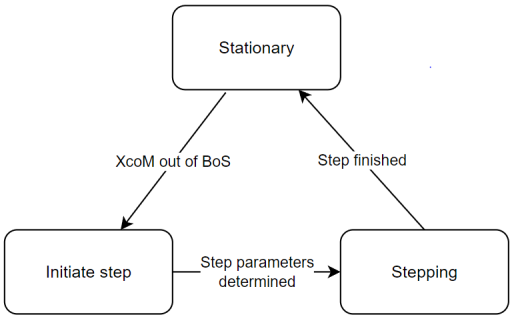


Figure 4.2: The state diagram used for the feet trajectory generation.

Stepping decisions

The robot will automatically step when necessary to remain stable. To determine whether a step is needed to maintain stability, the criterion of the XcoM is used. This criterion states that a step is required when the XcoM moves out of the BoS.

When making stepping decisions, the desired CoM velocity and acceleration should be considered. As can be explained with the LIP model, the acceleration of the CoM during normal movement is determined by the difference in the horizontal position of the CoP and the CoM. During movement, the acceleration is determined by the difference in the horizontal position of the CoP and XcoM.

If a step is taken at the exact moment the XcoM leaves the BoS, the XcoM will be close to the back foot after the step has been taken. The CoP can only be moved within the BoS. If the XcoM is close to the edge of the BoS, the maximum distance between the CoP and XcoM is limited, thus limiting the maximum achievable acceleration. Delaying the stepping decision will result in the XcoM being located further from the edge of the BoS after the step has been taken, thus allowing a wider range of accelerations.

The trajectory generator will initiate a step when the XcoM is a certain distance away from the edge of the BoS, thus ensuring that the XcoM does not end up too close to the edge of the BoS. This distance is set as a percentage of the reference velocity, up to a maximum value. In the current implementation, this percentage is set at 20%. This method results in a stable gait without limiting the possible acceleration of the CoM by stepping too close to the XcoM.

Stepping trajectory

The stepping trajectory is determined by two parts: the shape of the trajectory and the height and length of the motion. The shape of the trajectory is a sinusoid in both the horizontal and vertical direction. The main benefit of using these trajectory shapes is the low velocity at the start and end of the motion. The motion profile is given by

$$x_f = \frac{l_s}{2} \sin\left(t\frac{\pi}{t_T} - \frac{\pi}{2}\right) + \frac{l_s}{2} \tag{4.1}$$

$$y_f = \frac{h_s}{2} \sin\left(t\frac{2\pi}{t_T} - \frac{\pi}{2}\right) + \frac{h_s}{2}, \tag{4.2}$$

where l_s represents the step length, h_s the step height, x_f and y_f the horizontal and vertical position of the foot respectively and t_T the total step time.

As discussed in section 2.5.3, the energy needed for walking is minimal with minimal ground clearance, provided enough clearance to prevent scuffing. Since the optimal solution is determined by a quadratic program, the feet may deviate from the trajectory. The step height is thus kept low, however not so low as to risk scuffing due to controller decisions.

The stepping decision is based on the XcoM leaving the BoS, thus the goal of a step is to move the BoS such that the XcoM again lies within it. To accomplish this, the foot has to be placed further than the XcoM, keeping in mind the time it takes to execute the step. Using a fixed step time and assuming no disturbances while stepping, the position of the XcoM after a step is executed can be calculated with equation 2.5. The ideal step length is then determined by taking this position and adding 10 percent of the difference between this position and the current foot position. If the ideal step length exceeds the maximum step length, the maximum step length is used instead. A minimal step length is also implemented to prevent the BoS from becoming too small, which can result in the robot being required to take unnecessary steps.

Once the foot hits the ground, the foot reference is changed to keep the foot stationary on that spot. This is done to prevent the robot from trying to slide on the ground when the feet do not hit their exact target, resulting in unnecessary energy expenditure and less stable behavior.

4.3.2 Motion Reference Planner

The Motion Reference Planner(MRP) transforms the trajectory given by the reference generator into momentum rate and acceleration goals for the mid-level controller. These transformations are done by using PD control to correct for deviations between the actual and reference trajectories, after which the reference accelerations are added. Since the CoM goals are momentum rates, the desired accelerations have to be multiplied by the mass or inertia of the robot. The desired momenta rates and accelerations are given by

$$\dot{k} = P_\theta \left(\theta_{C,r} - I\dot{q}\right) \tag{4.3}$$

$$\dot{l} = P_{\mathrm{C}} \left(p_{\mathrm{C,r}} - p_{\mathrm{C}}\right) + D_C \left(\dot{p}_{\mathrm{C,r}} - \dot{p}_{\mathrm{C}}\right) + m_r \ddot{p}_{C,r} \tag{4.4}$$

$$\ddot{p} = P_{\mathrm{f}} \left(p_{\mathrm{f,r}} - p_{\mathrm{f}}\right) + D_{\mathrm{f}} \left(\dot{p}_{\mathrm{f,r}} - \dot{p}_{\mathrm{f}}\right) + \ddot{p}_{\mathrm{f,r}}, \tag{4.5}$$

where I is the inertia matrix, P and D are the proportional and derivative gains, subscripts θ,C and f refer to angle, CoM and foot and the additional subscript r refers to the reference value.

The controller is real-time and thus does not consider long-term strategies. Because of this, the controller can make decisions that are favorable in the short term but are not sustainable long term. One example is using the upper body angle to move the CoM, which will cause the robot to fall over. To prevent this, the MRP generates a chest angle tracking goal given by

$$\alpha_D = P_{\mathrm{a}} \left(-\frac{5}{180}\pi - S_{\mathrm{c}}\ddot{q}\right) + D_{\mathrm{a}} \left(0 - S_{\mathrm{c}}\dot{q}\right), \tag{4.6}$$

with α_D being the desired angular acceleration of the chest, P_{a} and D_{a} are the proportional and derivative gains and the selection matrix for joints that determine the chest orientation given by

$$S_{\mathrm{c}} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}. \tag{4.7}$$

All PD gains are chosen to be critically damped. This ensures accurate tracking while reducing overshoot. For the momentum rate controllers, the mass of the robot is incorporated into the gains. A limit is set on the maximal horizontal linear momentum rate of the CoM. This ensures that the momentum rate goals are not higher than the system can handle.

4.4 Quadratic program

The MRP outputs momentum rate and feet acceleration goals. The quadratic program (QP) reconciles these goals with the dynamics of the robot and finds the optimal joint torques at each controller time step. The optimal joint torque is found by minimizing the trajectory deviation, joint accelerations, and ground reaction forces.

The remainder of this section covers the different components of the QP. The order of these sections is chosen to give the best understanding of how the program is built up. First, the constraints are discussed. The constraints contain the dynamics of the robot and thus give an understanding of how the movement is calculated. After that, the decision variables are discussed. The last subsections cover the full QP.

4.4.1 Constraints

The QP used has a total of five equality constraints and one inequality constraint. Three constraints are used to capture the dynamics of the robot and ground contacts. The other three are used to facilitate the tracking of the tracking goals given by the high-level controller.

Overall dynamics

The overall dynamics of the robot are captured in an equation of motion (EoM) equality constraint. The forces applied on the degrees of freedom of the robot consist of the force generated in the actuated joints and external forces such as the ground reaction forces. The equation of motion is described as:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = S_\tau \tau + J^T(q)\rho, \tag{4.8}$$

where $M(q)$ is the mass matrix, $C(q,\dot{q})$ the matrix containing the Coriolis and centrifugal forces, G the gravitational forces, S_τ the selection matrix for the actuated joints, and Q the Jacobian for the contact forces.

The M, C, G, and Q matrices are calculated at each time step and depend on the configuration of the robot. The robot has seven degrees of freedom, of which the last four are actuated. Selection matrix S_τ is thus defined as:

$$S_\tau = \begin{pmatrix} 0_{3\times 4} \\ I_{4\times 4} \end{pmatrix}, \tag{4.9}$$

where I is an identity matrix.

The overall dynamics of the robot could also be described by the momentum rate equation given as 2.3. The main argument for choosing for using the EoM as the main dynamic equation is that the torque is included in the EoM equation. This allows the torque to be used as a decision variable, resulting in the optimal torque being determined by the QP. If the overall dynamics are described by the momentum rate equation, the required torque to execute the desired joint acceleration has to be determined outside of the QP via inverse dynamics. If torque is a limiting factor, the torque limits can easily be enforced when it is a decision variable. If the torque is not a decision variable, the torque limits have to be enforced via the $\ddot{q}$ and ρ limits indirectly. The EoM is chosen to make the torque limit more intuitive.

Contact constraint

The feet of the robot are modeled as point feet, resulting in a total of two possible contact points with the ground. The contact points can only apply force to the ground when in contact with the ground. The first contact constraint is a complementary constraint in the form of

$$d_\mathrm{f}\rho_\mathrm{c} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{4.10}$$

where d_f is the vertical distance between the foot and the ground and ρ_c is the grf of one foot.

Once in contact, the contact points are subject to two interaction constraints: the feet can only push on the ground and the feet must not slip. The first of these constraints can easily be

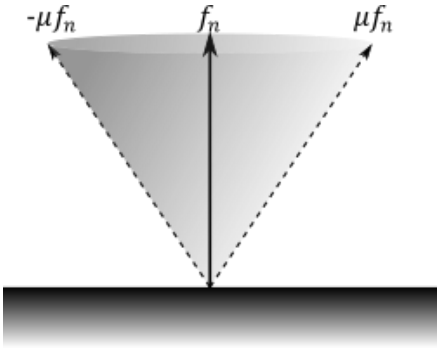


Figure 4.3: Friction cone according to static Coulomb friction. The friction is linearly dependent on the normal force. Slipping will occur when the tangential force exceeds the friction force, which is shown as the gray cone.

enforced by requiring the grf to be net positive along the normal vector of the contact surface. The ground reaction forces are solved in a Cartesian frame. To apply the constraint the Cartesian grfs have to be projected on the normal vector resulting in the following constraint:

$$P_{\mathrm{n}}\rho_{\mathrm{c}} \geq 0, \tag{4.11}$$

where ρ_c is the grf of one contact point and P_n is the vector projecting the grf onto normal vector given by

$$P_{\mathrm{n}}^T = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix}, \tag{4.12}$$

with θ the slope of the contact surface. Please note that P_{n} is transposed in the equation.

The friction is modeled as static Coulomb friction. This entails that the tangential force should not exceed the Coulomb friction. The friction is linearly dependent on the normal force, resulting in:

$$-\mu f_{\mathrm{n}} \leq f_{\mathrm{t}} \leq \mu f_{\mathrm{n}}, \tag{4.13}$$

with μ the friction coefficient, f_{n} the grf along the normal vector of the supporting surface and f_{t} the grf tangent to the normal vector. A visual representation of the friction cone is shown in figure 4.3.

To apply this constraint the grf has to be projected into the normal and its tangential vectors resulting in:

$$-\mu P_{\mathrm{n}}\rho_{\mathrm{c}} \leq P_{\mathrm{t}}\rho_{\mathrm{c}} \leq \mu P_{\mathrm{n}}\rho_{\mathrm{c}}, \tag{4.14}$$

where P_{n} is given by 4.12 and P_{t} is the tensor projecting the grf onto the tangential vector given by

$$P_{\mathrm{t}}^T = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}. \tag{4.15}$$

Combining equations 4.11 and 4.14 into one matrix results in a single inequality constraint

$$\begin{pmatrix} -P_{\mathrm{n}} \\ P_{\mathrm{t}} - \mu P_{\mathrm{n}} \\ -P_{\mathrm{t}} - \mu P_{\mathrm{n}} \end{pmatrix} \rho_{\mathrm{c}} \leq 0. \tag{4.16}$$

Trajectory tracking

The Motion Reference Planner generates a total of four references to track, one momentum rate, two feet accelerations and one chest angular acceleration. The tracking can be done with either constraints or include the equations directly into the cost. In this section we consider them as constraints first.

For the momentum tracking, a momentum constraint has to be added. Using equation 2.3 and replacing the momentum rate with the desired momentum rate gives:

$$\dot{A}(q)\dot{q} + A(q)\ddot{q} = \dot{h}_{\mathrm{D}}, \tag{4.17}$$

where $\dot{h}_D$ is the desired momentum generated by the MRP.

The velocity of the foot can be calculated with:

$$J_{\mathrm{f}}(q)\dot{q} = v_{\mathrm{f}}, \tag{4.18}$$

with J_f the geometric Jacobian of the foot and v_{f} the velocity of the foot.

Taking the time derivative and replacing the acceleration with the desired acceleration $\ddot{p}_D$ from the MRP gives the feet acceleration constraint

$$J_{\mathrm{f}}(q)\ddot{q} + \dot{J}_{\mathrm{f}}(q)\dot{q} = \ddot{p}_{\mathrm{D}}. \tag{4.19}$$

The chest orientation is determined by three rotational joints in series. The tracking can thus be done by adding the angular accelerations of these joints, resulting in

$$S_{\mathrm{c}}\ddot{q} = \alpha_D, \tag{4.20}$$

where S_{c} is given by equation 4.7.

4.4.2 Decision variables and cost functions

Choosing to use the EoM as the overall dynamics equation requires the quadratic program to be solved for the joint acceleration, ground reaction forces, and joint torques. As such, these variables are automatic inclusions in the cost function of the QP. However, these variables do not necessarily need to have a cost associated with them in order to find a solution.

It is desired to achieve the tracking goals while minimizing the amount of movement and energy required to achieve them. There is a multitude of decision variables that can be chosen to achieve this. For a clearer explanation, the decision variables will be split into operating costs and tracking costs.

Operating costs

The operating costs consist of the costs associated with executing the movement. Three clear variables can be associated with this cost: joint acceleration, ground reaction forces, and joint torque. By including a joint acceleration cost the QP minimizes the number of changes in the movement. This both helps create a more fluent movement pattern and minimizes the number of unnecessary movements. The inclusion of a grf cost can serve two purposes: it minimizes the contact forces, which reduces strain to both the environment as well as to the robot itself, and it will spread out the grf in the case of multiple contact points. Putting a cost on torque can serve the purpose of reducing the cost. However, as shown by Herzog et al. [17], decomposition of the EoM into the floating base and internal degrees of freedom leads to the equations

$$M_{\mathrm{b}}(q)\ddot{q} + C_{\mathrm{b}}(q)\dot{q} + G_{\mathrm{b}}(q) = J_{\mathrm{b}}^T\rho, \tag{4.21}$$

$$M_{\mathrm{j}}(q)\ddot{q} + C_{\mathrm{j}}(q)\dot{q} + G_{\mathrm{j}}(q) = \tau + J_{\mathrm{j}}^T\rho, \tag{4.22}$$

where subscript b and j represent the body and joint matrices respectively. The second shows that the torque, τ, is linearly dependent on the joint acceleration and grf. Thus associating a cost with the torque is not needed if costs are already applied on the joint acceleration and grf.

Applying costs to the joint acceleration and grf, the operation cost becomes

$$C_{\mathrm{o}} = \underset{\ddot{q},\rho,\tau}{\text{minimize}} \quad \ddot{q}^T C_{\mathrm{v}}\ddot{q} + \rho^T C_\rho \rho + \tau^T C_\tau \tau, \tag{4.23}$$

with C_{v} and C_ρ the costs for joint acceleration and grf and C_τ being a zero matrix for the torque. This cost has a total of 15 decision variables: 7 joint accelerations, 4 torques, 4 grfs.

Performance costs

The performance costs are used to facilitate the tracking. There are multiple options to implement the tracking into the cost. One approach is to directly put the tracking constraints into the cost function. For the momentum rate, using equation 4.17, this would result in

$$\underset{\ddot{q}}{\text{minimize}} \quad \left(\dot{A}\dot{q} + A\ddot{q} - \dot{h}_{\mathrm{D}}\right)^T C_{\mathrm{m}} \left(\dot{A}\dot{q} + A\ddot{q} - \dot{h}_{\mathrm{D}}\right). \tag{4.24}$$

Reformulating this into the form of $\frac{1}{2}\ddot{q}^T H\ddot{q} + g^T\ddot{q} + c$ results in

$$\underset{\ddot{q}}{\text{minimize}} \quad \ddot{q}^T A^T C_{\dot{h}} A\ddot{q} + 2\left(\dot{q}^T\dot{A}^T - \dot{h}_{\mathrm{D}}\right) C_{\dot{h}} A\ddot{q} + \left(\dot{q}^T\dot{A}^T C_{\mathrm{m}}\dot{A}\dot{q} + \dot{h}^T C_{\mathrm{m}}\dot{h} - 2\dot{h}C_{\mathrm{m}}\dot{A}\dot{q}\right). \tag{4.25}$$

Another approach is to introduce slack variables. The slack variables are added to the tracking constraints and function as errors values. The tracking can them be achieved by minimizing these slack values. For the momentum rate, this results in the cost

$$\underset{\ddot{q}}{\text{minimize}} \quad s_{\mathrm{m}}^T C_{\mathrm{m}} s_{\mathrm{m}}, \tag{4.26}$$

$$\text{subject to:} \quad \dot{A}\dot{q} + A\ddot{q} = \dot{h}_{\mathrm{D}} + s_{\mathrm{m}} \tag{4.27}$$

where s_m is the slack variable for the momentum rate constraint.

While both approaches essentially give the same result, as s_{m} represents the same error value as used in the cost function 4.24, the approach of slack values creates a shorter cost function. Furthermore, the cost function of the slack variable approach consists of only a quadratic part, while the direct approach also contains linear and constant parts.

The slack variable approach is chosen for the controller design as this will create a clearer cost function. Applying slack variables to all tracking constraints results in the performance cost

$$C_{\mathrm{t}} = \underset{s_{\mathrm{m}}, s_{\mathrm{f}}, s_{\mathrm{a}}}{\text{minimize}} \quad s_{\mathrm{m}}^T C_{\mathrm{m}} s_{\mathrm{m}} + s_{\mathrm{f}}^T C_{\mathrm{f}} s_{\mathrm{f}} + s_{\mathrm{a}}^T C_{\mathrm{a}} s_{\mathrm{a}}, \tag{4.28}$$

where s_{m}, s_{f} and s_{a} are the tracking errors of the momentum rate, feet acceleration and angular acceleration of the chest. Matrices C_{m}, C_{f} and C_{a} are cost function weighting matrices. This cost function has a total of 8 decision variables, 3 momentum rate errors, 4 feet and 1 angular acceleration error.

4.4.3 QP formulation

The full QP can now be formulated by a combination of the cost equations and the constraints. The cost is a combination of equations 4.23 and 4.28. The constraints consist of the dynamics constraint 4.8, the tracking constraints 4.17, 4.19 and 4.20 with the added deviation variables and the contact constraints 4.10 and 4.16. With this the full QP can be formulated:

$$\underset{\ddot{q}, \rho, \tau, s_{\mathrm{m}}, s_{\mathrm{f}}, s_{\mathrm{a}}}{\text{minimize}} \quad \ddot{q}^T C_{\mathrm{v}}\ddot{q} + \rho^T C_\rho \rho + \tau^T C_\tau \tau + s_{\mathrm{m}}^T C_{\mathrm{m}} s_{\mathrm{m}} + s_{\mathrm{f}}^T C_{\mathrm{f}} s_{\mathrm{f}} + s_{\mathrm{a}}^T C_{\mathrm{a}} s_{\mathrm{a}}, \tag{4.29}$$

subject to:

$$\begin{aligned} M(q)\ddot{q} + C(q)\dot{q} + G(q) &= S_\tau \tau + J^T(q)\rho, \\ \dot{A}(q)\dot{q} + A(q)\ddot{q} &= \dot{h}_{\mathrm{D}} + s_{\mathrm{m}}, \\ J_{\mathrm{f}}(q)\ddot{q} + \dot{J}_{\mathrm{f}}(q)\dot{q} &= \ddot{p}_{\mathrm{D}} + s_{\mathrm{f}}, \\ S_{\mathrm{c}}\ddot{q} &= \alpha_D + s_{\mathrm{a}}, \\ d_{\mathrm{f}}\rho &= 0, \\ Q_{\mathrm{c}}\rho &\leq 0, \end{aligned} \tag{4.30}$$

where Q_c is given by

$$Q_c = \begin{pmatrix} \begin{pmatrix} -P_n \\ P_t - \mu P_n \\ -P_t - \mu P_n \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} -P_n \\ P_t - \mu P_n \\ -P_t - \mu P_n \end{pmatrix} \end{pmatrix}. \tag{4.31}$$

The full QP written in the standard form of $\frac{1}{2}x^T Hx + g^T x + c$ can be found in appendix B.

4.4.4 Bounds

In addition to the constraints, several limits have to be enforced on the decision variables. The torque limits on robotic systems are normally determined by the maximal torque that can be delivered by the motors in the joints. For the model used in the thesis they are determined in the analysis step. Grf limits are enforced to ensure the robot does not launch itself of the ground or does not put unnecessary strain on the contact points.

The joint acceleration limits are used to enforce the joint limits. An overview of the used joint limits can be found in table C.1 in the appendix. The joint limits are enforced by limiting the maximum acceleration allowed in the direction of the joint limits. First the maximum joint acceleration is set. Then for each controller time step the distance b_i required for each joint to reach zero velocity is calculated with

$$b_i = \frac{\dot{q}_i^2}{2\ddot{q}_{\max}}, \tag{4.32}$$

where $\ddot{q}_{\max}$ is the maximum joint acceleration. If the current joint position plus the breaking distance b_i is larger than the joint limit, the joint is forced into maximal deceleration. This results in the following limit for each joint

- If $q_i + b_i \geq q_m$ then $\ddot{q}_{i,l} = -\ddot{q}_{\max}$,
- Else $\ddot{q}_{i,l} = \ddot{q}_{\max}$

where $\ddot{q}_{i,l}$ is the limit for joint i.

The deviation variables are unbound since they are not directly connected to a physical property.

Chapter 5

Analysis method

This chapter will cover the analysis method used to analyze the control parameters. Section 5.1 details the method used to analyze the control parameters. Section 5.2 discusses the control parameters that will be analysed. Section 5.3 covers the variables which will be used for the analysis.

5.1 Approach

The robot should be capable of walking at a constant velocity and recovering from force impulses large enough to require at least one step to recover balance. During these two scenarios, the performance with different parameter values will be analyzed. Then the optimal parameters values will be determined and analyzes of the final optimal design will be performed.

Walking will be analyzed by creating a CoM trajectory starting the initial CoM position of the robot and moving in positive x-direction at a constant velocity of 0.5 m/s. The push recovery will be analyzed by creating a stationary CoM trajectory at the initial location of the robot's CoM. After 0.5 seconds, a 700N horizontal force will be applied on the CoM for 0.2 seconds, resulting in a 140 Ns force impulse. This force impulse is strong enough for the robot to require multiple steps to recover balance.

Because almost all control parameters are influenced by each other, a baseline model is required to analyze the impact of individual control parameters. The baseline is created by estimating values for the control parameters. These estimated values will then be tested by running the simulation and tuned till the design goals are met.

With this baseline, the individual control parameters can be analyzed by testing different values. For each parameter, a range of values will be created. For each value, a simulation will be run and the analysis variables will the plotted. From these plots, the ideal value will be determined.

As torque is often a limiting factor in exoskeleton systems, the torque limit is the first parameter to be analyzed. The minimal operating torque limit will be determined by determining the lowest torque limit where the robot with the baseline controller is still able to achieve the design goals and where increasing the limit does not drastically increase the performance. This torque limit will then be used as the baseline parameter value for the analysis of the other parameters. This allows the other control parameters to be analyzed at a lower torque, which will increase the effects of changing the values.

One important thing to note is that only the torque limit will be changed in the baseline controller after its analysis. The other analysis will be done with the same baseline. The results of one analysis thus do not influence the other variables.

5.2 Control parameters selection

The control parameters can be divided into three groups: the decision variable limits, the decision variable weights and the PD values of the MRP.

The decision variable limits control the maximum values for different physical properties, which are often determined by the design of the system rather than the controller. However, it is still relevant to analyze these in a simulated environment since this can give a indication of the design requirement of the robot. The joint acceleration limit is used to enforce the joint limits. The joint acceleration that can be achieved in limited by the configuration of the robot and the torque limits. Thus it is not relevant to analyze. It is assumed that the ground is able to handle the GRFs the robot wants to generate. This parameter is still analyzed to see how much GRF are actually required to still achieve the design goals. The torque limit is important and will thus be analyzed. The slack limits are virtual and thus do not have any physical significance. These limits will not be analyzed.

The cost function of the QP has six decision variables. The torque weight is set to 0 and will not be analyzed for reasons explained in section 4.4.2. The weight parameters for the other decision variables will be analyzed.

The MRP uses PD control to create tracking goals that execute the desired trajectories. While the PD values can be chosen by analyzing the step response, this does not take the interaction with the QP controller into account. Because of that, these values will also be analyzed. The derivative value is chosen such that the PD controller is critically dampened. As such, when the proportional parameter is changed, the derivative value will change automatically.

This will result in the following nine parameters that will be analyzed:

1. Torque limits
2. GRF limits
3. Joint acceleration weight
4. GRF weight
5. Momentum rate deviation weight
6. Feet acceleration deviation weight
7. Chest angulare acceleration weight
8. Momentum rate PD values
9. Feet acceleration PD values

5.3 Analysis variables

There are two factors to be analyzed: the tracking performance and the operational factors.

The controller has two goals: tracking a given CoM trajectory and remaining stable. The performance regarding the first goal can be analyzed by looking at the accuracy of the CoM tracking. To achieve the second goal, it required for the QP to be able to track the feet trajectories generator by the trajectory generator in the high-level controller. Furthermore, to maintain long-term stability, the controller attempts to keep the chest at a certain angle. The overall performance of the robot can thus be analyzed by looking at these three tracking goals. This is done by looking at the average error between the robot and the trajectories and the peak errors.

The operational factors consist of operational costs and effects. These are the joint torques, GRFs, and the average energy expenditure. The energy expenditure is related to the torque. The average expenditure over a whole simulation is given as

$$u^2 = \sum_{i=1}^{n_t} \left(\sum_{k=1}^{4} \left(\tau_{k,i}^2 \right) \right) \tag{5.1}$$

where k are the actuated joints and n_t are the number of time steps in the simulation. For the torques and GRFs, both the average and the peak value are analyzed.

Chapter 6

Analysis results and discussion

This section will cover and briefly discuss the results of the analysis. The analysis contains multiple steps where the steps are dependent on the results of the previous steps. As such, this section covers both the results and a brief discussion of the results. The baseline values can be found in appendix D. The walking simulations are done at a walking speed of 0.5 m/s. The push recovery is simulated with a force impulse of 140 Ns applied at the CoM.

Section 6.1 covers the torque analysis. Section 6.2 covers the baseline performance. The baseline performance is covered after the torque analysis as this analysis is done with the optimal torque from the torque analysis implemented. This order is chosen so a fair comparison can be made between the baseline performance and the final design performance. Section 6.3 covers the analysis all parameters except for the torque limit. Section 6.4 covers the performance of the final design.

An overview of the configuration of the robot at different time steps during the simulations is shown in figure 6.3.

6.1 Torque limit analysis

The results of the torque limit analysis can be found in appendix E. Tests show that torque limits under 200 Nm are not feasible for this controller. These torque values do not allow the controller to follow the feet trajectory, resulting in the robot toppling.

The performance graphs show that a higher torque limit increases the tracking accuracy for all trajectories of the robot. However, increasing the torque limit also increases the operation cost. The improvement gained from increasing the torque limit slows above 220 Nm. This is clearly illustrated by the average feet error, highlighted in figure 6.1.

As the torque limit is often a limiting factor of exoskeleton systems, the other parameters will be analysed with a baseline torque of 220 Nm. The graphs show that increasing the torque limit above 220 Nm does not have large effect.

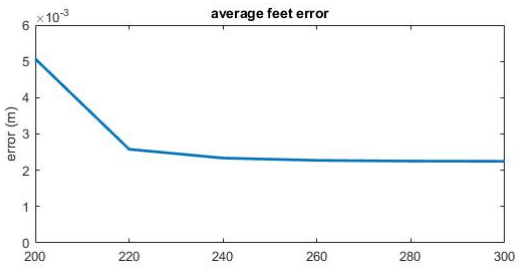
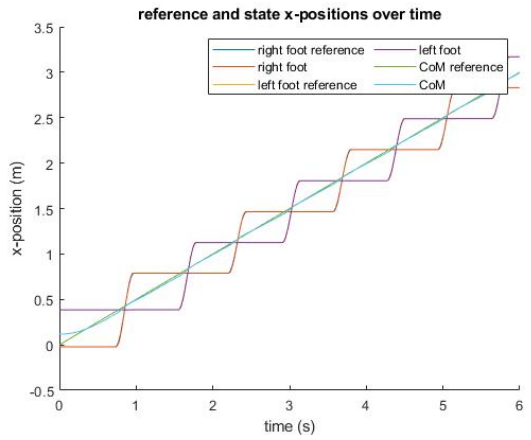
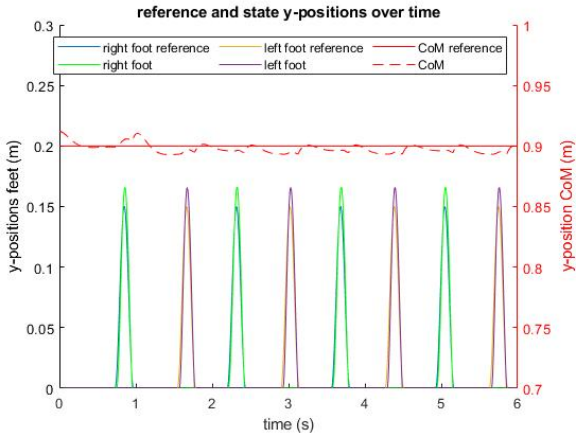


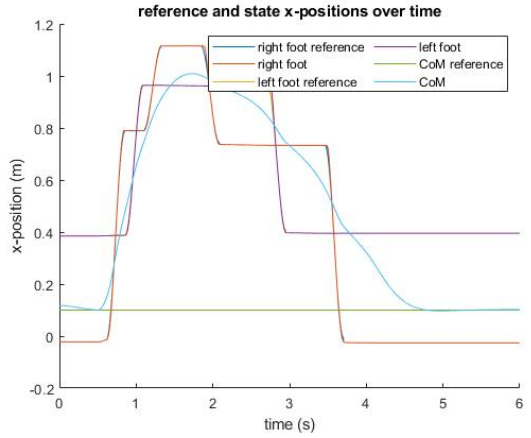
Figure 6.1: Average feet error for different torque limits. Increasing the torque limit improves the feet tracking. However, the improvement gained above 220 Nm is limited.



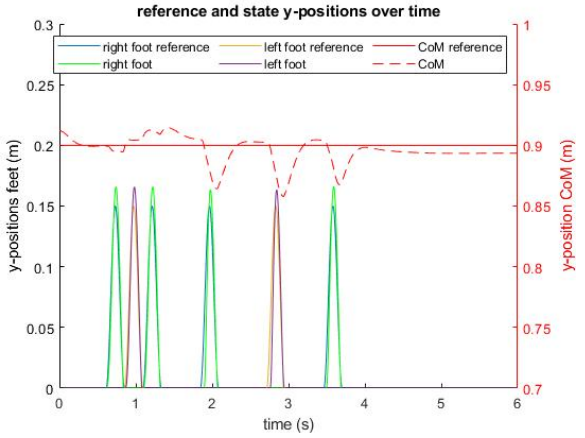
(a) Tracking performance in x direction during walking. A step is taken when the XcoM leaves the BoS.



(b) Tracking performance in y direction during walking. The y-positions of the feet use the left y-axis. The y-positions of the CoM use the right y-axis.



(c) Tracking performance in x direction during push recovery. A step is taken when the CoM leaves the BoS.



(d) Tracking performance in y direction during push recovery. The y-positions of the feet use the left y-axis. The y-positions of the CoM use the right y-axis.

Figure 6.2: Tracking performance of the baseline controller with a torque limit of 220 Nm.

6.2 Baseline performance

The baseline performance is analyzed with a torque limit of 220 Nm. Torque limit is a property determined by the physical design of a robotic system and often not a chosen control parameter. Furthermore, due to the impact of the torque limit, analysing the baseline model with a torque limit of 220 Nm will allow a fair comparison with the final design.

The tracking performances over time are displayed in figure 6.2. The values of the performance and operation variables are displayed in tables 6.1 and 6.2, for walking and push recovery respectively.

6.3 Parameter Analysis

The full results of the parameter analysis can be found in appendix F. The tests are performed with a torque limit of 220 Nm.

Table 6.1: The values of the performance and operational variables of the base line controller during walking.

Variable	**Average value**	**Peak value**
CoM error	0.012 m	0.027 m
Feet error	$2.6 \cdot 10^{-3}$ m	0.034 m
Chest angle devation	0.015 rad	0.058 rad
Joint torques	70.4 Nm	220 Nm
Ground reacion force	355.3 N	1024 N
u^2	$3.0 \cdot 10^4$ N^2m^2	$1.17 \cdot 10^5$ N^2m^2

Table 6.2: The values of the performance and operational variables of the base line controller during push recovery.

Variable	**Average value**	**Peak value**
CoM error	0.40 m	0.91 m
vertical CoM error	$8.3 \cdot 10^{-3}$ m	0.042 m
Feet error	$2.3 \cdot 10^{-3}$ m	0.79 m
Chest angle devation	0.019 rad	0.058 rad
Torque	60.6 Nm	220 Nm
Ground reacion force	359.1 N	1059 N
u^2	$2.8 \cdot 10^4$ N^2m^2	$1.3 \cdot 10^5$ N^2m^2

6.3.1 Ground reaction force limit

The graphs show that performance increases significantly for limits up to 750 N for normal walking. Limits lower than 750 are no feasible for the push response however. Values higher than 750 N increase the peak grf, however, do not increase the tracking performance of the controller.

6.3.2 Ground reaction force weight

For weight values lower than 10^{-1} changing the ground reaction force weight has no effect. However for values above 10^{-1} the controller becomes unstable. Moving requires a temporary increase in the grf. Increasing the cost of the grf results in the controller only starting to move once the error becomes too large. The feet trajectory tracking is then too weak to perform a stable gait, resulting in the toppling of the robot.

6.3.3 Joint Acceleration Weight

Changing the joint acceleration cost shows much of the same behaviour as the grf cost. For values up to 10, changing the cost does not have a noticeable impact. For value higher than 10, the feet trajectory tracking becomes weaker. This results in a higher average torque, which in turn increases the amount of energy used.

6.3.4 Momentum rate error weight

Momentum trajectory tracking is one of the primary tasks of the controller. The data shows that choosing a momentum rate error weight lower than 10 results in poor tracking. This leads to unstable behaviour. Increasing the weight to above 1000, will result in poor tracking of the feet. The tracking of the momentum rate is too strong, causing an unstable gait.

6.3.5 Feet acceleration error weight

The feet tracking become too weak when the feet acceleration error weight is decreased to 1000. Increasing the error weight improves the feet tracking and lowers the average torque and thus energy spending. However, due to the balancing effects in the cost, higher values of the error cost decreases the chest angular acceleration tracking.

Removing the error variable from the feet acceleration constraint, thus making it a full equality constraint, creates highly unstable behaviour. The robot cannot match the desired foot acceleration at the start of a step while maintaining proper tracking of the other tracking goals in order to stay stable.

6.3.6 Chest angular acceleration error weight

Increasing the chest angular acceleration error weight decreases the chest angle error while not influencing the feet and CoM errors for values up to 10^4. Values above 10^4 increase the angle tracking slightly but lowers the feet trajectory tracking and increases operation costs.

6.3.7 Momentum rate PD values

The PD controller of the MRP is critically dampened. The derivative gain changes when varying the proportional gain due to the following relationship: $D_{\mathrm{C}} = \sqrt{2mP_C}$, where m is the mass of the robot.

Increasing the CoM PD gains lowers the CoM error during walking, with deminishing returns for higher values. The feet error increases for higher CoM PD gains in what seems a linear relationship. In the case of being pushed the PD gains do not have a large effect. Since the momentum rate that can be generated by the robot, while remaining stable, is limited, there is a limit on the desired momentum rate that can be generated by the MRP. For high CoM errors, such as after a push, this limit is reached for most PD values.

Higher PD values increase the operational costs for walking but slightly lower the operational costs of when the robot is pushed.

6.3.8 Feet acceleration PD values

The feet PD values are also critically damped, resulting the following relation between proportional and derivative gain: $D_{\mathrm{f}} = \sqrt{2P_f}$. Increasing the feet PD values decreases the feet error, especially at lower values. For walking the increase also causes an increase in the CoM error. The operational costs decrease with increasing PD values.

6.4 Final Design performance

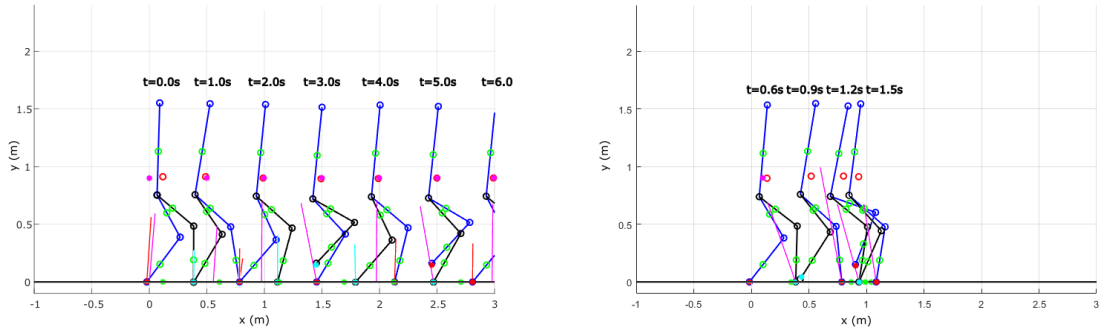
The control parameters of the final design can be found in table 6.3. The parameters were chosen based on the ideal values determined with the results of the analysis.

The results of the parameter analysis show that higher values of C_{f} and P_{f} increase the performance. However, both have mostly the same effect: making the feet trajectory tracking stronger. This causes the effects to stack and setting both parameters on their separate ideal values will cause an unstable gait. To prevent this, the chosen values for the final design are slightly less than the ideal values that came out of the analysis.

Figure 6.3 shows the position and configuration of the robot at different time steps in the simulations. The tracking performances over time are displayed in figure 6.4. The values of the performance and operational variables are shown in tables 6.4 and 6.5.

Table 6.3: The final design values of the controller parameters.

Variable name	**Value**
torque limit	220 Nm
grf limit	1000 N
C_{v}	$1.0 \cdot 10^{-2}$
C_{ρ}	$1.0 \cdot 10^{-4}$
C_{τ}	0
C_{m}	$1.0 \cdot 10^{1}$
C_{f}	$5.0 \cdot 10^{5}$
C_{a}	$1.0 \cdot 10^{5}$
P_{C}	1500
P_{f}	1500



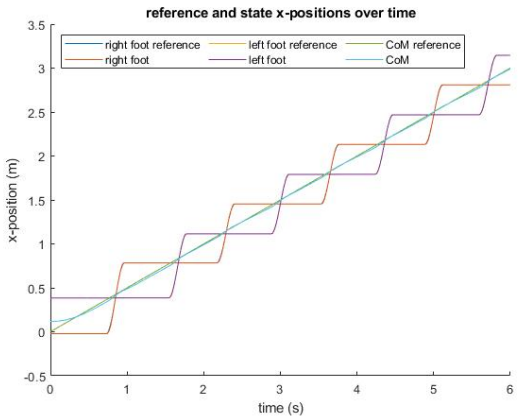
(a) Walking in positive x-direction at a constant velocity of 0.5 m/s.

(b) Push recovery. Force impulse is applied between 0.5 and 0.7 seconds.

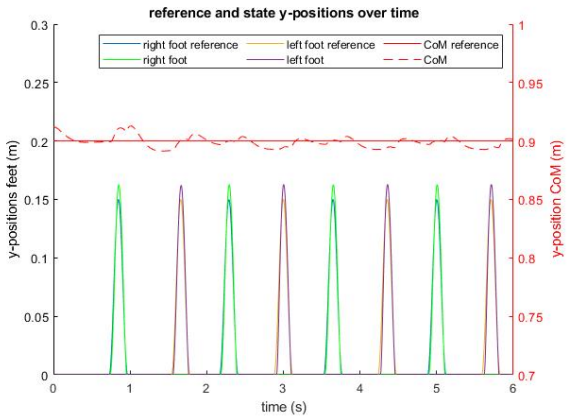
Figure 6.3: Position of the robot at different time steps during simulation using the final design of the controller. The red circle is the CoM and the purple asterisk the CoM reference. The red and cyan asterisks are the references of the right and left foot respectively. The red and cyan lines are grf at the right and left contact points, the magenta line is the net grf of all contact points. The green asterisk is the XcoM.

Table 6.4: The values of the performance and operational variables of the final design during walking

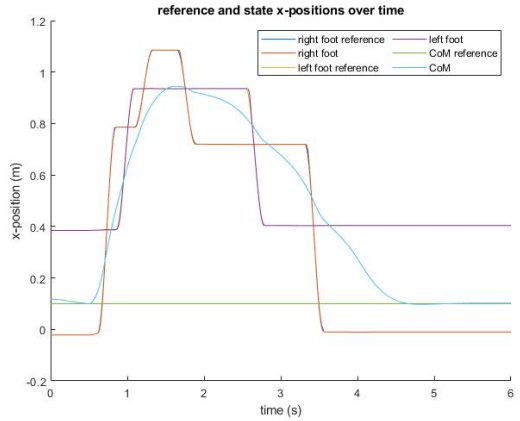
Variable	**Average value**	**Peak value**
CoM error	0.013 m	0.028 m
Feet error	$1.8 \cdot 10^{-3}$ m	0.034 m
Chest angle devation	0.017 rad	0.058 rad
Joint torques	69.6 Nm	220 Nm
Ground reacion force	356.1 N	1027 N
u^2	$3.0 \cdot 10^{4}$ $\mathrm{N^2m^2}$	$1.17 \cdot 10^{5}$ $\mathrm{N^2m^2}$



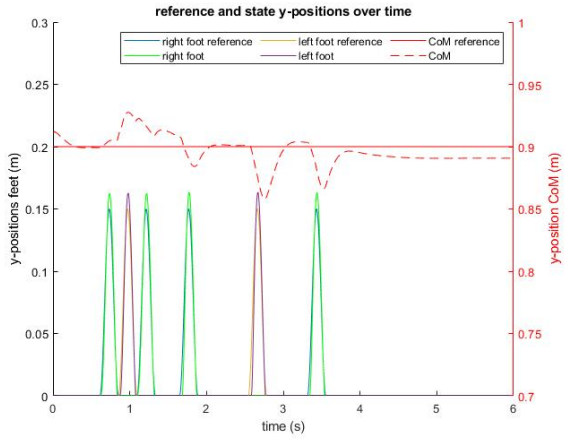
(a) Tracking performance in x direction during walking. A step is taken when the XcoM leaves the BoS.



(b) Tracking performance in y direction during walking. The y-positions of the feet use the left y-axis. The y-positions of the CoM use the right y-axis.



(c) Tracking performance in x direction during push recovery. A step is taken when the CoM leaves the BoS.



(d) Tracking performance in y direction during push recovery. The y-positions of the feet use the left y-axis. The y-positions of the CoM use the right y-axis.

Figure 6.4: Tracking performance of the final design.

Table 6.5: The values of the performance and operational variables of the final design during push recovery

Variable	**Average value**	**Peak value**
CoM error	0.36 m	0.85 m
vertical CoM error	$9.3 \cdot 10^{-3}$ m	0.028 m
Feet error	$1.5 \cdot 10^{-3}$ m	0.055 m
Chest angle devation	0.024 rad	0.058 rad
Torque	59.6 Nm	220 Nm
Ground reacion force	362.2 N	1051 N
u^2	$2.8 \cdot 10^4$ N^2m^2	$1.3 \cdot 10^5$ N^2m^2

Chapter 7

Discussion

7.1 Performance Final Design

The controller was made to achieve two goals: follow a given CoM trajectory and recover from a push with a force impulse large enough to require at least one step to recover balance. The final design is able to achieve both these goals.

7.1.1 High-level controller

The reference generator uses a reactive step planner based on the XcoM stability criterion. This means that a step will be taken when the XcoM leaves the BoS, with a delay based on the desired CoM velocity. This behavior is displayed clearly in figure 6.4a. This design allows the controller to respond to disturbances without requiring new CoM trajectories, as shown in figure 6.4c. One limitation of the state machine managing the step decisions, figure 4.2, is that a step always has to be completed before new feet trajectories are created. This makes the system unable to adjust to changing circumstances while stepping.

The motion reference planner is capable of translating the given CoM trajectory and the feet trajectories generated by the RG into tracking goals. The tracking goals achieve accurate tracking of the trajectories and can correct deviations from the trajectory. The design of the MRP allows the robot to move to a given CoM position without requiring a full CoM trajectory. Figure 6.4c shows that the robot is capable of moving back to the constant CoM target after balance recovery.

7.1.2 Quadratic program performance

The quadratic program has as goal to reconcile the tracking goals with the dynamics of the robot and find the optimal balance between the tracking goals and energy costs. The QP is capable of translating the tracking goals into accurate torques. While the tracking goals are met, it is not fully capable of balancing these goals with the energy consumption. Increasing the weight of the operational decision variables leads to unstable behavior. Whether this instability is caused by the QP itself or the trajectories generated by the high-level controller cannot be concluded with the results of the analysis.

7.1.3 Walking performance

While walking at a speed of 0.5 m/s, the robot is capable of tracking the CoM trajectory with an average error of 0.013 meters and a peak error of 0.028 meters. The CoM error data is taken after 0.4 seconds into the simulation. This is the time the robot needs to correct the error caused by the initial CoM position of the robot not being the same as the desired CoM position. Including the first 0.4 seconds would thus give inaccurate data.

The feet trajectory can be tracked with an average error of $1.8 \cdot 10^{-3}$ meters and a peak error of 0.034 meters. The peak errors occur at the top of the feet trajectory arc. The likely reason for this is that not enough torque can be generated to achieve the desired feet acceleration.

The robot is capable of walking with a stable, reliable gait while tracking the CoM trajectory. In its current implementation, the robot is capable of walking at a maximum velocity of 0.60 m/s or 2.16 km/h. Increasing the torque limit to 300 Nm allows the robot to reach stable velocities up to 1.25 m/s or 4.50 km/h. Suggesting that the robot could achieve normal walking velocities with a better high-level controller.

7.1.4 Push recovery

The robot is able to recover from a force impulse large enough to require multiple steps.

The robot will automatically step when necessary to remain stable. This allows the robot to maintain balanced when disturbed without requiring an adjusted CoM trajectory.

The robot is capable of recovering from horizontal push disturbance impulses up to 140 Ns. The robot recovers balance after 0.85 m. After recovering balance, the robot moves back to the desired CoM position. During the simulation, the peak vertical CoM error is 0.28m with an average error of $9.3 \cdot 10^{-3}$. The total recovery time is 4.2 seconds.

7.1.5 Improvements on Baseline

The final design parameter values are determined by the analysis done with the base line model. The analysis showed that most parameters were already near their optimal values. The parameters that saw significant change where the parameters related to the feet tracking, C_f from $1.0 \cdot 10^5$ to $5.0 \cdot 10^5$ and P_f from 1000 to 1500, and the GRFs, C_ρ from $1.0 \cdot 10^{-4}$ to $1.0 \cdot 10^{-6}$.

During walking, the only performance improves were seen in the feet trajectory tracking, where the average error decreased by 31%. No other improvements were made for the walking simulation.

For the push recovery simulation, more improvements were gained. Similar to the walking simulation, the feet error got reduced by 35%. Furthermore, the peak vertical CoM error got reduced by 33% and the distance to recover was reduced by 0.06m.

7.2 Design Limitations

7.2.1 Movement pattern

The gait generated by the high-level controller is purely reactionary, only initiating a step when it is required to maintain balance. This system allows the robot to walk and recover from disturbances. Since it is independent of the desired CoM position, it does not require the CoM trajectory to adjust to disturbances. However, there are a couple of limitations to this implementation.

The first major limitation is that a reactionary feet trajectory generator does not create a fluent gait. The double support phase is long, so it requires short step times. During walking, the gait consists of about 35%, while normal human gait consists of about 80% single support phase [20]. This means that the average feet velocity has to be about 2.3 times during a step then during normal human gait to achieve the same walking speed. The steps thus require high feet acceleration changes, which require high torques. Furthermore, the high accelerations of the stepping leg during the single support phase generate a momentum rate on the CoM. This causes the CoM to deviate from its trajectory which will have to be compensated for during the double support phase.

The second limitation of the reactionary step planner is that it requires a large range of motion. The system reacts to the XcoM and does not take the position of the CoM into

account. In some situations, this can result in the desired foot position moving outside of the range of motion of the foot. In these cases, the controller will try to move the foot as close to the desired position as possible. However, this can result in undesired or unstable behavior. One example is the feet hitting the ground, which can cause unexpected ground reaction forces that cause the robot to become unstable. To counter this, the robot has a crouched gait, which increases the horizontal range of motion of the feet. This is not desired since it requires higher knee torque and thus results in higher than necessary energy requirements.

7.2.2 Point feet

The contact points of the robot are modeled as point feet, resulting in only two contact points. While this simplifies the dynamics, it has a couple of limitations. The absence of feet and ankle torques limits the step planner. During normal gait, a lot of momentum can be generated by the feet pushing off the ground. Since these are absent in the model, all ground reaction forces have to be generated by the knee and hip joints. This results in a higher torque requirement since more force has to be generated by these joints.

Furthermore, the absence of the feet decreases the range of motion of the contact points relative to the CoM. To generate momentum, a bipedal system has to push against the supporting surface. This requires the limb to extend. With feet, this extension can be created by extending the ankle as well as the knee and hip. For a model with point feet, this extension has to come solely from knee and hip extension. This requires the knees and hips to not be fully extended while stationary, otherwise, the momentum required for walking cannot be generated. This crouched stance requires more torque and thus increases the energy expenditure.

Another limitation is that during the single support phase, there is only one contact point with the ground. This fixes the CoP on the same contact point, while a model with feet is capable of moving the CoP within the area of the foot. Since the CoP is fixed, the direction of the momentum cannot be altered once a step is initiated.

7.3 Analysis limitations

The controllers of exoskeletons and humanoid robots are complicated systems. They often involve multiple smaller controllers working together. This results in a lot of different control parameters. The effects of individual parameters on the final performance are not immediately clear.

The goal of the parameter analysis was to analyze the effects of individual parameters. This was done by creating a baseline model from which the effects of changing different parameters were analyzed.

The main limitation of the analysis is the interdependence of parameters. Most control parameters influence each other. Changing the value of one parameter can have a different effect depending on the value of another parameter. To fully analyze the effect of individual parameters, combinations of different values of interdependent variables have to be analyzed. However, the number of control parameters involved does not make this feasible.

Another limitation that follows from the previous one is the dependence on the baseline. The influence of individual parameters was analyzed by choosing different values from the baseline. A result that may happen is that changing the value of the parameter does not seem to have a large effect, while the effect may be larger if different baseline values were chosen for the other control parameters.

7.4 Quadratic Program

When it comes to the QP there are two important choices made with regards to the cost function. The first is that all tracking goals have been added to the cost. The second choice is to implement the cost with the use of slack values in the constraints.

The constraint equations in a QP have to be met. As such, if the tracking equations are used as constraint equations, the tracking will very strong when feasible. The main downside of this method is its feasibility. Using the feet tracking as an example, if the feet acceleration goal cannot be met, the QP will not be solvable. This will cause the robot to crash or a backup controller has to be designed to deal with these cases. Including the tracking into the cost allows for errors when necessary.

The second decision was to implement the cost on the tracking goals via slack variables. The use of slack simplifies the cost function since it only contains quadratic parts. Furthermore, the output of the cost function now contains the slack variables, which are the errors on the tracking goals. This will a more intuitive insight into the decisions made by the QP. The downside of using slack variables is an increase in the number of constraint equations.

The simulations have shown that both using slack variables and applying a cost to all tracking goals results in a controller with strong trajectory tracking. The added flexibility of adding the feet tracking to the cost function allows the system to deviate from infeasible changes in feet acceleration. The analysis shows that removing the cost from the feet acceleration, making it a full equality constraint, leads to unstable behavior. It is unknown whether the use of slack variables over the direct implementation of the tracking in the cost leads to any differences in the computation time.

7.5 Applicability to Exoskeletons

The goal of this thesis was to explore the option of using a momentum-based controller for the Symbitron exoskeleton. This was done by approaching the exoskeleton with a two-dimensional humanoid (2DM) robot. Using a 2DM robot as a platform creates a system that has fewer variables than an exoskeleton. This makes it simpler to test different controller variations and analyze the effect of different controller parameters. However, this does also brings with it some limitations to the applicability of the controller to an exoskeleton.

An important aspect of humanoid control is task priority. The main focus of a humanoid system is balance. The requirements for balance are mostly the same for any bipedal robot. Furthermore, tracking goals for a 2DM robot are the same as is used for exoskeleton control. Because of this, the relative cost between performance decision variables concluded by the analysis should apply to an exoskeleton.

The high-level controller was built to provide a simple balance system and tracking goal generator. However, this controller has a couple of limitations as discussed in section 7.2.1. These limitations make it undesirable to apply this high-level controller on an exoskeleton. The balance criterion applies to an exoskeleton, however, the feet trajectory should be more sophisticated and generate a smoother gait.

The quadratic program has shown to be capable at achieving the tracking goals. The cost function and constraints should be applicable to an exoskeleton when expanded to three dimensions.

The model uses point feet. While this is sufficient for a 2DM, feet will have to be added in order to expand it to a three-dimensional model. The complexity of the feet lies mostly in the additional contact points. These contact points can be added to the feet tracking constraint, as these equations track the Cartesian accelerations of the contact points. If these are added, the QP should be able to function in the same form as for the 2DM.

7.6 Contribution to the field

While humanoid robotic control is a complex field, this thesis attempted to contribute to the understanding of these control systems. Using a momentum-based controller for humanoid robots in itself is not a novel design. It has been applied by Koolen et al. in 2013 and 2016 [28, 33], for example. However, this thesis does give a couple of new insights.

To my knowledge, the control systems used so far put the feet tracking as an equality constraint. The analysis of the controller designed in this thesis shows that this is not required to achieve a stable gait and adding it to the cost allows the system to be more flexible and more robust.

Furthermore, the analysis performed on the control parameters gives a more detailed insight into the effects of these parameters on achieving the tracking goals. While the parameter values of other momentum-based controllers will have been chosen based on a form of analysis. However, to my knowledge, no analysis has been published.

Chapter 8

Recommendations

8.1 Recommended controller settings

The final controller parameters can be found in table 6.3. While these parameters are specific for this design, they give indications of recommended controller settings for other bipedal systems.

8.1.1 Overall quadratic program design

Regarding the overall QP design, it is recommended to include all tracking goals in the cost function. The results show including the tracking functions in the cost allows for strong trajectory tracking while still allowing for some errors when necessary. These errors can be preferred or even necessary when sudden changes in the reference momenta rates or acceleration occur, for example when the movement is disturbed. A further benefit is that it is no longer required for the reference momenta rates and accelerations to be smooth. If the change in desired acceleration is too steep, the controller is able to approach it as best as possible without the QP becoming infeasible.

8.1.2 Cost function

Since the QP balances the costs of the different decision variables in the cost, increasing the weight of one variable has the same result as lowering the others. As such, recommendations for the cost function weights can only be given in the form of which variables to prioritize and the cost of certain variables relative to others. The results of the parameter analysis give a clear indication of the recommended prioritization in the cost function.

Regarding the performance variables, the first tracking priority should be the feet. The experiments have shown that when using lower values for the feet acceleration error weight makes the QP choose to use the legs to compensate for momentum rate errors. While this might be very efficient in the short term, it is not a feasible long-term strategy and leads to instability. The second priority should be the chest angle. When low priority is given to the chest angle, the QP decides to create momentum by leaning forward or backward, since this requires less torque and ground reaction forces. This often results in the robot toppling. The last prioritization should be the momentum rate. While the input of the controller is only a CoM trajectory, the overall momentum rate is less important than maintaining balance. The analysis indicates that if a stable gait is achieved, accurate CoM tracking can also be achieved.

For the operational cost parameters, a strong recommendation cannot be given. The movement pattern created by the high-level controller requires quick steps. This results in a requirement for high joint acceleration and torque during the stepping arc. Increasing the weight of the operational weight thus cause instability. A controller with a more fluent movement pattern might have different recommended settings for the operational weights.

8.2 Future Work

In order to develop this controller further, a recommendation can be given on which steps to take before implementing it for a exoskeleton. It is first recommended to expand the humanoid model with feet. Adding feet in a two-dimensional model will increase the amount of contact points from two to four. A model with feet has a larger range of motion, which should lead to more robust movement and a smoother gait. Furthermore, feet are required to achieve a stable three dimensional model as point feet do not give a proper BoS to maintain balance.

The high-level controller used in this thesis is very basic and likely won't perform well in a three dimensional setting. A controller that provides a smoother gait pattern will decrease the required torque limit and bring it more in line with the capabilities of motors used in exoskeleton design. Assuming the human gait is near optimal, a gait with a longer single support phase is recommended.

Bibliography

[1] W. H. Organization, "Spinal cord injury," 2013. Accessed on: 01/11/2019.

[2] A. K. Varma, A. Das, G. Wallace, J. Barry, A. A. Vertegel, S. K. Ray, and N. L. Banik, "Spinal cord injury: A review of current therapy, future treatments, and basic science frontiers," *Neurochemical Research*, vol. 38, pp. 895–905, May 2013.

[3] A. Esquenazi, M. Talaty, and A. Jayaraman, "Powered exoskeletons for walking assistance in persons with central nervous system injuries: A narrative review," *PMR*, vol. 9, pp. 46–62, 1 2017.

[4] A. J. Young and D. P. Ferris, "State of the art and future directions for lower limb robotic exoskeletons," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 2, pp. 171–182, 2017.

[5] K. Anam and A. A. Al-Jumaily, "Active exoskeleton control systems: State of the art," *Procedia Engineering*, vol. 41, pp. 988 – 994, 2012. International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012).

[6] S. Wang, L. Wang, C. Meijneke, E. van Asseldonk, T. Hoellinger, G. Cheron, Y. Ivanenko, V. La Scaleia, F. Sylos-Labini, M. Molinari, F. Tamburella, I. Pisotta, F. Thorsteinsson, M. Ilzkovitz, J. Gancet, Y. Nevatia, R. Hauffe, F. Zanow, and H. van der Kooij, "Design and control of the mindwalker exoskeleton," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 2, pp. 277–286, 2015.

[7] A. J. Young and D. P. Ferris, "State of the art and future directions for lower limb robotic exoskeletons," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 2, pp. 171–182, 2017.

[8] N. Aliman, R. Ramli, and S. Haris, "Design and development of lower limb exoskeletons," *Robot. Auton. Syst.*, vol. 95, p. 102–116, Sept. 2017.

[9] S. Jatsun, A. Malchikov, and A. Yatsun, "Comparative analysis of the industrial exoskeleton control systems," in *Proceedings of 14th International Conference on Electromechanics and Robotics "Zavalishin's Readings"* (A. Ronzhin and V. Shishlakov, eds.), (Singapore), pp. 63–74, Springer Singapore, 2020.

[10] M. Mistry, J. Nakanishi, G. Cheng, and S. Schaal, "Inverse kinematics with floating base and constraints for full body humanoid robot control," in *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, pp. 22–27, 2008.

[11] J. Vantilt, K. Tanghe, M. Afschrift, A. Bruijnes, K. Junius, J. Geeroms, E. Aertbeliën, F. De Groote, D. Lefeber, I. Jonkers, and J. Schutter, "Model-based control for exoskeletons with series elastic actuators evaluated on sit-to-stand movements," *Journal of NeuroEngineering and Rehabilitation*, vol. 16, 06 2019.

[12] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous robots*, vol. 35, no. 2, pp. 161–176, 2013.

[13] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 295–302, 2014.

[14] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Resolved momentum control: humanoid motion planning based on the linear and angular momentum," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 2, pp. 1644–1650 vol.2, 2003.

[15] C. Meijneke, G. van Oort, V. Sluiter, E. van Asseldonk, N. L. Tagliamonte, F. Tamburella, I. Pisotta, M. Masciullo, M. Arquilla, M. Molinari, A. R. Wu, F. Dzeladini, A. J. Ijspeert, and H. van der Kooij, "Symbitron exoskeleton: Design, control, and evaluation of a modular exoskeleton for incomplete and complete spinal cord injured individuals," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, pp. 1–1, 2021.

[16] L. Lanari and S. Hutchinson, "Inversion-based gait generation for humanoid robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1592–1598, 2015.

[17] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, 2016.

[18] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[19] C. G. Atkeson, B. P. W. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, M. Gennert, J. P. Graff, P. He, A. Jaeger, J. Kim, K. Knoedler, L. Li, C. Liu, X. Long, T. Padir, F. Polido, G. G. Tighe, and X. Xinjilefu, "No falls, no resets: Reliable humanoid behavior in the darpa robotics challenge," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 623–630, 2015.

[20] H. F. N. Al-Shuka, B. Corves, W.-H. Zhu, and B. Vanderborght, "Multi-level control of zero-moment point-based humanoid biped robots: a review," *Robotica*, vol. 34, no. 11, p. 2440–2466, 2016.

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

[22] H. F. N. Al-Shuka, F. Allmendinger, B. Corves, and W.-H. Zhu, "Modeling, stability and walking pattern generators of biped robots: a review," *Robotica*, vol. 32, no. 6, p. 907–934, 2014.

[23] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *Int. J. Rob. Res.*, vol. 31, p. 1094–1113, Aug. 2012.

[24] L. Lanari and S. Hutchinson, "Planning desired center of mass and zero moment point trajectories for bipedal locomotion," pp. 637–642, 11 2015.

[25] A. Hof, M. Gazendam, and W. Sinke, "The condition for dynamic stability," *Journal of Biomechanics*, vol. 38, no. 1, pp. 1–8, 2005.

[26] M. Vukobratovic and B. Borovac, "Zero-moment point - thirty five years of its life.," *I. J. Humanoid Robotics*, vol. 1, pp. 157–173, 03 2004.

[27] A. L. Hof, "The 'extrapolated center of mass' concept suggests a simple control of balance in walking," *Human Movement Science*, vol. 27, no. 1, pp. 112–125, 2008.

[28] T. Koolen, S. Bertrand, G. Thomas, T. de Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt, "Design of a momentum-based control framework and application to the humanoid robot atlas," *International Journal of Humanoid Robotics*, vol. 13, no. 01, p. 1650007, 2016.

[29] A. R. Wu and A. D. Kuo, "Determinants of preferred ground clearance during swing phase of human walking," *Journal of Experimental Biology*, vol. 219, no. 19, pp. 3106–3113, 2016.

[30] G. Meinsma, *Optimal Control*. University of Twente, November 2019.

[31] R. Orsolino, M. Focchi, C. Mastalli, H. Dai, D. G. Caldwell, and C. Semini, "Application of wrench-based feasibility analysis to the online trajectory optimization of legged robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3363–3370, 2018.

[32] A. Macchietto, V. Zordan, and C. R. Shelton, "Momentum control for balance," in *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, (New York, NY, USA), Association for Computing Machinery, 2009.

[33] T. Koolen, J. Smith, G. Thomas, S. Bertrand, J. Carff, N. Mertins, D. Stephen, P. Abeles, J. Englsberger, S. McCrory, J. van Egmond, M. Griffioen, M. Floyd, S. Kobus, N. Manor, S. Alsheikh, D. Duran, L. Bunch, E. Morphis, L. Colasanto, K.-L. H. Hoang, B. Layton, P. Neuhaus, M. Johnson, and J. Pratt, "Summary of team ihmc's virtual robotics challenge entry."

[34] A. F. Soliman and B. Ugurlu, "Robust locomotion control of a self-balancing and underactuated bipedal exoskeleton: task prioritization and feedback control," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5626–5633, 2021.

[35] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[36] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.

Appendix A

Contact model

This work follows [36], but is adapted from the 3D case to the 2D sagittal case. The code was implemented by Ander Vallinas-Prieto and Arvid Keemink.

Given our dynamics:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = S_\tau \tau + J^T(q)\rho,$$

we first rewrite $h(q,\dot{q}) = C(q,\dot{q})\dot{q} + G(q)$ so that we obtain:

$$M(q)\ddot{q} = S_\tau \tau - h(q,\dot{q}) + J^T \rho$$

Using Euler integration with time step Δt, denoting the contact impulse that happens over this duration as $\lambda = \Delta t \cdot \rho$ and omitting dependence on q, we obtain:

$$\begin{aligned} M(\dot{q}^+ - \dot{q}^-) &= \Delta t(S_\tau \tau - h) + J^T \lambda \\ M\dot{q}^+ &= M\dot{q}^- + \Delta t(S_\tau \tau - h) + J^T \lambda \\ \dot{q}^+ &= \dot{q}^- + M^{-1}\left(\Delta t(S_\tau \tau - h) + J^T \lambda\right) \end{aligned}$$

where $\dot{q}^+$ is the generalized velocity for the next time step. The goal is to find an impulse that will bring the velocity of the contact-end-effector to zero.

For each contact point $i = 1, 2$ we have contact-point Jacobian J_i and can transform these EoM to those frames:

$$\begin{aligned} J_i\dot{q}^+ &= J_i\dot{q}^- + J_i M^{-1}\left(\Delta t(S_\tau \tau - h) + \sum_{k=1}^{2} J_k^T \lambda_k\right) \\ \dot{v}_i^+ &= \tau_i^* + J_i M^{-1} J_i^T \lambda_i + J_i M^{-1} \sum_{k=1, k\neq i}^{2} J_k^T \lambda_k, \end{aligned}$$

where $\tau_i^* = J_i\dot{q}^- + J_i M^{-1}\Delta t(S_\tau \tau - h)$. Then if we define[1]:

$$c_i = \tau_i^* + J_i M^{-1} \sum_{k=1, k\neq i}^{2} J_k^T \lambda_k,$$

we would have $v_i^+ = c_i + J_i M^{-1} J_i^T \lambda_i$ and $\lambda_{v0} = -\left(J_i M^{-1} J_i^T\right)^{-1} c_i$, where λ_{v0} is the impulse needed to stay obtain zero velocity of the contact point after impact.

The following steps we iterate until convergence, starting with $\lambda_i = 0$, if we have more than one contact point:

[1] Using the sum is superfluous when having only 2 contact points, but its structure shows a direct generalization for an arbitrary number of contact points

- If the contact is opening ($c_{i,y} > 0$) then $\lambda_i = \lambda_i(1-\alpha)$.
- Else if we are stationary ($\mu\lambda_{v0,y} \geq |\lambda_{v0,x}|$) then $\lambda_i = \alpha\lambda_{v0} + (1-\alpha)\lambda_i$
- Else if we are slipping ($\mu\lambda_{v0,y} < |\lambda_{v0,x}|$) then we find the impulse λ_i^* by solving a QP

$$\lambda_i^* = \arg\min_{\lambda_i} \lambda_i^T J_i M^{-1} J_i^T \lambda_i - + c_i^T \lambda_i$$

s.t.

conform to friction cone

$$\begin{bmatrix} 0 & 1 \end{bmatrix} J_i M^{-1} J_i^T \lambda_i = -c_{i,y}$$

and $\lambda_i = \alpha\lambda_i^* + (1-\alpha)\lambda_i$.

For a single contact point, $\alpha = 1$ and we stop after a single iteration. We also would obtain the simplification that $c_i = \tau_i^*$.

The ground reaction forces (average over Δt) given by ρ are readily determined from this procedure:

$$\rho = \lambda / \Delta t$$

Appendix B

Final quadratic program in standard form

This appendix puts the QP formulated in equations 4.29 and 4.30 into the standard form.

The vector with decision variables is defined as

$$x = \begin{pmatrix} \ddot{q} \\ \rho \\ \tau \\ s_{\mathrm{m}} \\ s_{\mathrm{f}} \\ s_{\mathrm{a}} \end{pmatrix}, \tag{B.1}$$

where the individual variables are defined as

$$\ddot{q} = \begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \\ \ddot{q}_4 \\ \ddot{q}_5 \\ \ddot{q}_6 \\ \ddot{q}_7 \end{pmatrix}, \rho = \begin{pmatrix} f_{\mathrm{x,r}} \\ f_{\mathrm{y,r}} \\ f_{\mathrm{x,l}} \\ f_{\mathrm{y,l}} \end{pmatrix}, \tau = \begin{pmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{pmatrix}, s_{\mathrm{m}} = \begin{pmatrix} s_{\mathrm{m,a}} \\ s_{\mathrm{m,x}} \\ s_{\mathrm{m,y}} \end{pmatrix}, s_{\mathrm{f}} = \begin{pmatrix} s_{\mathrm{f,x,r}} \\ s_{\mathrm{f,y,r}} \\ s_{\mathrm{f,x,l}} \\ s_{\mathrm{f,y,l}} \end{pmatrix}, s_{\mathrm{a}} = s_{\mathrm{a}}, \tag{B.2}$$

where subscripts r and l represent the right and left foot respectively, subscript a represents angular. Adding the individual variables up results in a total of 23 decision variables.

The standard form of a quadratic program has the form of

$$\frac{1}{2} x^T H x + g^T x + c. \tag{B.3}$$

The cost contains only quadratic components, thus the g and c components are zero. H is a 23-by-23 symmetric matrix given by

$$H = \begin{pmatrix} C_{\mathrm{v}} & 0_{7\times4} & 0_{7\times4} & 0_{7\times3} & 0_{7\times4} & 0_{7\times1} \\ 0_{4\times7} & C_{\rho} & 0_{4\times4} & 0_{4\times3} & 0_{4\times4} & 0_{4\times1} \\ 0_{4\times7} & 0_{4\times4} & C_{\tau} & 0_{4\times3} & 0_{4\times4} & 0_{4\times1} \\ 0_{3\times7} & 0_{3\times7} & 0_{3\times7} & C_{\mathrm{m}} & 0_{3\times4} & 0_{3\times1} \\ 0_{4\times7} & 0_{4\times7} & 0_{4\times7} & 0_{4\times3} & C_{\mathrm{f}} & 0_{4\times1} \\ 0_{1\times7} & 0_{1\times7} & 0_{1\times7} & 0_{1\times3} & 0_{1\times4} & C_{\mathrm{a}} \end{pmatrix} \tag{B.4}$$

$$H = \begin{pmatrix} C_{\mathrm{v}} & & & & \\ & C_{\rho} & & 0 & \\ & & C_{\tau} & & \\ & & & C_{\mathrm{m}} & \\ & 0 & & C_{\mathrm{f}} & \\ & & & & C_{\mathrm{a}} \end{pmatrix} \tag{B.5}$$

The inequality constraints have the form of

$$A_{\mathrm{ineq}} x \leq b. \tag{B.6}$$

Matrix A and vector b are given by

$$A_{\mathrm{ineq}} = \begin{pmatrix} 0_{6\times 7} & Q_{\mathrm{c}} & 0_{6\times 12} \end{pmatrix}, b = \begin{pmatrix} 0_{6\times 1} \end{pmatrix}, \tag{B.7}$$

where Q_{c} is given by matrix 4.31.

The equality constraints have the form of

$$A_{\mathrm{eq}} x = b_{\mathrm{eq}}, \tag{B.8}$$

of which the matrices are given by

$$A_{\mathrm{eq}} = \begin{pmatrix} M(q) & -J^T(q) & -S_{\tau} & 0_{7\times 3} & 0_{7\times 4} & 0_{7\times 1} \\ A(q) & 0_{3\times 4} & 0_{3\times 4} & I_{3\times 3} & 0_{3\times 4} & 0_{3\times 1} \\ J(q) & 0_{4\times 4} & 0_{4\times 4} & 0_{4\times 3} & I_{4\times 4} & 0_{4\times 1} \\ S_{\mathrm{c}} & 0_{1\times 4} & 0_{1\times 4} & 0_{1\times 3} & 0_{1\times 4} & 1 \\ 0_{4\times 4} & d_{\mathrm{f}} & 0_{4\times 4} & 0_{4\times 3} & 0_{4\times 4} & 0_{4\times 1} \end{pmatrix}, b_{\mathrm{eq}} = \begin{pmatrix} -C(q)\dot{q} - G(q) \\ \dot{h}_{\mathrm{D}} - \dot{A}(q)\dot{q} \\ \ddot{p}_D - \dot{J}_{\mathrm{f}}(q)\dot{q} \\ \alpha_{\mathrm{D}} \\ 0_{4\times 1} \end{pmatrix}, \tag{B.9}$$

where d_{f} is the diagonal matrix

$$d_{\mathrm{f}} = \begin{pmatrix} d_{\mathrm{f,r}} & 0 & 0 & 0 \\ 0 & d_{\mathrm{f,r}} & 0 & 0 \\ 0 & 0 & d_{\mathrm{f,l}} & 0 \\ 0 & 0 & 0 & d_{\mathrm{f,l}} \end{pmatrix}. \tag{B.10}$$

Appendix C

Joint limits

Table C.1: The joint limits of the humanoid model. The first two joint are translation joints. The other joints are rotation joint.

Joint	**lower limit**	**upper limit**
q1	-∞ m	∞ m
q2	-∞ m	∞ m
q3	-175°	175°
q4	10°	175°
q5	-175°	175°
q6	-300°	-60°
q7	-175°	-10°

Appendix D

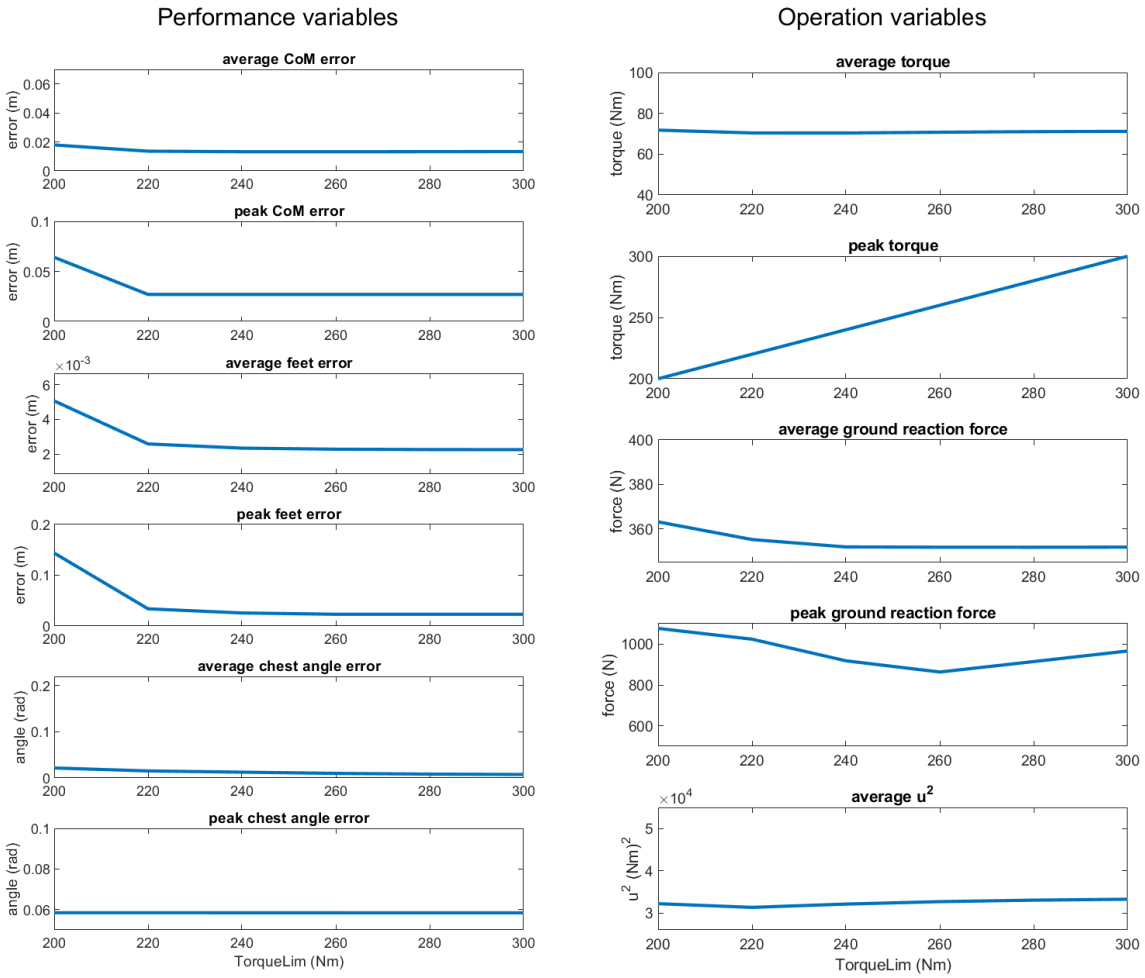
Base line values

Table D.1: The baseline values of the controller parameters used in the parameter analysis.

Variable name	**Value**
torque limit	500 Nm
grf limit	1000 N
C_{v}	$1.0 \cdot 10^{-2}$
C_{ρ}	$1.0 \cdot 10^{-6}$
C_{τ}	0
C_{m}	10
C_{f}	$1.0 \cdot 10^{5}$
C_{a}	$1.0 \cdot 10^{5}$
P_{C}	1500
P_{f}	1000

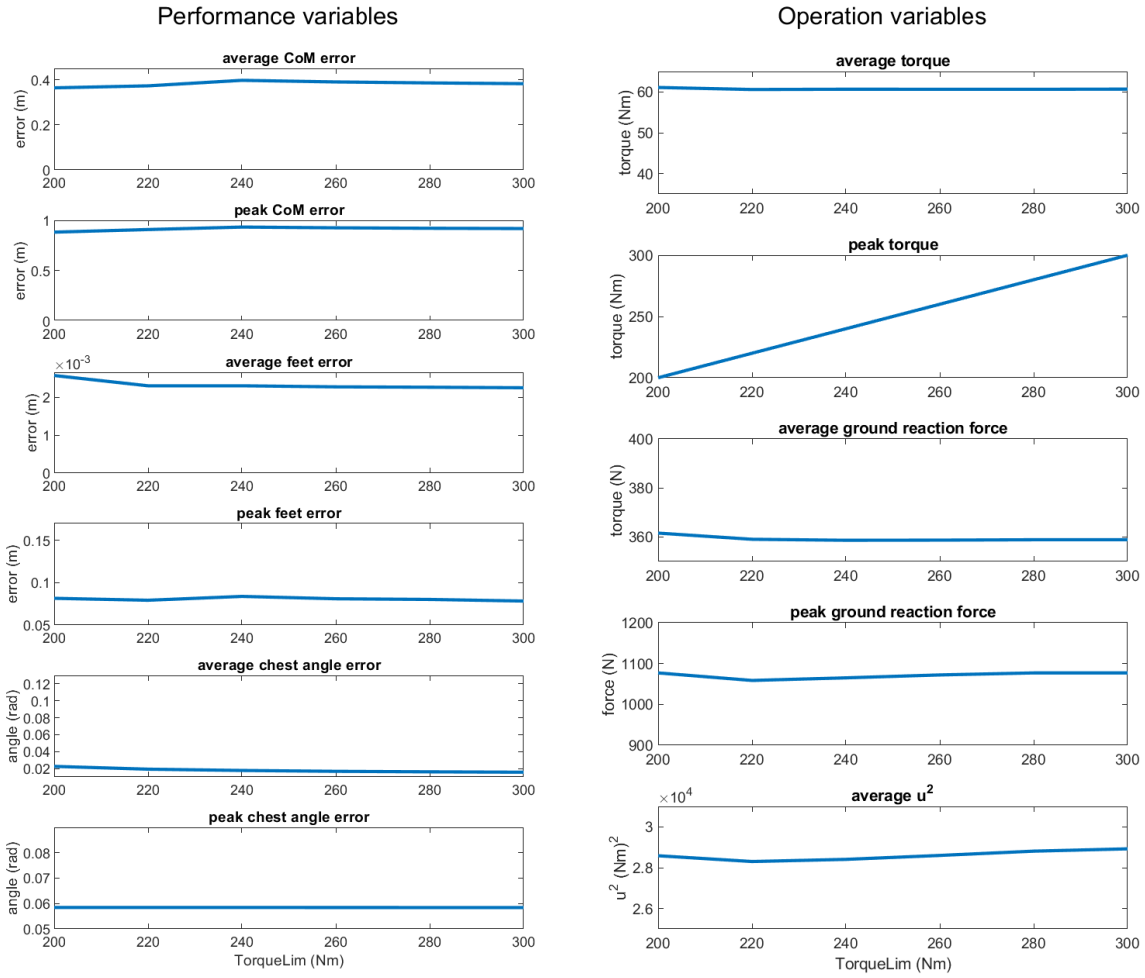
Appendix E

Torque limit analysis data



(a) (b)

Figure E.1: Walking analysis of different torque limit values. Values under 200 Nm have shown to be infeasible with this controller.



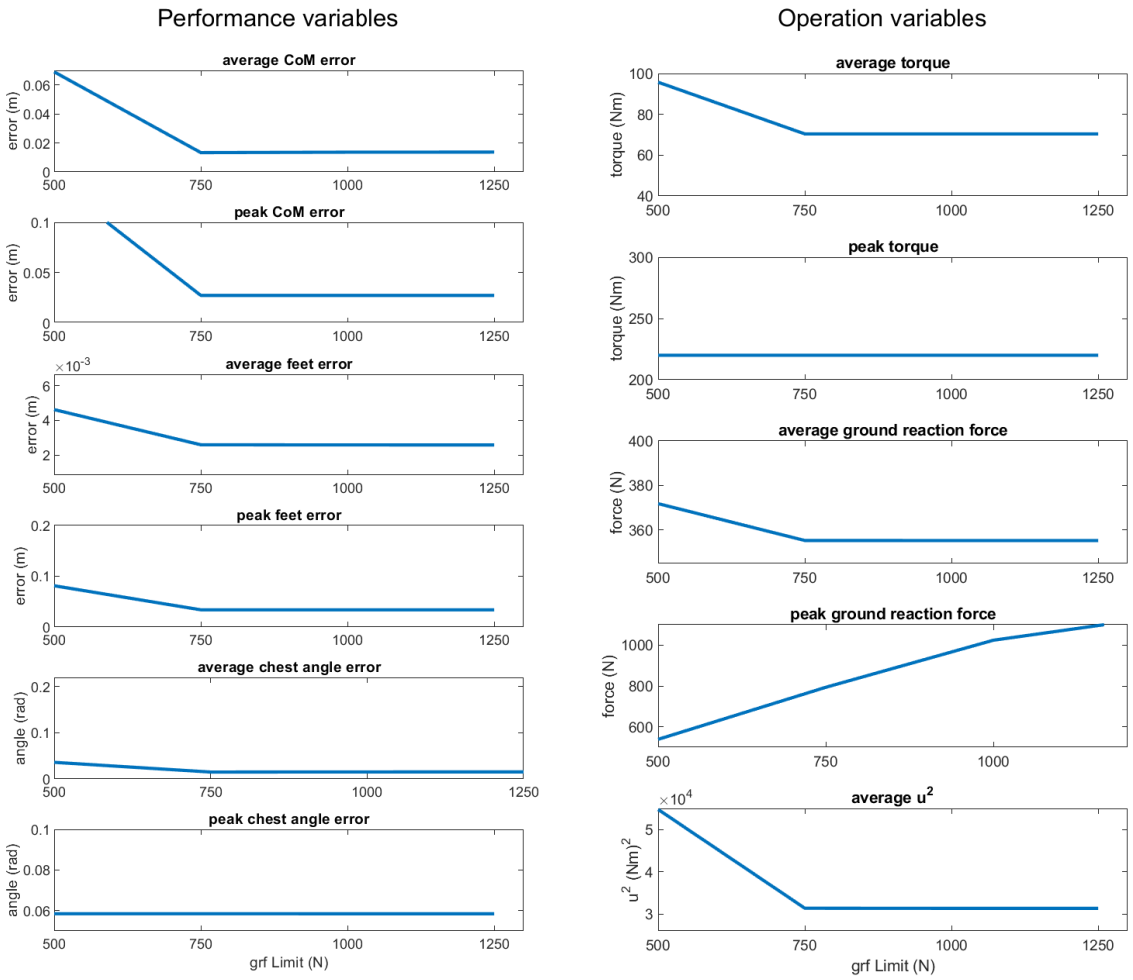
(a) (b)

Figure E.2: Push disturbance analysis of different torque limit values. Values under 200 Nm have shown to be infeasible with this controller.

Appendix F

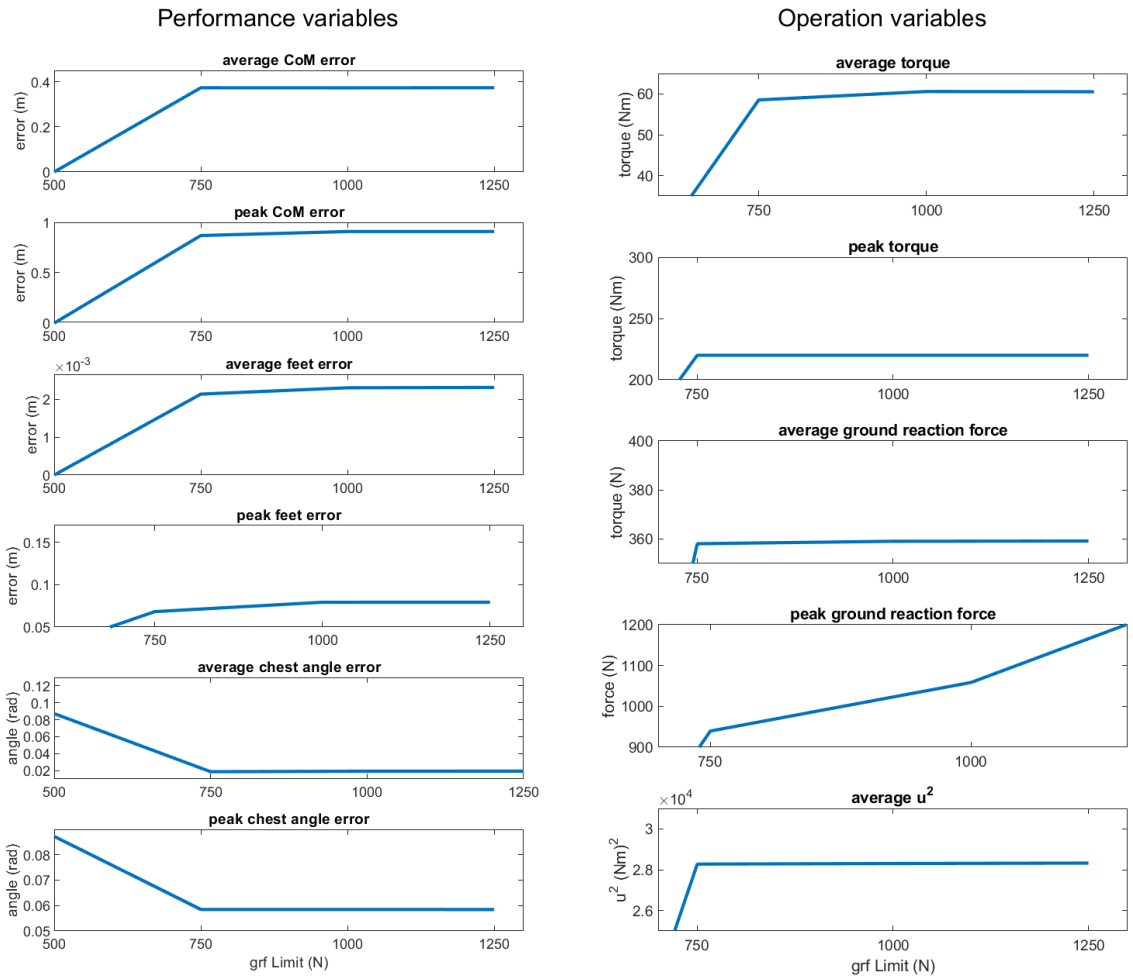
Parameter analysis data

F.1 Ground reaction forces limits



(a) (b)

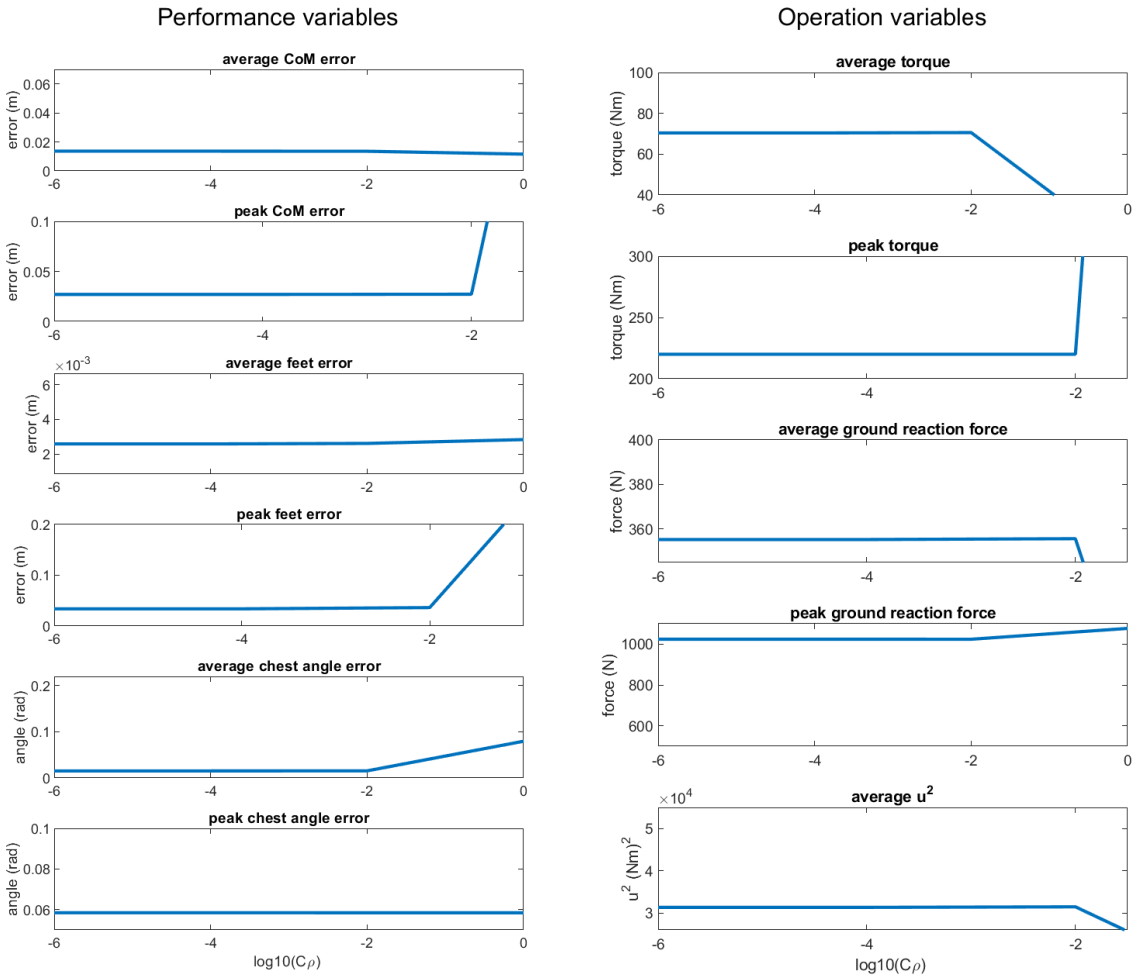
Figure F.1: Walking analysis of different ground reaction force limit values



(a) (b)

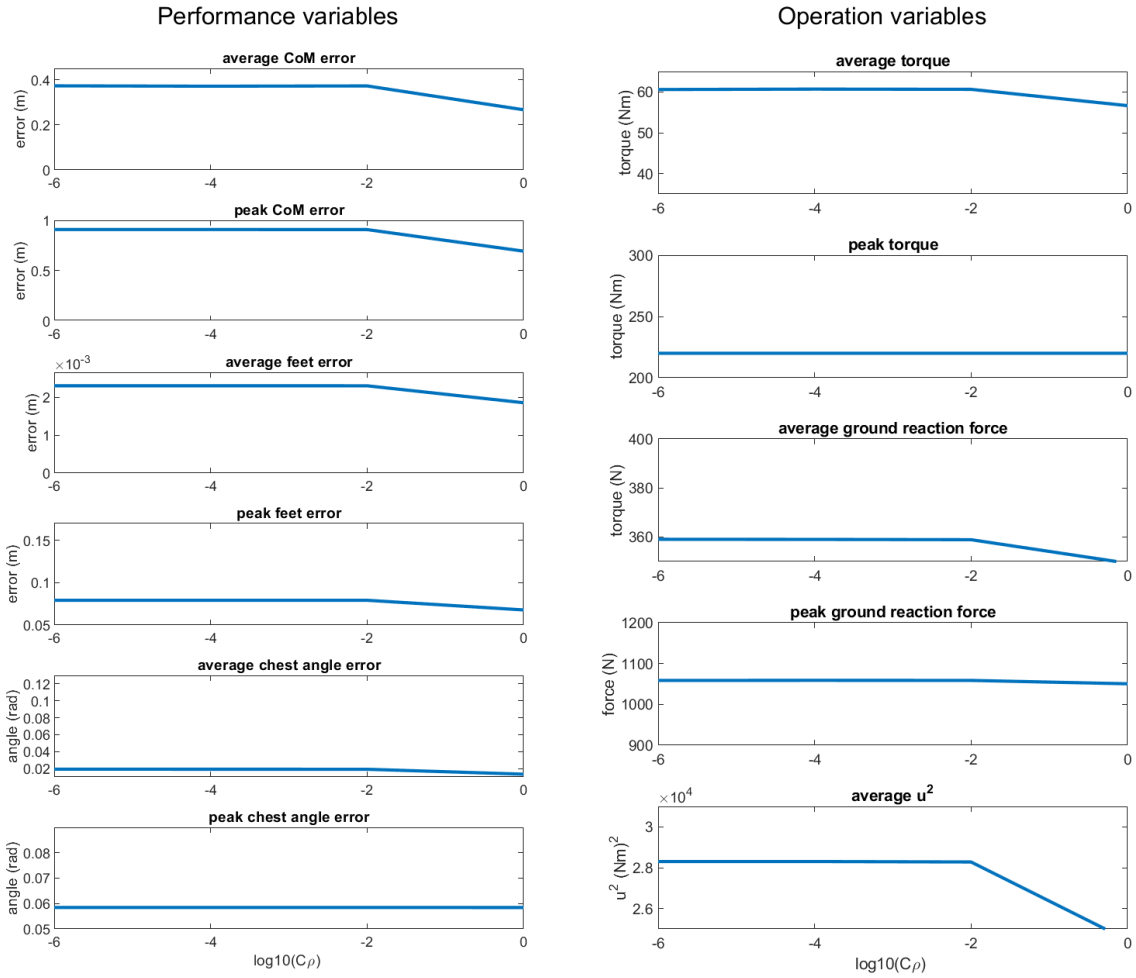
Figure F.2: Push analysis of different ground reaction force limit values

F.2 Ground reaction force weight



(a) (b)

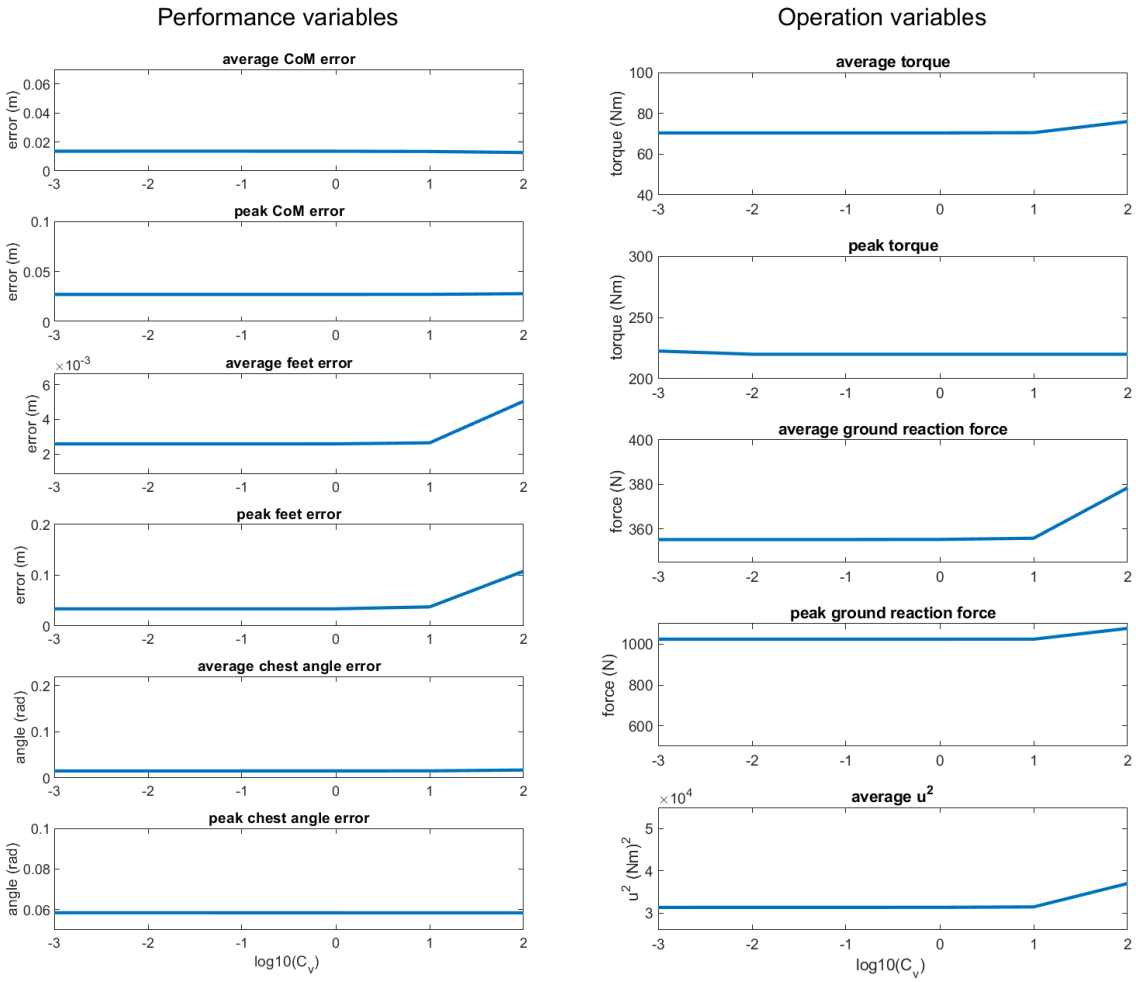
Figure F.3: Walking analysis of different joint acceleration weight,C_ρ, values



(a) (b)

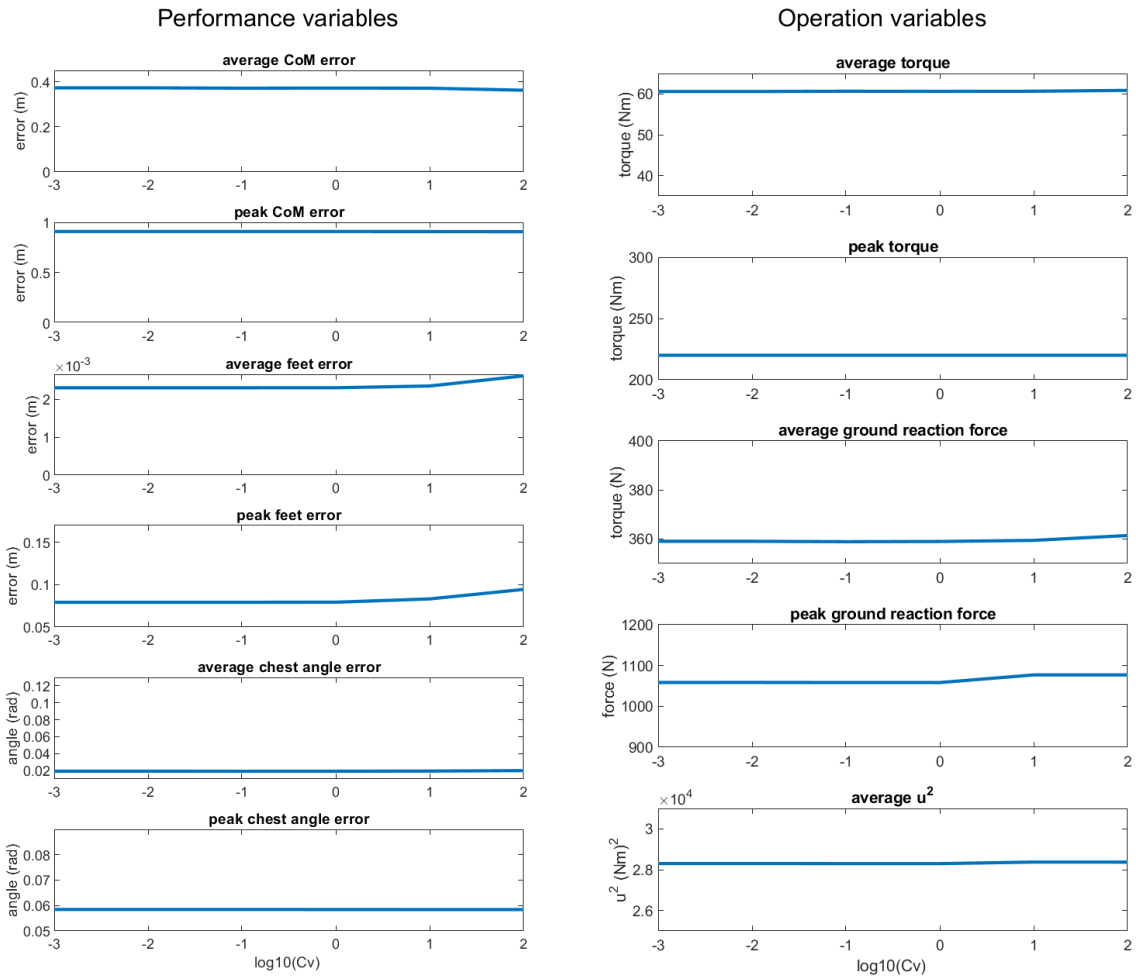
Figure F.4: Push analysis of different joint acceleration weight,C_ρ, values

F.3 Joint Acceleration Weight



(a) (b)

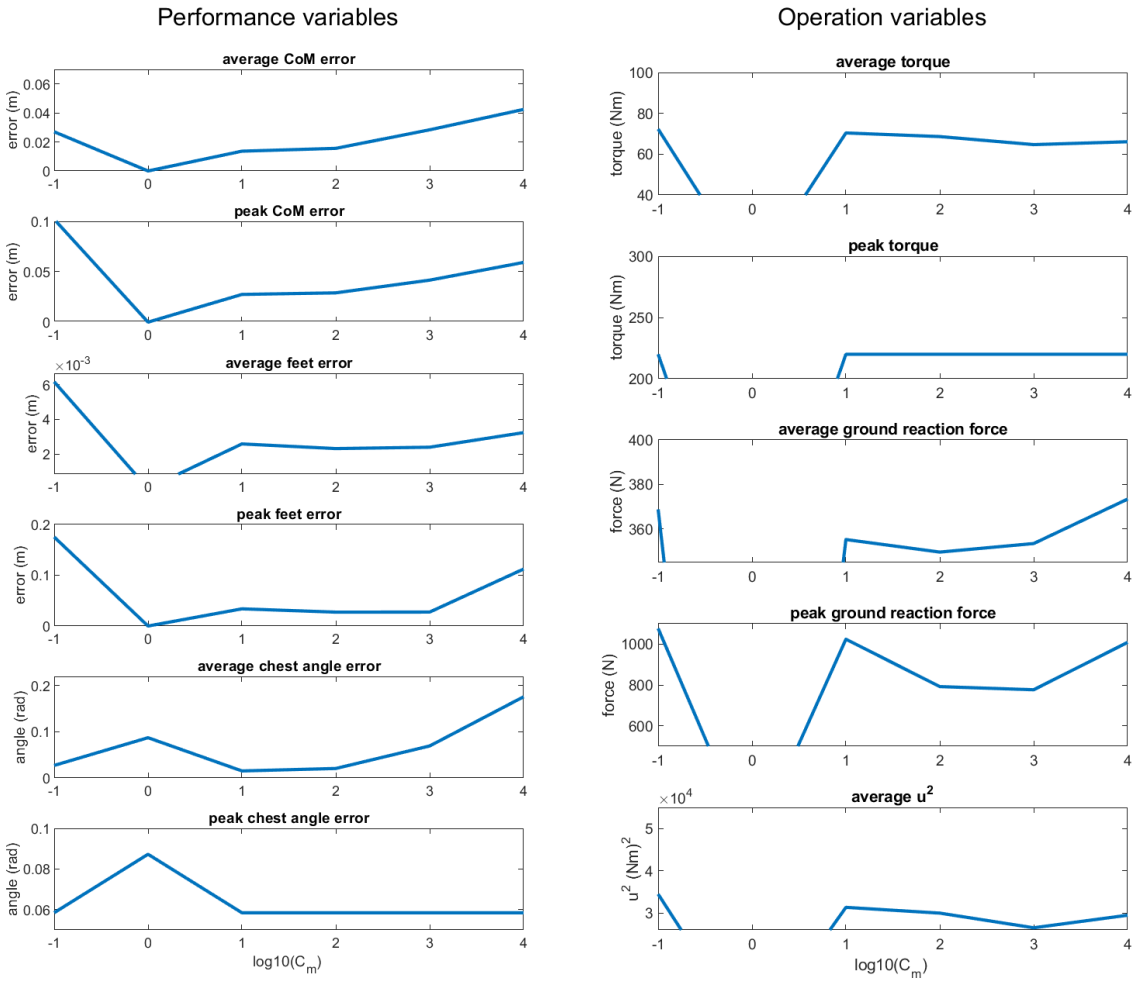
Figure F.5: Walking analysis of different joint acceleration weight,C_{v}, values



(a) (b)

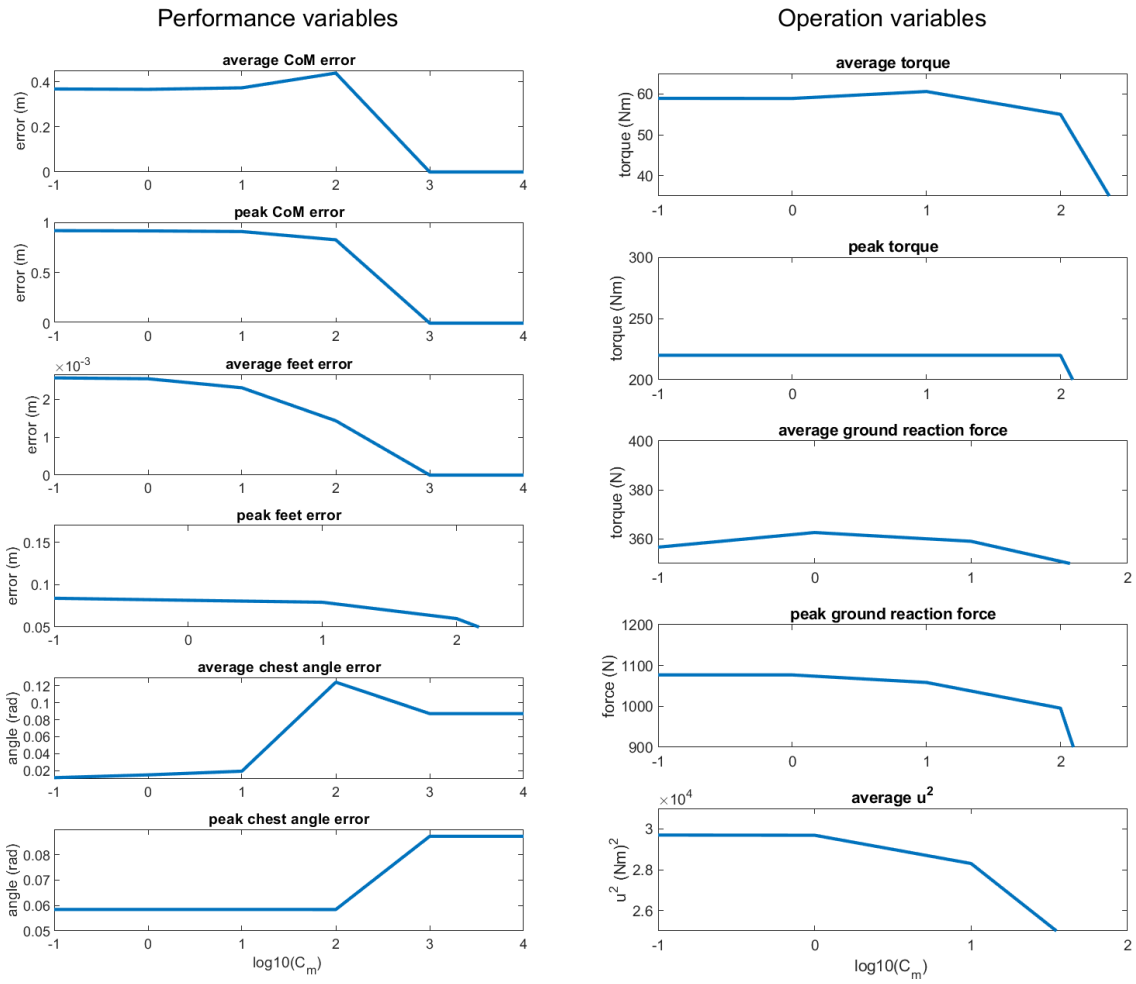
Figure F.6: Push analysis of different joint acceleration weight,C_{v}, values

F.4 Momentum Rate Deviation Weight



(a) (b)

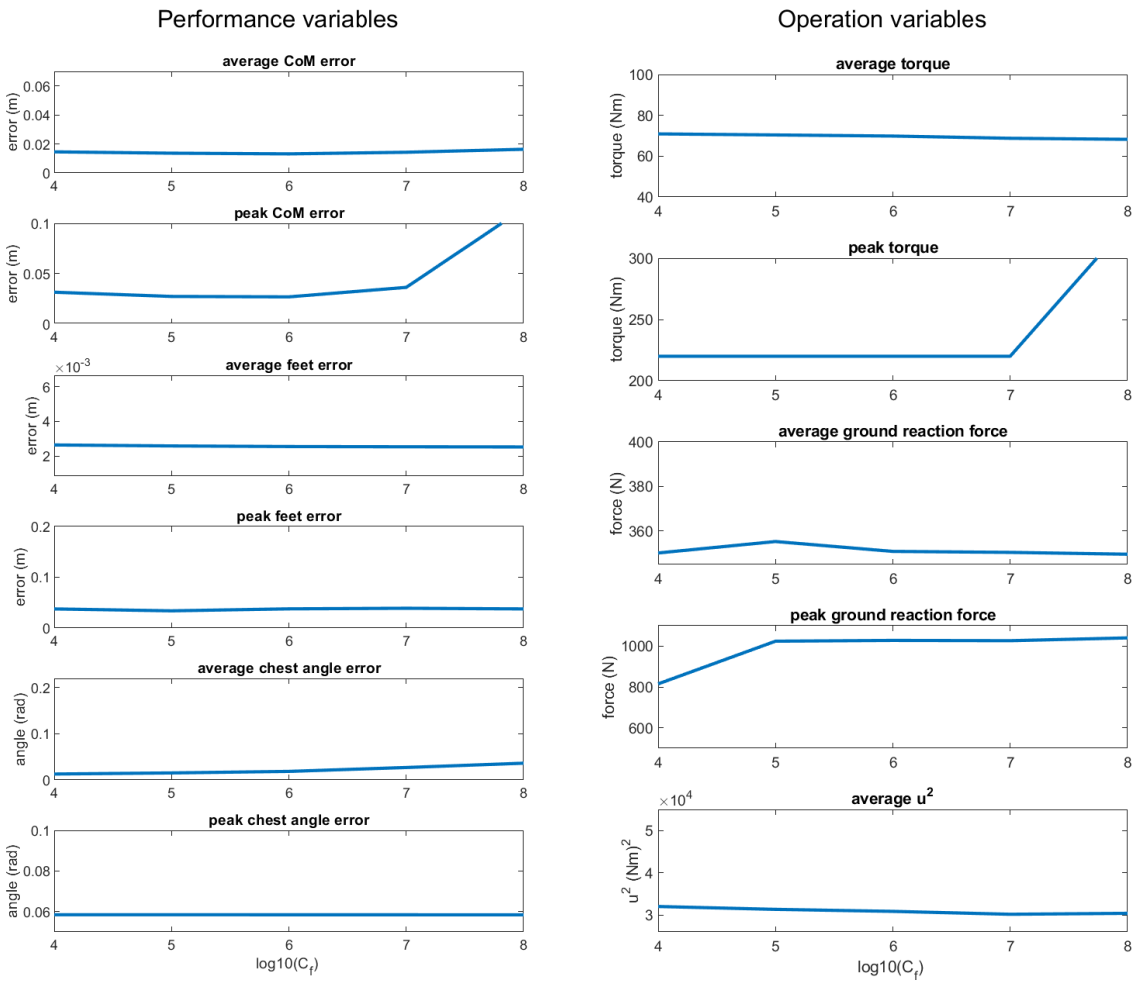
Figure F.7: Walking analysis of different momentum rate deviation weight,C_{m}, values



(a) (b)

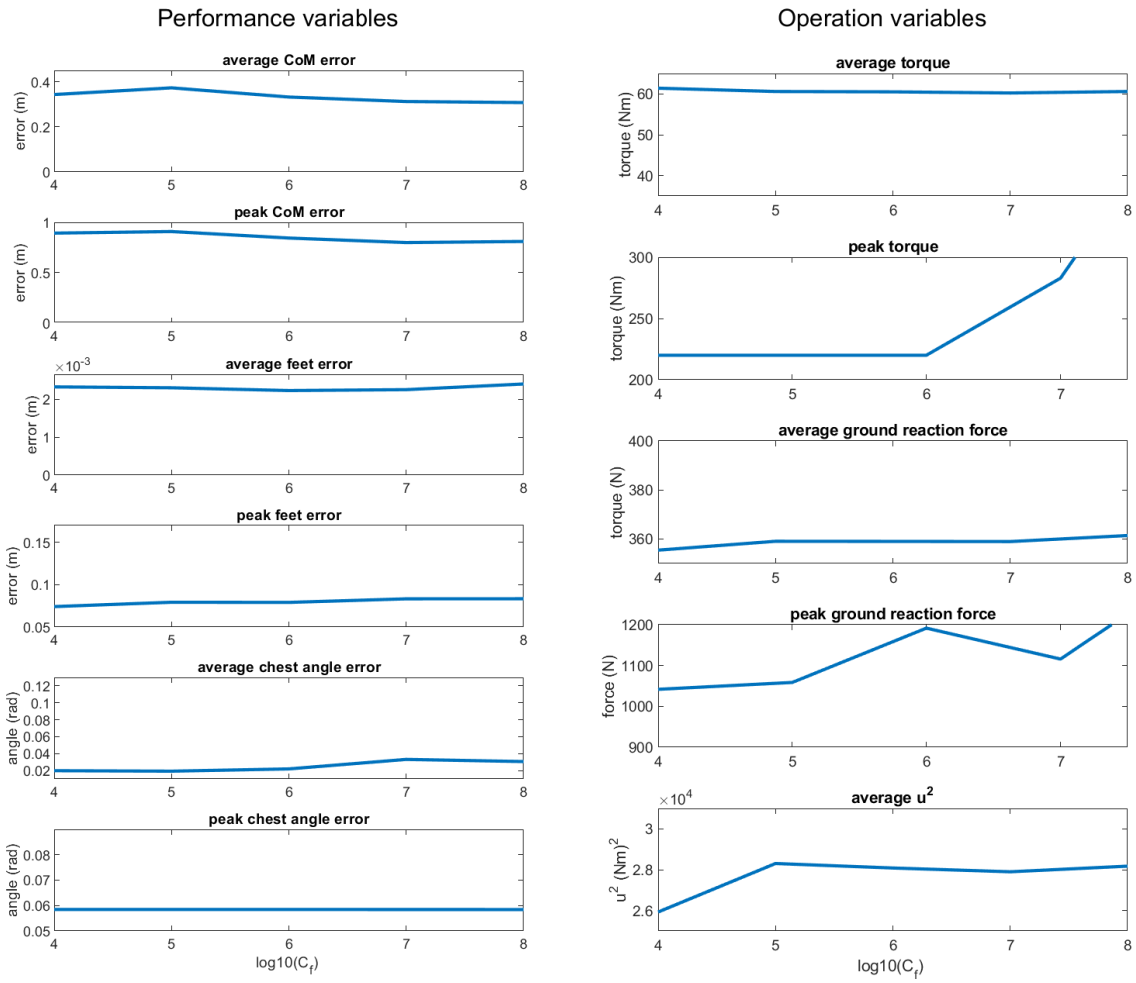
Figure F.8: Push analysis of different momentum rate deviation weight,C_{m}, values

F.5 Feet Acceleration Deviation Weight



(a) (b)

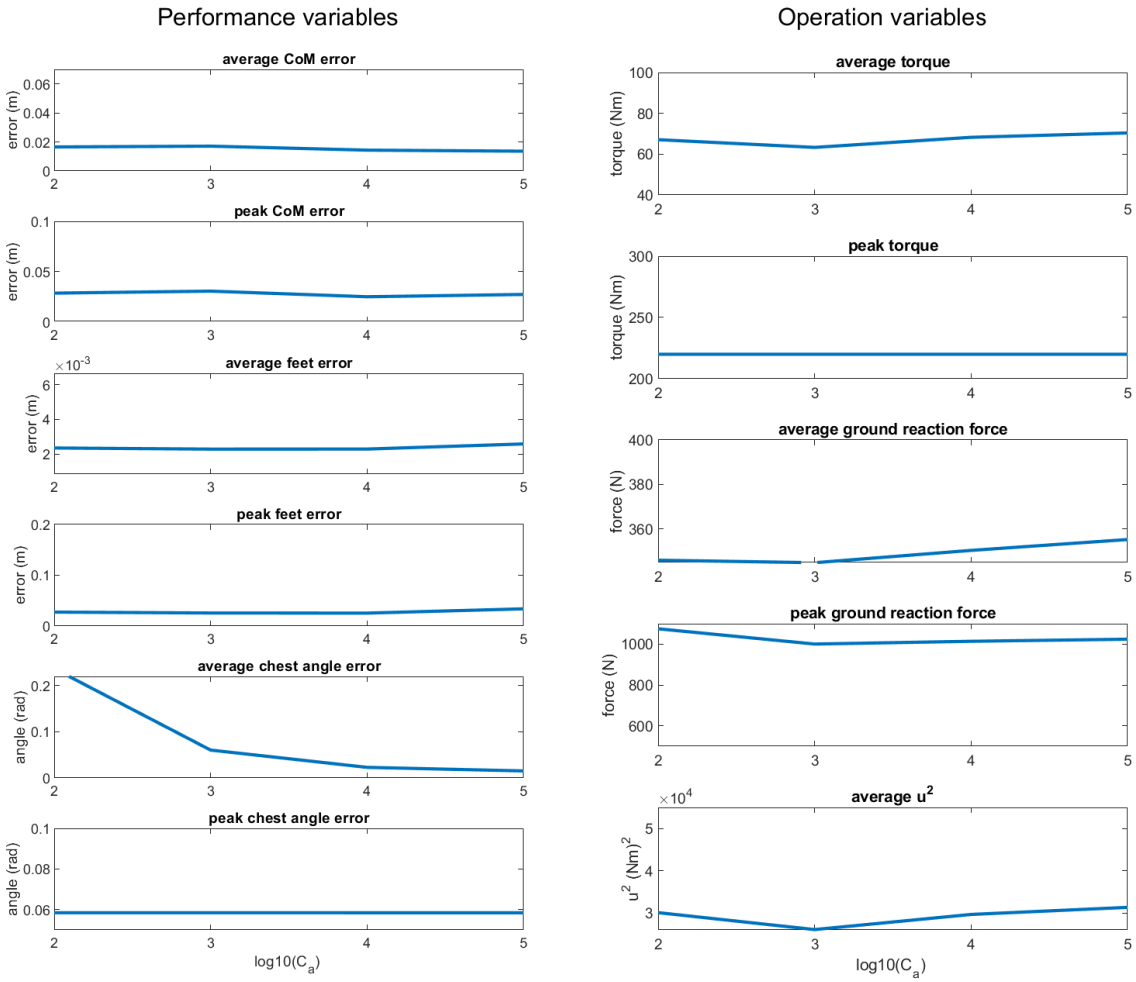
Figure F.9: Walking analysis of different joint acceleration weight,C_{v}, values



(a) (b)

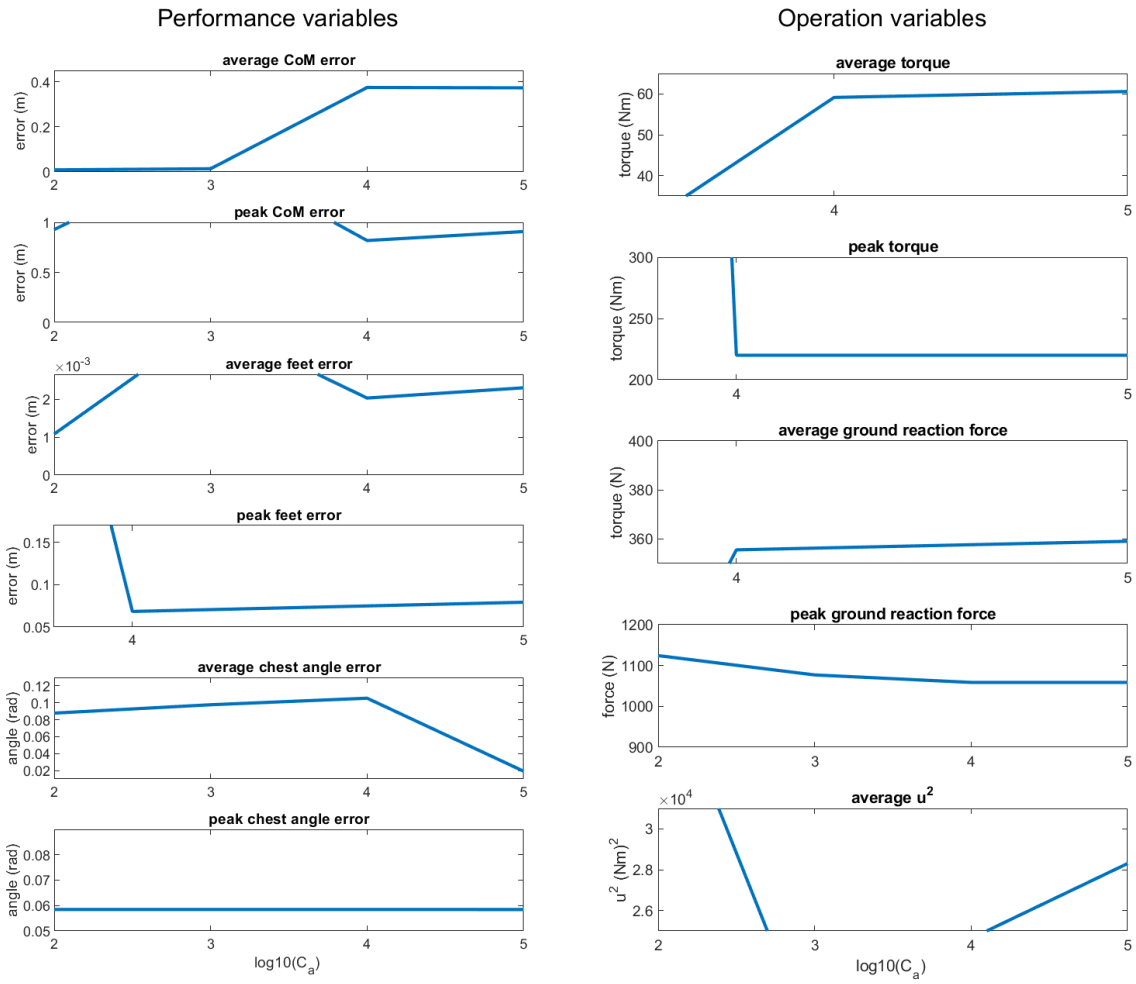
Figure F.10: Push analysis of different joint acceleration weight,C_{v}, values

F.6 Chest Angular Acceleration Deviation Weight



(a) (b)

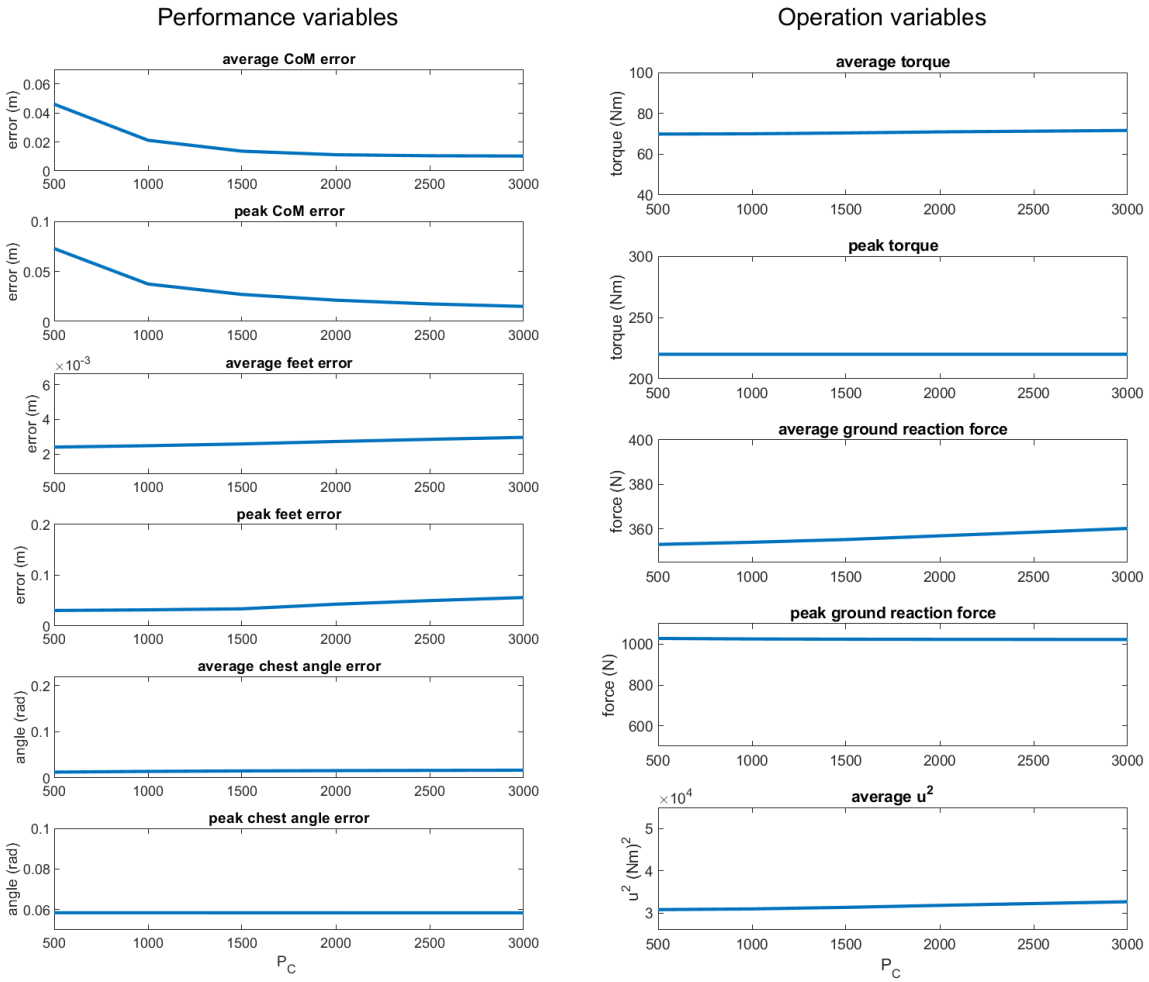
Figure F.11: Walking analysis of different joint acceleration weight,C_{v}, values



(a) (b)

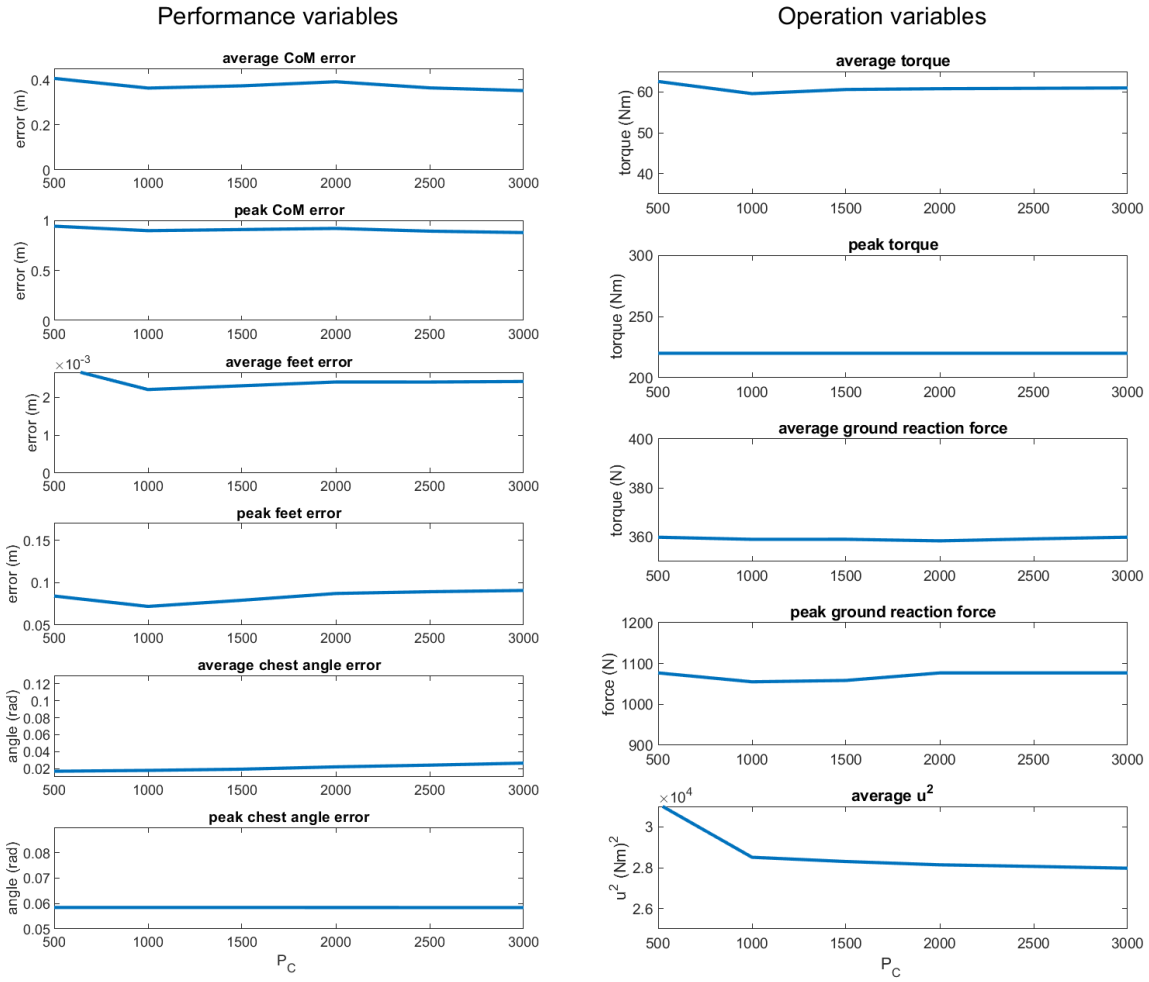
Figure F.12: Push analysis of different joint acceleration weight,C_{v}, values

F.7 Momentum Rate PD Values



(a) (b)

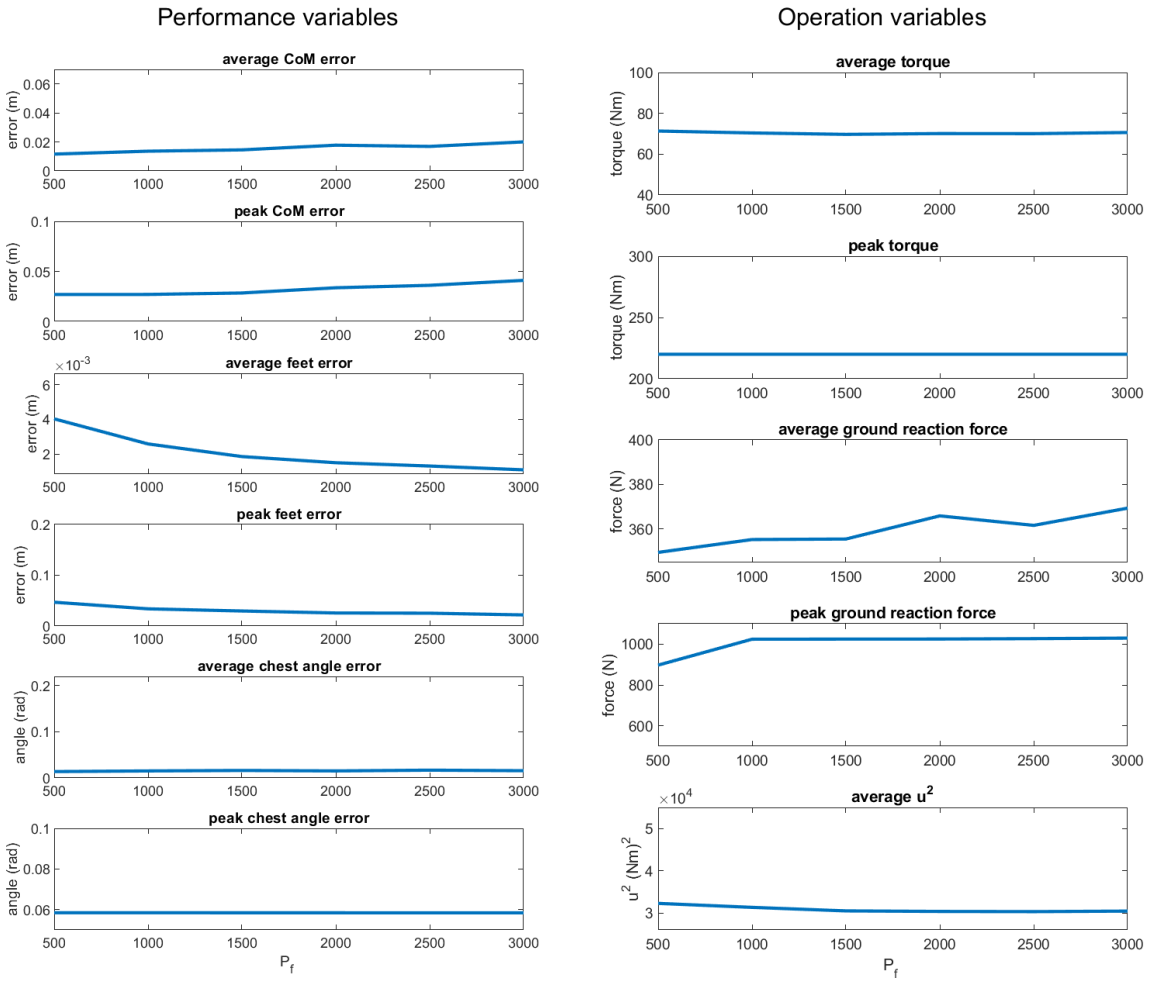
Figure F.13: Walking analysis of different Momentum Rate proportional gain ,P_{C}, values. The system is critically damped, making the derivative gain $D_{\mathrm{C}} = \sqrt{2mP_C}$, with m being the mass of the robot.



(a) (b)

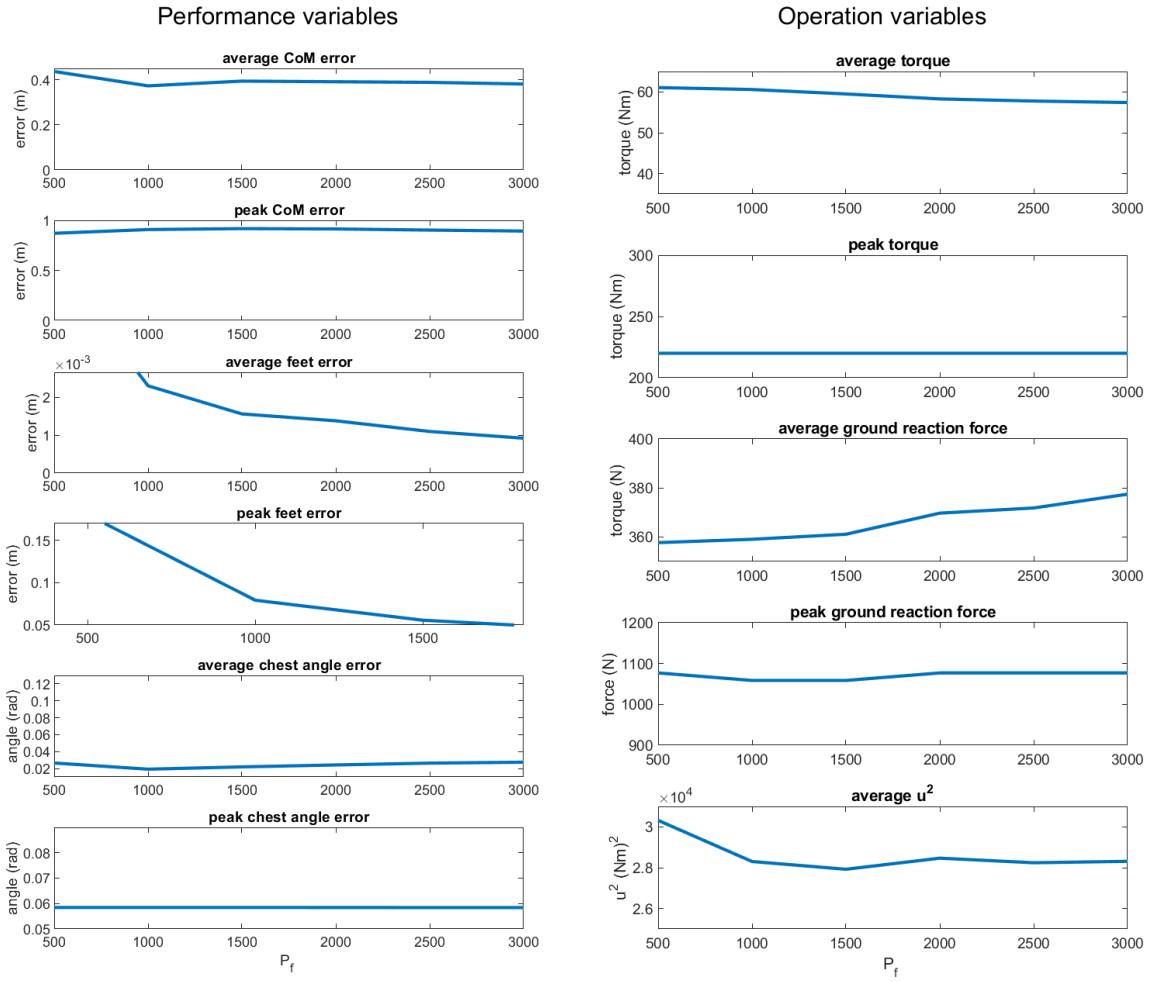
Figure F.14: Push analysis of different Momentum Rate proportional gain ,P_{C}, values. The system is critically damped, making the derivative gain $D_{\mathrm{C}} = \sqrt{2mP_C}$, with m being the mass of the robot.

F.8 Feet Acceleration PD Values



(a) (b)

Figure F.15: Walking analysis of different Momentum Rate proportional gain ,P_{f}, values. The system is critically damped, making the derivative gain $D_{\mathrm{f}} = \sqrt{2P_f}$.



(a) (b)

Figure F.16: Push analysis of different Momentum Rate proportional gain ,P_{f}, values. The system is critically damped, making the derivative gain $D_{\mathrm{f}} = \sqrt{2P_f}$.

Appendix G

Final design parameter values

Table G.1: The baseline values of the controller parameters used in the parameter analysis.

Variable name	**Value**
torque limit	220 Nm
grf limit	1000 N
C_{v}	$1.0 \cdot 10^{-2}$
C_{ρ}	$1.0 \cdot 10^{-4}$
C_{τ}	0
C_{m}	10
C_{f}	$5.0 \cdot 10^{5}$
C_{a}	$1.0 \cdot 10^{5}$
P_{C}	1500
P_{f}	1500

Appendix H

Final design torque over time

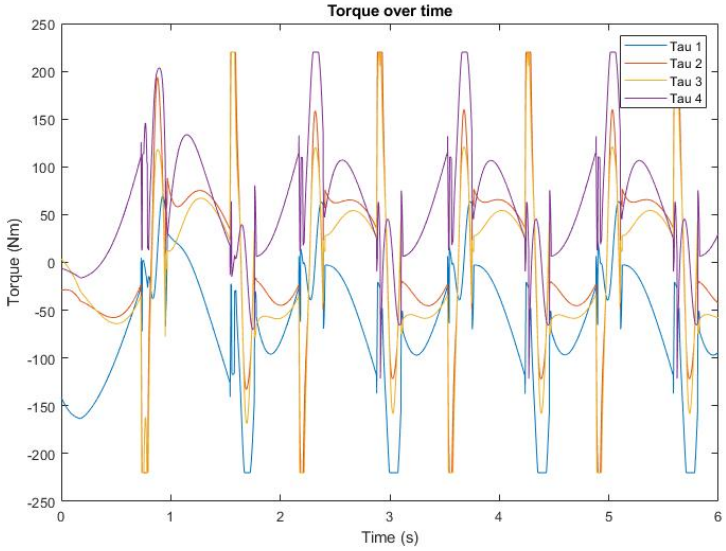


Figure H.1: Torque over time during walking simulation of the final design

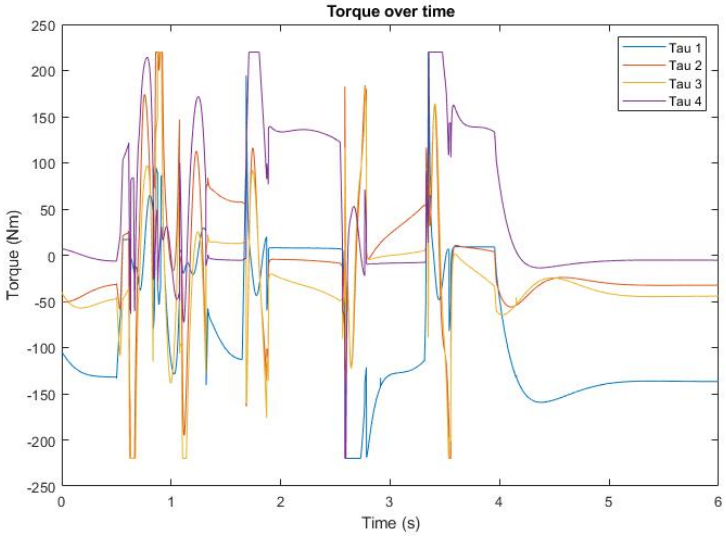


Figure H.2: Torque of time during push recovery simulation of the final design