MASTER THESIS

# Optimizing Trajectories of Bipeds for Robustness Against Unknown Disturbances

Sjors de Bruin

FACULTY OF ENGINEERING TECHNOLOGY
DEPARTMENT OF BIOMECHANICAL ENGINEERING

**EXAMINATION COMMITTEE**
Dr. E.H.F van Asseldonk
Dr. Ir. A.Q.L Keemink
Dr. H. Koroglu

**DAILY SUPERVISOR**
Dr. Ir. A.Q.L. Keemink

**DOCUMENT NUMBER**
BE-977

UNIVERSITY OF TWENTE.

12-2023

## Abstract

Walking is an activity that most people take for granted every day. However, not everyone is so fortunate. For instance, a large part of people who encounter a spinal cord injury have to rely on wheelchairs to move around. Even though wheelchairs are a reliable and cheap solution, they have numerous disadvantages such as restricted access to public transport and buildings, and health issues related to the sitting lifestyle. As an alternative or companion to wheelchairs, lower limb exoskeletons could be a solution. These devices show great potential to improve impaired people's lives, but they need to be improved in multiple aspects as energy consumption and disturbance handling. This thesis focuses on the disturbance handling of exoskeletons. The user of an exoskeleton continuously exerts (disturbance) forces on the exoskeleton. These forces can cause the exoskeleton to slip or fall. This failure occurs due to the robot facing physical limits such as friction or motor torque limits. Therefore, it is useful to consider these limits when planning a trajectory and stay as far away as possible from them.

This thesis presents a method to optimize trajectories of bipedal robots to withstand a force on the center of mass (CoM) that cannot be counteracted without changing the planned trajectory. These trajectories are obtained by transcribing the optimization problem to a direct collocation problem, which is solved as a non-linear problem via an interior point method. The trajectory optimization was performed on a planar 7-DoF walker model. This study shows that optimizing for robustness for bipedal robots leads to trajectories that are more robust against disturbances on the CoM. To implement the optimization algorithm in a real-world application, two shortcomings have to be overcome. First, the optimization needs to be applied and tested on a 3D model instead of a planar model. Second, the computational time should decrease to enable online implementation.

# Contents

# 1 General Introduction

## 1.1 Spinal Cord Injury

Around 250.000-500.000 people suffer from a spinal cord injury (SCI) each year [1]. People who suffer from SCI have damage to their spinal cord. This is mostly caused by trauma, but can also be caused by disease. An SCI can result in loss of sense or motor control in the upper limbs, lower limbs, or body. In very severe cases it can even cause loss of organ functions. The extent to which the patient is paralyzed is determined by the severity and the location of the SCI. The higher up the spine the injury, the more muscles are affected by the injury. Further, SCIs are categorized as complete or incomplete. With a complete SCI, all feeling and motor control is lost in the affected parts. In incomplete SCI only a part of the feeling and motor control is lost. Almost half of the patients have a complete SCI [2]. This has an enormous impact on their daily lives, mostly due to the loss of their ability to walk. For mobility, they often rely on manual wheelchairs. Even though a wheelchair is an accessible and helpful tool, wheelchairs have multiple issues that constrain the lives of their users. Besides issues such as restricted access to public transport and buildings, people in wheelchairs also experience the negative physical effects of (manual) wheelchairs. Under wheelchair users, there is a high prevalence of shoulder and wrist pain due to the repetitive and straining motion [3]. This pain can lead to a downward spiral in which decreasing exercise leads to a decrease in the physical fitness of wheelchair users. This can lead to conditions such as obesity, diabetes, cardiovascular diseases, metabolic syndromes, and osteoporotic conditions.

## 1.2 Exoskeletons as a Solution

As an alternative or companion to wheelchairs, this thesis will focus on lower limb exoskeleton robots (LLE). Van Dijsseldonk et al. found that exoskeleton use can improve patients' lives with complete SCIs on both a social and physical level [4]. However, the users also reported that it had some shortcomings as an assistive device during daily life. The main issue was needing a buddy and crutches when using the exoskeleton. This was found to be limiting the individual's independence. Ideally, the exoskeleton should be used independently. To achieve this, four general aspects need to be improved for exoskeletons, or more generally for all bipedal robots [5]:

1. The ability to adapt to various terrains.
2. The ability to handle unknown disturbances.
3. To switch between different types of gait.
4. To be energy efficient.

## 1.3 Scope of this Thesis

This thesis will focus on the improvement of disturbance handling. When an unknown force is applied to a legged robot, it can cause it to fail to track the original trajectory, slip, or fall over. This failure occurs due to the robot reaching its physical limits. These can be friction limits, torque limits, or configuration limits. Exoskeletons continuously experience disturbances from their users. To prevent the disturbances from the users causing the exoskeleton to fall, these disturbances must influence the exoskeleton as minimally as possible. This can be achieved by preventing the robot from crossing its physical limits.

Considering physical limits when planning the robot's motion is widely done for legged robots. One of the most recent methods is using the feasible wrench polytope (FWP). This polytope contains all the wrenches on the CoM of the robot that adheres to friction, torque, and configuration constraints [6]. However, the 'distance' to these constraints is rarely considered. So when the planned control wrench is close to the edge of the FWP, a small disturbance could push the system into a situation in which the wrench needed to counteract the disturbance is outside of the FWP.

## 1.4 Goal of this Thesis

The goal of this thesis is to optimize trajectories for exoskeletons for robustness against disturbances on the CoM and in a broader sense for all bipedal robots. The idea is to prove the added value of optimizing robustness. If successful it can be used as a stepping stone to implementing this type of optimization in model predictive control.

# 2 Background

In this chapter, three key concepts that are used in this thesis will be discussed. First, non-linear programming will be explained. Next, the concept and mathematical representation of polytopes will be discussed. Last, the Chebyshev center and radius are discussed.

## 2.1 Nonlinear Programming

A nonlinear program is a specific form of a constrained minimization problem that has either nonlinear terms in the objective or the constraints. A NLP is typically represented as [7]:

$$\min_{\mathbf{z}} \quad \mathbf{J}(\mathbf{z}) \qquad \text{s.t.} \tag{1a}$$

$$\mathbf{f}(\mathbf{z}) = \mathbf{0} \tag{1b}$$

$$\mathbf{g}(\mathbf{z}) \leq \mathbf{0} \tag{1c}$$

$$\mathbf{z}_{\text{low}} \leq \mathbf{z} \leq \mathbf{z}_{\text{upp}} \, . \tag{1d}$$

The goal is to minimize the objective function $\mathbf{J}(\mathbf{z})$ by adjusting the decision variables $\mathbf{z}$, while satisfying all constraints. These constraints can be equality constraints ($\mathbf{f}(\mathbf{z}) = \mathbf{0}$), inequality constraints ($\mathbf{g}(\mathbf{z}) \leq \mathbf{0}$) and/or boundaries on the decision variables $\mathbf{z}_{\text{low}} \leq \mathbf{z} \leq \mathbf{z}_{\text{upp}}$ . In this thesis, the NLP is solved using an interior point method. An explanation of the interior point method is given in Appendix C.

## 2.2 Polytopes

A polytope can be defined in two ways, it can be defined by a set of points or by a set of linear inequalities [8]. In this section, the focus will be on convex polytopes. The definition for a convex polytope using a set of points is as follows: Consider a set S in $\mathbb{R}^n$. S is a convex polytope if and only if a line between any two points in S, is fully in S. The definition for a convex polytope using linear inequalities is: A convex polytope is an intersection of a finite number of closed halfspaces which also has an upper bound on the distance of any point in the polytope with respect to the origin. A closed halfspace is the set of points, satisfying a linear inequality as given in Eqn. (2).

$$ax \leq b \tag{2}$$

So a convex polytope can be represented as:

$$\mathbf{Ax} \leq \mathbf{b}, \tag{3}$$

under the condition that the inequalities define a closed space.

## 2.3 Chebyshev Centre

The Chebyshev Centre is the centre of the maximum inscribed ball in a polytope [9]. In the coming section, a derivation will be done on how to find the Chebyshev Centre. Consider a ball defined by:

$$B = \{\boldsymbol{x}_c + \boldsymbol{u} : \|\boldsymbol{u}\|_2 \leq r\} \, , \tag{4}$$

where $\boldsymbol{x}_c$ is the centre of the ball, $r$ is the radius of the ball and $\boldsymbol{u}$ is the set of points within the ball. Also, consider a polytope described by a set of linear equations:

$$\mathbf{a}_i^T \mathbf{x} = b_i, \quad \text{for } i = 1, 2, \cdots n. \tag{5}$$

Where $n$ is the number of constraints. To find the largest inscribed ball, the radius must be maximized and all points $\mathbf{x}_c + \mathbf{u}$ must lie within the polytope:

$$\mathbf{a}_i^T (\mathbf{x}_c + \mathbf{u}) = b_i \quad \text{for } i = 1, 2, \cdots n. \tag{6}$$

To find the largest radius the supremum needs to be found:

$$\sup_{\|\boldsymbol{u}\|_2 \leq r} \left\{ \boldsymbol{a}_i^T \left( \boldsymbol{x}_c + \boldsymbol{u} \right) \right\} \leq b_i, \quad \text{for } i = 1, 2, \cdots, n \tag{7}$$

Since $\mathbf{u}$ is the parameter that is related to the radius, it is the only parameter that needs to be optimized:

$$\boldsymbol{a}_i^{\boldsymbol{T}} \boldsymbol{x}_c + \sup_{\|\boldsymbol{u}\|_2 \leq r} \left\{ \boldsymbol{a}_i^{\boldsymbol{T}} \boldsymbol{u} \right\} \leq b_i, \quad \text{for } i = 1, 2, \cdots, n \tag{8}$$

The largest value of the inner product is when they are in the same direction, therefore:

$$\sup_{\|\boldsymbol{u}\|_2 \leq r} \left\{ \boldsymbol{a}_i^{\boldsymbol{T}} \boldsymbol{u} \right\} = \boldsymbol{a}_i^T \left( \|\boldsymbol{u}\|_2^{\max} \frac{a_i}{\|\mathbf{a}_i\|_2} \right) = r \cdot \|\mathbf{a}_i\|_2 \tag{9}$$

Then the maximum radius can be found by minimizing $-r$:

$$\min_{\mathbf{x}_c} \quad -r \qquad \text{s.t.} \tag{10a}$$

$$\boldsymbol{a}_i^{\boldsymbol{T}} \boldsymbol{x}_c + \|\mathbf{a}_i\|_2 \quad \text{for } i = 1, 2, \cdots, n. \tag{10b}$$

# 3 Research Paper

# Optimizing Trajectories of Bipedal Robots for Robustness Against Disturbances

S. de Bruin, A.Q.L. Keenmink, E.H.F. van Asseldonk, H. Koroglu

***Abstract*—Robustness against disturbances is one of the key aspect for stable gait and balance of bipedal robots. This work presents a method to optimize trajectories to withstand a force on the center of mass that cannot be counteracted without changing the planned trajectory. These trajectories are obtained by transcribing the optimization problem to a direct collocation problem, which is solved as a non-linear problem via an interior point method. This study shows that optimizing for robustness for bipedal robots leads to more trajectories that are more robust against disturbances on the CoM.**

## I. Introduction

Disturbance rejection against unknown forces is an important aspect of maintaining balance for legged robots. When an unknown external force is applied to a legged robot, it can cause the robot to fail to track the original trajectory, slip, or fall over. This failure occurs due to the robot facing physical limits such as friction or motor torque limits. Therefore, it is useful to take these limits into account when planning a trajectory and stay as far away as possible from them. This is especially important for lower limb exoskeletons as these can be considered as a legged robot with disturbances coming from the user.

Considering physical limitations is already widely done in trajectory planning for legged robots. The planning of the trajectory of the robot can be performed both offline and online. When a trajectory is planned offline, all necessary joint positions, velocities, and torques to perform a movement are determined beforehand. This planned trajectory is then executed by low-level controllers that can control the joint angles and/or torques. In online planning, the trajectory is planned and simultaneously executed by low-level controllers. The advantage of online planning is that the trajectory can be replanned when facing an unexpected disturbance. A way to implement online trajectory planning is model predictive control (MPC). MPC is an iterative control strategy that optimizes trajectories for a specified period of time using a model of the robot it controls. Each timestep the horizon of the time period is shifted and the trajectories are optimized again given the current state of the system [1].

Considering friction limits when planning trajectories can be done by using the contact wrench cone (CWC). The CWC contains all contact wrenches on the center of mass (CoM) of the robot that satisfy the friction constraints [2]. As long as the contact wrenches are in their CWC the planned motion can be performed without slipping. Multiple works showed that this can be used to generate stable gait or balance [3]–[5]. However, more constraints apply to bipedal robots, e.g. torque and configuration limits. To include these constraints in the planning of trajectories, the CWC can be extended to a feasible wrench polytope (FWP). The FWP contains all wrenches that comply with friction, actuation, and configuration limits [6]. The FWP is a combination of the contact wrench cone (CWC) and the actuation wrench polytope (AWP). The AWP contains all wrenches that satisfy all configuration and torque limits of the robot, so these are the wrenches that the robot can generate given the current configuration of the robot and the torque limits of the actuators [6]. Multiple works have been published that successfully used the FWP in the planning of trajectories of their robots [6]–[8].

Even though these methods result in stable gait and/or stances, the 'level' of robustness is rarely considered. This level of robustness is especially important for exoskeletons since they continuously experience disturbances from the user. When the planned trajectory gets close to the limits, a small disturbance can lead to the robot falling or slipping.

Orsolino et. al. consider a robustness metric when planning a trajectory [6]. They compute an FWP using a vertex description. The stability metric they use is the amount the FWP can shrink while still containing the gravito-inertial wrench. This metric is optimized when planning the trajectory of the robot. They were able to implement this in online trajectory optimization. However, they use a quadrupedal robot instead of a bipedal robot. Furthermore, during the generated gait only one of the legs moves during a step. This allowed them to assume quasi-static conditions during the optimization. Also, they assumed that the FWP does not change when performing a step. These assumptions are not valid when planning gait trajectories for bipedal robots.

Ferrolho et al. proposed a method using the smallest unrejectable force (SUF) [9], [10]. They defined the SUF as the smallest force that cannot be rejected without changing the original trajectory. They optimized the SUF on the 6-DoF end-effector which was mounted on a quadruped. The results showed increased robustness against disturbances on the end-effector. However, the optimization was only performed and tested during stance and not during locomotion.

This work optimizes trajectories for bipedal robots while maximizing the SUF on the CoM. The contributions are threefold. First, it assesses the robustness for disturbances on the CoM instead of an end-effector. Second, the robustness is also tested during locomotion instead of only during stance. Thirdly, the robustness is optimized on bipedal robots instead of quadrupeds. The outcome of this work could show that optimizing for robustness leads to a more stable gait.

## II. METHODS

This section describes the used methods of this work. First the used model and the used optimization technique will be discussed. Hereafter, the implementation of the optimization algorithm is discussed.

### A. Model Formulation

A legged robot can be modeled as a floating body with limbs attached to it. This work uses a 7-DoF planar model of a bipedal robot, as shown in Fig. 1. The vector of generalized coordinates is denoted by $\mathbf{q}$. The Equation of Motion (EoM) of the floating frame interacting with its environment can be written as:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\dot{\mathbf{q}}, \mathbf{q})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{S}\mathbf{u} + \mathbf{J}_{\lambda}^{T}(\mathbf{q})\boldsymbol{\lambda} + \mathbf{J}_{\text{com}}^{T}(\mathbf{q})\boldsymbol{\kappa}, \tag{1}$$

where $\mathbf{M}(\mathbf{q})$ is the mass matrix that describes the rotational and translational inertia of the system; $\mathbf{C}(\dot{\mathbf{q}}, \mathbf{q})$ describes the Coriolis and centripetal behavior of the system; $\mathbf{g}(\mathbf{q})$ describes the potential forces in the system; $\mathbf{S}$ is a selection matrix that maps the joint torques $\mathbf{u}$ to the actuated generalized forces; $\mathbf{J}_{\lambda}^{T}(\mathbf{q})$ maps the ground reaction forces (GRF) $\boldsymbol{\lambda}$ to generalized forces; $\mathbf{J}_{\text{com}}^{T}(\mathbf{q})$ maps a force $\boldsymbol{\kappa}$ on the center of mass (CoM) to generalized forces. To convert the EoM to an equation that describes the dynamics, the EoM is rewritten to:

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(-\mathbf{C}(\dot{\mathbf{q}}, \mathbf{q})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}) + \mathbf{S}\mathbf{u} + \mathbf{J}_{\lambda}^{T}(\mathbf{q})\boldsymbol{\lambda} + \mathbf{J}_{\text{com}}^{T}\boldsymbol{\kappa}). \tag{2}$$

This expression is used to define a function that describes the dynamics.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\kappa}). \tag{3}$$

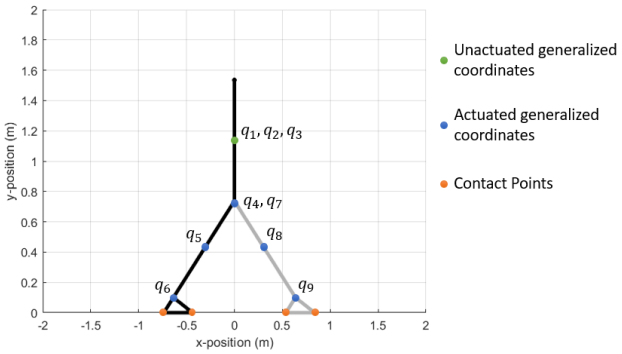


Fig. 1: 7-DoF bipedal model, with generalized coordinates and contact points. $q_1$ and $q_2$ are the x and y-coordinate of the upper body. The angle of the upper body is $q_3$. The angles of the hip, knee, and ankle joints are $q_4 - q_9$.

### B. Trajectory Optimisation

In trajectory optimization, non-linear programs (NLP) are used to optimize trajectories. The optimizer tries to optimize an objective function under a set of constraints. To be able to use an NLP to solve the optimization problem, the continuous time optimization problem has to be converted to a discrete optimization problem. For this a direct transcription method is used, also known as collocation. The first step in collocation is to write the continuous time problem into a discrete-time problem [11]. This is done using so-called knot points. On each knot point, all variables need to be defined:

$$\begin{aligned} t &\rightarrow [t_0\ t_1\ \dots\ t_N], \\ \mathbf{x} &\rightarrow [\mathbf{x}_0\ \mathbf{x}_1\ \dots\ \mathbf{x}_N], \\ \dot{\mathbf{x}} &\rightarrow [\dot{\mathbf{x}}_0\ \mathbf{x}_1\ \dots\ \dot{\mathbf{x}}_N], \end{aligned} \tag{4}$$

where $N$ is the number of knot points; $t$ is the time; $\mathbf{x} = \begin{bmatrix} \mathbf{q}^T\ \dot{\mathbf{q}}^T \end{bmatrix}^T$ and; $\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}}^T\ \ddot{\mathbf{q}}^T \end{bmatrix}^T$. To make the transition of states between knot points physically possible, a set of constraints is set on the dynamics. The dynamics are approximated with some quadrature method. The idea is that the state change of the system between knot points is equal to the integral of the approximation. This work uses first-order trapezoidal collocation for integration. The first-order trapezoidal integration constraints are defined as:

$$\mathbf{x}_{k+1} - \mathbf{x}_k - \frac{h_k}{2}(\mathbf{f}_{k+1} + \mathbf{f}_k) = \mathbf{0}. \tag{5}$$

Where $\mathbf{x}_{k+1}$ are the states on the next knot point, $\mathbf{x}_k$ are the states on the current knot point, $h_k$ is the time step between the knot points, and $\mathbf{f}_{k+1}$ and $\mathbf{f}_k$ are the output of the dynamics on both knot points, as defined in Eqn. 3. A derivation on how to formulate this constraint is given in Appendix A.

### C. Problem Formulation

To improve convergence for the robust trajectory generation, the process is split into three parts. First, a trajectory is generated that does not take into account any robustness metrics. The second step is to obtain the robustness metrics for the trajectory found in the previous step. The last step is to use the found trajectory and robustness of the previous steps as an initial guess for the robust optimization. This method is an adapted version of the method described by Ferrolho et al. [10]. In the following subsections, each step will be discussed.

#### 1) Non-Robust Trajectory Generation

Two parts are essential to construct an NLP: the decision variables and the constraints. The decision variables are the states of the system and the forces (in this case the ground reaction forces and the joint torques) acting on the system at each knot point. So the vector of decision variables $\mathbf{z}$ is defined as:

$$\mathbf{z} = \begin{bmatrix} \mathbf{x}_0^T\ \mathbf{u}_0^T\ \boldsymbol{\lambda}_0^T\ \mathbf{x}_1^T\ \mathbf{u}_1^T\ \boldsymbol{\lambda}_1^T\ \cdots\ \mathbf{x}_N^T\ \mathbf{u}_N^T\ \boldsymbol{\lambda}_N^T \end{bmatrix}^T. \tag{6}$$

Next, constraints should be defined. Three sets of constraints will be defined: boundary constraints, collocation constraints, and contact constraints.

Boundary constraints constrain the decision variables. Those are needed because the decision variables are physically constrained in reality, e.g. joints have maximal angles, actuators have torque limits, and GRFs have to be positive in the y-direction as feet cannot pull on the ground.

The next set of constraints is the collocation constraints, as given in Eqn. 5.

Last, constraints regarding contact with the ground need to be constructed, these differ when the foot is in contact with the ground. Three types of constraints are involved; constraints regarding foot velocity, foot position, and constraints regarding the GRFs. There are two constraints for position and velocity. The y-position of the contact point of the foot that is in contact with the ground should be equal to zero, if not in contact, it should be larger than zero. In addition, during contact, the horizontal velocity of the foot should be equal to zero, if not in contact it is unconstrained. These constraints are mathematically represented as:

$$\mathbf{C}_c(\mathbf{z}) = \begin{cases} p_y^{\text{foot}} = 0 & \text{Contact} \quad (7a) \\ v_x^{\text{foot}} = 0 & \text{Contact} \quad (7b) \\ p_y^{\text{foot}} \geq 0 & \text{No contact.} \quad (7c) \end{cases}$$

When in contact the GRFs should be within the friction cone. When the GRFs are within the friction cone, the foot does not slide. The constraint that the GRF is within the friction cone is described by:

$$\begin{bmatrix} -1 & -\mu \\ 1 & -\mu \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \lambda_k^x \\ \lambda_k^y \end{bmatrix} = \mathbf{A}_\lambda \boldsymbol{\lambda}_k \leq \mathbf{b}_f = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad (8)$$

where $\mu$ is the static friction coefficient, $\lambda_x$ and $\lambda_y$ are the horizontal and vertical component of $\boldsymbol{\lambda}$ respectively and $k$ is the current knot point index. When the foot is not in contact the GRFs are equal to zero. Note that these constraints need to hold for all contact points, so each contact point has a set of GRF constraints. Whether a foot is in contact with the ground is determined beforehand for each knot point and contact point, this will be referred to as the contact sequence. The contact sequence was based on the human contact sequence during gait [12]. Using all the constraints, the total NLP can be defined as:

$$\min_{\mathbf{z}} \quad \frac{1}{N}\sum_{k=1}^{N}(w_u \mathbf{u}_k^T \mathbf{u}_k + w_\lambda \boldsymbol{\lambda}_k^T \boldsymbol{\lambda}_k) \qquad \text{s.t.} \quad (9a)$$

$$\mathbf{x}^{\text{LB}} \leq \mathbf{x}_{k=2,\ldots,N-1} \leq \mathbf{x}^{\text{UB}} \quad (9b)$$

$$\mathbf{u}^{\text{LB}} \leq \mathbf{u}_k \leq \mathbf{u}^{\text{UB}} \quad (9c)$$

$$0 \leq \lambda_{y,k} \quad (9d)$$

$$\mathbf{x}_1^{\text{LB}} \leq \mathbf{x}_1 \leq \mathbf{x}_1^{\text{UB}} \quad (9e)$$

$$\mathbf{x}_{\text{N}}^{\text{LB}} \leq \mathbf{x}_{\text{N}} \leq \mathbf{x}_{\text{N}}^{\text{UB}} \quad (9f)$$

$$\mathbf{x}_{k+1} - \mathbf{x}_k - \frac{h_k}{2}(\mathbf{f}_k + \mathbf{f}_{k+1}) = \mathbf{0} \quad (9g)$$

$$\mathbf{C}_c(\mathbf{z}) = \begin{cases} p_{y,k}^{\text{foot}} = 0 & \text{Contact} \quad (9h) \\ v_{x,k}^{\text{foot}} = 0 & \text{Contact} \quad (9i) \\ \mathbf{A}_\lambda \boldsymbol{\lambda}_k \leq \mathbf{b}_f & \text{Contact} \quad (9j) \\ p_{y,k}^{\text{foot}} \geq 0 & \text{No contact} \quad (9k) \\ \boldsymbol{\lambda}_k = \mathbf{0} & \text{No contact.} \quad (9l) \end{cases}$$

Here Eqn. 9e and Eqn. 9f, are additional constraints on the initial and final state to enforce a movement from the start location to the goal.

*2) Obtaining Robustness Metrics*

By solving Eqn. 9, a locally optimal trajectory is obtained, but the robustness metrics still need to be found. This subsection will discuss what the robustness metric is and how it is used in the forming of the NLP.

The robustness is defined as the largest inscribed hypersphere, of which the radius is called the Chebyshev radius, inside the polytope formed by the GRF and torque constraints. The larger the radius, the more stable the system is against disturbances on the CoM. Fig. 2 shows a two-dimensional example of a polytope with the largest inscribed ball in it.

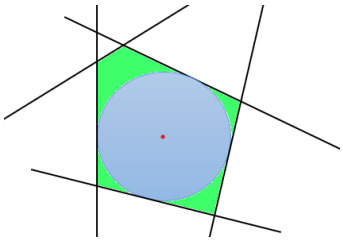

Fig. 2: Schematic example of a 2-dimensional polytope, with the largest inscribed ball in it. The black lines indicate hyperplanes that define the halfspaces. The green area is the polytope defined by these halfspaces. The red dot is the center of the largest inscribed sphere (the blue circle).

The conversion of this idea to an NLP is an adaptation of the work of Ferrolho et al. [9]. First, extended torques and GRFs are defined. These are the combination of the planned torques and GRFs plus the contributions of the disturbance force to them, which is represented as:

$$\mathbf{u}^+ = \mathbf{u} + \mathbf{K}_u \boldsymbol{\kappa} \quad (10)$$

$$\boldsymbol{\lambda}^+ = \boldsymbol{\lambda} + \mathbf{K}_\lambda \boldsymbol{\kappa}, \quad (11)$$

where $\mathbf{K}_u$ and $\mathbf{K}_\lambda$ are matrices that map the force on the CoM to the joint torque space and GRF space respectively. There should be no variation in position, velocity, and acceleration if we were to be robust, substituting Eqn. 10 and Eqn. 11 in the EoM gives:

$$\mathbf{0} = \left(\mathbf{S}\mathbf{K}_u + \mathbf{J}_\lambda^T \mathbf{K}_\lambda + \mathbf{J}_{\text{com}}^T\right) \boldsymbol{\kappa}. \quad (12)$$

Next, Eqn. 10 and Eqn. 11 can be used to redefine the torque constraints and friction cone constraints:

$$\mathbf{A}_u \mathbf{u}^+ \leq \mathbf{b}_u \quad (13)$$

$$\mathbf{A}_\lambda \boldsymbol{\lambda}^+ \leq \mathbf{0}, \quad (14)$$

where $\mathbf{b}_u$ is a vector of torque limits. In the next step, each row is taken from Eqn. 13 and Eqn. 14, then Eqn. 10 and Eqn. 11 are substituted in these equations, this gives:

$$\mathbf{a}_{u,i}\left(\mathbf{u} + \mathbf{K}_u \boldsymbol{\kappa}\right) \leq \mathbf{b}_{u,i} \quad (15)$$

$$\mathbf{a}_{\lambda,i}\left(\boldsymbol{\lambda} + \mathbf{K}_\lambda \boldsymbol{\kappa}\right) \leq \mathbf{0}. \quad (16)$$

Here, $a_{u,i}$ is the $i^{th}$ row of $\mathbf{A}_{u,i}$ and $a_{\lambda,i}$ is the $i^{th}$ row of $\mathbf{A}_{\lambda,i}$. All points $\mathbf{K}_u \boldsymbol{\kappa}$ inside the ball have to satisfy $||\boldsymbol{\kappa}|| \leq \rho$,

where $\rho$ is the radius of the ball. To find the largest radius, the following equation needs to be solved:

$$\mathbf{a}_{u,i}\mathbf{u} + \sup_{||\boldsymbol{\kappa}||\leq\rho} \{\mathbf{a}_{u,i}\mathbf{K}_u\boldsymbol{\kappa}\} \leq \mathbf{b}_{u,i} \tag{17}$$

The largest value of $\boldsymbol{\kappa}$ is obtained when $\mathbf{a}_{u,i}\mathbf{K}_u$ and $\boldsymbol{\kappa}$ are parallel. This gives:

$$\mathbf{a}_{u,i}\mathbf{u} + ||\mathbf{a}_{u,i}\mathbf{K}_u||\,\rho \leq \mathbf{b}_{u,i}. \tag{18}$$

The same holds for the GRF constraints:

$$\mathbf{a}_{\lambda,i}\boldsymbol{\lambda} + ||\mathbf{a}_{\lambda,i}\mathbf{K}_\lambda||\,\rho \leq \mathbf{0}. \tag{19}$$

To remove the bilinear product, $\mathbf{K}_u\rho$ and $\mathbf{K}_\lambda\rho$ are redefined as $\mathbf{K}_u\rho = \overline{\mathbf{K}}_u$ and $\mathbf{K}_\lambda\rho = \overline{\mathbf{K}}_\lambda$. Substituting this in Eqn. 12, 18 and 19 gives:

$$\mathbf{S}\overline{\mathbf{K}}_u + \mathbf{J}_\lambda^T\overline{\mathbf{K}}_\lambda + \mathbf{J}_{\text{com}}^T\rho = \mathbf{0}, \tag{20}$$

$$\mathbf{a}_{u,i}\mathbf{u} + \left|\left|\mathbf{a}_{u,i}\overline{\mathbf{K}}_u\right|\right| \leq \mathbf{b}_{u,i} \tag{21}$$

$$\mathbf{a}_{\lambda,i}\boldsymbol{\lambda} + \left|\left|\mathbf{a}_{\lambda,i}\overline{\mathbf{K}}_\lambda\right|\right| \leq \mathbf{0}. \tag{22}$$

Due to the structure of $\mathbf{S}$ in Eqn. 20, the number of decision variables can be reduced by splitting the equation:

$$\begin{bmatrix} \mathbf{0} \\ \mathbb{I} \end{bmatrix}\overline{\mathbf{K}}_u = -\begin{bmatrix} \mathbf{J}_\lambda^{T_{\text{upp}}} \\ \mathbf{J}_\lambda^{T_{\text{legs}}} \end{bmatrix}\overline{\boldsymbol{K}}_\lambda - \begin{bmatrix} \mathbf{J}_{\text{com}}^{T_{\text{upp}}} \\ \mathbf{J}_{\text{com}}^{T_{\text{legs}}} \end{bmatrix}\rho \tag{23}$$

Where $\mathbb{I}$ is an identity matrix. From the top part an equality is obtained:

$$\mathbf{J}_\lambda^{T_{\text{upp}}}\overline{\mathbf{K}}_\lambda + \mathbf{J}_{\text{com}}^{T_{\text{upp}}}\rho = \mathbf{0} \tag{24}$$

From the lower part of Eqn. 23 an expression for $\overline{\mathbf{K}}_u$ is found, which is substituted into Eqn. 21:

$$\mathbf{a}_{u,i}\mathbf{u} + \left|\left|\mathbf{a}_{u,i}\left(-\mathbf{J}_\lambda^{T_{\text{legs}}}\overline{\mathbf{K}}_\lambda - \mathbf{J}_{\text{com}}^{T_{\text{legs}}}\rho\right)\right|\right| \leq \mathbf{b}_{u,i}. \tag{25}$$

The last constraint is that the radius should be positive:

$$\rho \geq 0. \tag{26}$$

Next, the NLP can be constructed. First, matrix $\overline{\mathbf{K}}_\lambda$ is transformed to a vector:

$${}_r\overline{\mathbf{K}}_\lambda = \begin{bmatrix} \mathbf{k}_{\lambda,1} & \mathbf{k}_{\lambda,2} & \cdots & \mathbf{k}_{\lambda,N_r} \end{bmatrix}, \tag{27}$$

where $\mathbf{k}_{\lambda,i}$ is the $i^{th}$ row of $\overline{\mathbf{K}}_\lambda$ and $N_r$ is the number of rows of $\overline{\mathbf{K}}_\lambda$. Next, the decision variables are defined:

$$\mathbf{z}_r = \begin{bmatrix} {}_r\overline{\mathbf{K}}_{\lambda,0} & \boldsymbol{\rho}_0 & {}_r\overline{\mathbf{K}}_{\lambda,1} & \boldsymbol{\rho}_1 & \cdots & {}_r\overline{\mathbf{K}}_{\lambda,N} & \boldsymbol{\rho}_N \end{bmatrix}^T, \tag{28}$$

Hereafter, the NLP can be defined as:

$$\min_{\mathbf{z}_r} \quad \frac{1}{N}\sum_{k=1}^{N} -w_\rho\rho_k \qquad \text{s.t.} \tag{29a}$$

$$\mathbf{J}_{\lambda,k}^{T_{\text{upp}}}\overline{\mathbf{K}}_{\lambda,k} + \mathbf{J}_{\text{com},k}^{T_{\text{upp}}}\rho_k = \mathbf{0} \tag{29b}$$

$$\mathbf{a}_{\lambda,i}\boldsymbol{\lambda}_k + \left|\left|\mathbf{a}_{\lambda,i}\overline{\mathbf{K}}_{\lambda,k}\right|\right| \leq \mathbf{0} \tag{29c}$$

$$\mathbf{a}_{u,i}\mathbf{u}_k + \left|\left|\mathbf{a}_{u,i}\left(-\mathbf{J}_{\lambda,k}^{T_{\text{legs}}}\overline{\mathbf{K}}_{\lambda,k} - \mathbf{J}_{\text{com},k}^{T_{\text{legs}}}\rho_k\right)\right|\right| \leq \mathbf{b}_{u,i} \tag{29d}$$

$$\rho_k \geq 0, \tag{29e}$$

where all torques, GRFs, and Jacobians are obtained from the solution found in Eqn. 9. Since the interior point method that solves Eqn. 29 uses the gradients of the constraints, the norms in Eqn. 29c and Eqn. 29d can give computational errors when the norm approaches zero. To overcome this problem, a small constant is added to the calculation of these norms in the final implementation. However, this raises a new problem when solving Eqn. 29. When the found trajectory of Eqn. 9 approaches the torque and GRF limits, the added constants in the norm cause the problem to become infeasible. To overcome this problem a slack variable $s$ is introduced. The slack allows the optimizer to find a solution even when the problem is infeasible. By adding the slack to the objective function the slack will be zero unless the optimizer cannot satisfy the constraints without the slack variable. Adding this slack leads to the following NLP, changes compared to Eqn. 29 are colored blue:

$$\min_{\mathbf{z}_r,s} \quad \frac{1}{N}\sum_{k=1}^{N} -w_\rho\rho_k + w_s s_k^2 \qquad \text{s.t.} \tag{30a}$$

$$\mathbf{J}_{\lambda,k}^{T_{\text{upp}}}\overline{\mathbf{K}}_{\lambda,k} + \mathbf{J}_{\text{com},k}^{T_{\text{upp}}}\rho_k = \mathbf{0} \tag{30b}$$

$$\mathbf{a}_{\lambda,i}\boldsymbol{\lambda}_k + \left|\left|\mathbf{a}_{\lambda,i}\overline{\mathbf{K}}_{\lambda,k}\right|\right| + s_k \leq \mathbf{0} \tag{30c}$$

$$\mathbf{a}_{u,i}\mathbf{u}_k + \left|\left|\mathbf{a}_{u,i}\left(-\mathbf{J}_{\lambda,k}^{T_{\text{legs}}}\overline{\mathbf{K}}_{\lambda,k} - \mathbf{J}_{\text{com},k}^{T_{\text{legs}}}\rho_k\right)\right|\right| + s_k \leq \mathbf{b}_{u,i} \tag{30d}$$

$$\rho_k \geq 0, \tag{30e}$$

*3) Robust Trajectory Generation*

The robust NLP formulation is a combination of the NLPs presented in Sec. II-C1 and Sec. II-C2. Combining these gives the following NLP, changes compared to Eqn. 9 are colored blue:

$$\min_{\mathbf{z},\mathbf{z}_r,s} \frac{1}{N}\sum_{k=1}^{N}(w_u\mathbf{u}_k^T\mathbf{u}_k + w_\lambda\boldsymbol{\lambda}_k^T\boldsymbol{\lambda}_k - w_\rho\rho_k + w_s s_k^2) \qquad \text{s.t.} \tag{31a}$$

$$\mathbf{x}^{\text{LB}} \leq \mathbf{x}_{k=2,\ldots,N-1} \leq \mathbf{x}^{\text{UB}} \tag{31b}$$

$$\mathbf{u}^{\text{LB}} \leq \mathbf{u}_k \leq \mathbf{u}^{\text{UB}} \tag{31c}$$

$$0 \leq \lambda_{y,k} \tag{31d}$$

$$\mathbf{x}_1^{\text{LB}} \leq \mathbf{x}_1 \leq \mathbf{x}^{\text{UB}_1} \tag{31e}$$

$$\mathbf{x}_{\text{N}}^{\text{LB}} \leq \mathbf{x}_{\text{N}} \leq \mathbf{x}_{\text{N}}^{\text{UB}} \tag{31f}$$

$$\mathbf{x}_{k+1} - \mathbf{x}_k - \frac{h_k}{2}(\mathbf{f}_k + \mathbf{f}_{k+1}) = \mathbf{0} \tag{31g}$$

$$\mathbf{J}_{\lambda,k}^{T_{\text{upp}}}\overline{\mathbf{K}}_{\lambda,k} + \mathbf{J}_{\text{com},k}^{T_{\text{upp}}}\rho_k = \mathbf{0} \tag{31h}$$

$$\mathbf{a}_{u,i}\mathbf{u}_k + \left|\left|\mathbf{a}_{u,i}\left(-\mathbf{J}_{\lambda,k}^{T_{\text{legs}}}\overline{\mathbf{K}}_{\lambda,k} - \mathbf{J}_{\text{com},k}^{T_{\text{legs}}}\rho_k\right)\right|\right| + s_k \leq \mathbf{b}_{u,i} \tag{31i}$$

$$\rho_k \geq 0 \tag{31j}$$

$$\mathbf{C}_c(\mathbf{z}) = \begin{cases} p_{y,k}^{\text{foot}} = 0 & \text{Contact} \quad (31k) \\ v_{x,k}^{\text{foot}} = 0 & \text{Contact} \quad (31l) \\ \mathbf{a}_{\lambda,i}\boldsymbol{\lambda}_k + \left|\left|\mathbf{a}_{\lambda,i}\overline{\mathbf{K}}_{\lambda,k}\right|\right| + s_k \leq \mathbf{0} & \text{Contact} \quad (31m) \\ p_{y,k}^{\text{foot}} \geq 0 & \text{No contact} \quad (31n) \\ \boldsymbol{\lambda}_k = \mathbf{0} & \text{No contact} \quad (31o) \\ \overline{\mathbf{K}}_{\lambda,k} = \mathbf{0} & \text{No contact.} \quad (31p) \end{cases}$$

## III. Evaluation Technique

To assess the influence of optimizing robustness, the system needs to be disturbed. To be able to disturb the system with the SUF, the direction has to be determined as the optimizer only returns the magnitude. The method to determine the SUF direction will be discussed first. Hereafter, the method of disturbance rejection will be discussed.

### A. Determine SUF Direction

The NLP returns the magnitude of the SUF, but not the direction of it. The direction of the SUF is where the ball touches the constraints. Consider Eqn. 13. The normalised vector perpendicular to this constraint $\boldsymbol{\xi}$ is given by:

$$\boldsymbol{\xi} = \frac{\mathbf{K}_u^T \mathbf{a}_{u,i}^T}{\left|\left|\mathbf{K}_u^T \mathbf{a}_{u,i}^T\right|\right|}. \tag{32}$$

Substituting $\boldsymbol{\xi}$ into Eqn. 13 gives:

$$\mathbf{a}_{u,i}\left(\mathbf{u} + \mathbf{K}_u \boldsymbol{\xi} \boldsymbol{\rho}\right) \leq \mathbf{b}_{u,i}. \tag{33}$$

The SUF direction is found when

$$\mathbf{a}_{u,i}\left(\mathbf{u} + \mathbf{K}_u \boldsymbol{\xi} \boldsymbol{\rho}\right) - \mathbf{b}_{u,i} = \mathbf{0}. \tag{34}$$

The same can be applied to GRF constraints, which gives:

$$\mathbf{a}_{\lambda,i}\left(\boldsymbol{\lambda} + \mathbf{K}_\lambda \boldsymbol{\xi} \boldsymbol{\rho}\right) = \mathbf{0}. \tag{35}$$

When the solution of Eqn. 31 is substituted in Eqn. 34 or Eqn. 35 and it holds, the SUF with its direction is found. The SUF itself is then given by:

$$\text{SUF} = \boldsymbol{\xi}\rho. \tag{36}$$

### B. Disturbance Rejection

To investigate the disturbance reaction a new NLP is created, as this resembles one timestep of an MPC control scheme and can show how the system would behave when implemented in an MPC framework. An ideal observer is assumed as the optimizer knows the exact disturbance force. From the obtained trajectories from the previous sections, a point in time is chosen to apply a disturbance. This point is now referred to as the instance of disturbance (IoD). In this instance, a linear disturbance force is applied to the CoM. From that instance, the trajectory to the initial goal is calculated again, with the original solution from that point as an initial guess. To simulate the real-life behavior, the optimizer cannot adjust the torques, positions, and velocities at the first knot point. So they have to be the same as the initial guess at the IoD. This leads to the following NLP, changes compared to Eqn. 31 are colored blue:

$$\begin{aligned}
\min_{\mathbf{z},\mathbf{z}_r,s} \quad & \frac{1}{N - \text{IoD}} \sum_{k=\text{IoD}}^{N} \left(w_u \mathbf{u}_k^T \mathbf{u}_k + w_\lambda \boldsymbol{\lambda}_k^T \boldsymbol{\lambda}_k \right. \\
& \left. - w_\rho \rho_k + w_s s_k^2\right) \quad \text{s.t.} && \text{(37a)} \\
& \mathbf{x}^{\text{LB}} \leq \mathbf{x}_{k=\text{IoD}+1,\ldots,N-1} \leq \mathbf{x}^{\text{UB}} && \text{(37b)} \\
& \mathbf{x}^{\text{LB IoD}} \leq \mathbf{x}_{\text{IoD}} \leq \mathbf{x}^{\text{UB IoD}} && \text{(37c)} \\
& \mathbf{x}_{\text{N}}^{\text{LB}} \leq \mathbf{x}_{\text{N}} \leq \mathbf{x}_{\text{N}}^{\text{UB}} && \text{(37d)} \\
& 0 \leq \lambda_{y,k} && \text{(37e)} \\
& \mathbf{u}^{\text{LB}} \leq \mathbf{u}_k \leq \mathbf{u}^{\text{UB}} && \text{(37f)} \\
& \mathbf{u}^{\text{LB IoD}} \leq \mathbf{u}_{\text{IoD}} \leq \mathbf{u}^{\text{UB IoD}} && \text{(37g)} \\
& \mathbf{x}_{k+1} - \mathbf{x}_k - \frac{h_k}{2}(\mathbf{f}_k + \mathbf{f}_{k+1}) = \mathbf{0} && \text{(37h)} \\
& \mathbf{J}_{\lambda,k}^{T_{\text{upp}}} \overline{\mathbf{K}}_{\lambda,k} + \mathbf{J}_{\text{com},k}^{T_{\text{upp}}} \rho_k = \mathbf{0} && \text{(37i)} \\
& \mathbf{a}_{u,i}\mathbf{u}_k + \left|\left|\mathbf{a}_{u,i}\left(-\mathbf{J}_{\lambda,k}^{T_{\text{legs}}} \overline{\mathbf{K}}_{\lambda,k} - \mathbf{J}_{\text{com},k}^{T_{\text{legs}}} \rho_k\right)\right|\right| \\
& \quad + s_k \leq \mathbf{b}_{u,i} && \text{(37j)} \\
& \rho_k \geq 0 && \text{(37k)}
\end{aligned}$$

$$\mathbf{C}_c(\mathbf{z}) = \begin{cases}
p_{y,k}^{\text{foot}} = 0 & \text{Contact} \quad \text{(37l)} \\
v_{x,k}^{\text{foot}} = 0 & \text{Contact} \quad \text{(37m)} \\
\mathbf{a}_{\lambda,i}\boldsymbol{\lambda}_k + \left|\left|\mathbf{a}_{\lambda,i}\overline{\mathbf{K}}_{\lambda,k}\right|\right| + s_k \leq \mathbf{0} & \text{Contact} \quad \text{(37n)} \\
p_{y,k}^{\text{foot}} \geq 0 & \text{No contact} \quad \text{(37o)} \\
\boldsymbol{\lambda}_k = \mathbf{0} & \text{No contact} \quad \text{(37p)} \\
\overline{\mathbf{K}}_{\lambda,k} = \mathbf{0} & \text{No contact.} \quad \text{(37q)}
\end{cases}$$

### C. Software

The NLP is constructed in MATLAB 2022b. CasADi was used to automatically compute the gradients of the NLP [13]. Finally, IPOPT was used to solve the NLP [14].

## IV. Results

To evaluate the generated trajectories and thereby also evaluate the performance of the proposed method, the behavior is studied during two tasks. One task is a standing task in which the robot has to stand and move its upper body 6 cm forward. During the other task, the robot has to walk 2 m in 5 steps at predefined contact instances. First, the overall robustness is determined for different $w_\rho$ during a standing and walking task. This shows the relation between the weight on robustness and the increase in robustness. To see how a more robust trajectory is achieved by the optimizer, the differences between robust and non-robust trajectories for both tasks will be evaluated. This is done by comparing the torques, GRFs, and trajectories.

Besides investigating the robustness of the planned trajectories, it is also important to assess the difference in behavior when disturbed. The SUF defines the maximum force in any direction that can be rejected without changing the originally planned trajectory. However, the robot can also encounter disturbances larger than this force. Therefore the extent to

which the system can handle forces larger than the SUF will be tested as well. Last, the robustness after a disturbance is evaluated to see the influence of optimizing for robustness in a disturbance rejection strategy.

### A. *Influence of the Weight on Robustness*

Two sets of trajectories are generated, one for the standing task, and the other for the walking task. The average $\rho$ over the whole trajectory for different weights on the robustness is shown in Fig. 3. In general, the robustness is higher when a higher weight on the robustness is applied, with an exception for no weight on robustness when walking. If the weights get too high the algorithm sometimes fails to converge. The optimum between robustness and steady convergence is roughly around a weight of 1–2. It is also notable that there is only a small increase in overall robustness during walking for higher weights on the robustness compared to the increase during the standing task. For small weights, it even seems to barely influence the overall robustness. This is probably because, during double support, the optimizer can adjust the stance to increase robustness, which is not possible during a single stance, this higher robustness during double stance is visible in Fig. 8.

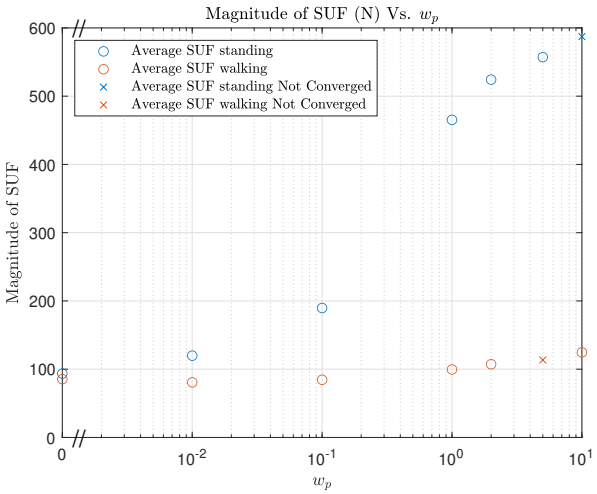


Fig. 3: Average $\rho$ for standing and walking tasks with different weights.

### B. *Non-Robust Vs. Robust Trajectories*

In this subsection, the difference between non-robust ($w_\rho = 0$) and robust ($w_\rho = 1$) trajectories will be discussed for both tasks. First, the standing task will be discussed, after this the walking task with humanlike contact sequences and the walking task with longer contact sequences, more like exoskeletons .

#### 1) *Standing*

Fig. 4 shows the magnitude of the SUF during the standing task. The robustness is higher over the whole trajectory when it is optimized for robustness.

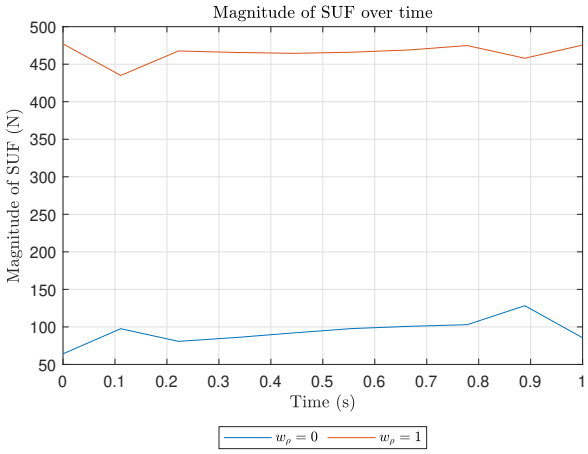


Fig. 4: Magnitude of the SUF during non-robust and robust while the upper body has to move 6 cm forwards.

Fig. 5 shows 3 snapshots of the standing tasks of the non-robust and robust trajectory. In the figure can be seen that the optimizer widens the stance to increase the robustness. Also the GRFs point 'inwards' to be more resilient against disturbances. This wider stance does increase the joint torques. This is shown in Fig. 6.

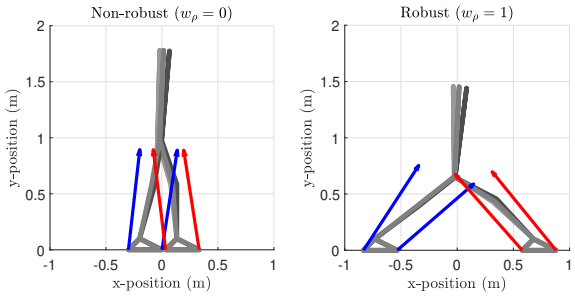


Fig. 5: Snapshots of the stance and GRFs during the standing task. On the left the snapshot of the non-robust optimization is shown, and on the right of the robust optimization. The blue arrows show the normalized GRFs on the left foot, and the red arrows show the normalized GRFs on the right foot. Note that these GRFs are only shown for t = 0.44s

Fig. 7 shows the GRFs for each contact point over time for both non-robust and robust trajectories. It shows that during the non-robust trajectory, the x-direction of the GRF on each contact point is close to zero. For the robust trajectory the GRFs on the left foot are increased in the x-direction and on the right foot the force is increased in the negative x-direction. This could also be seen in Fig. 5. Fig. 7 also shows that there is a higher GRF on the left heel during the start of the trajectory and a higher GRF on the right toe at the end of the trajectory. This means that the weight distribution has shifted during the movement. This happens to a lesser extent during the robust optimization, so the weight is distributed more evenly over the contact points during the whole trajectory.

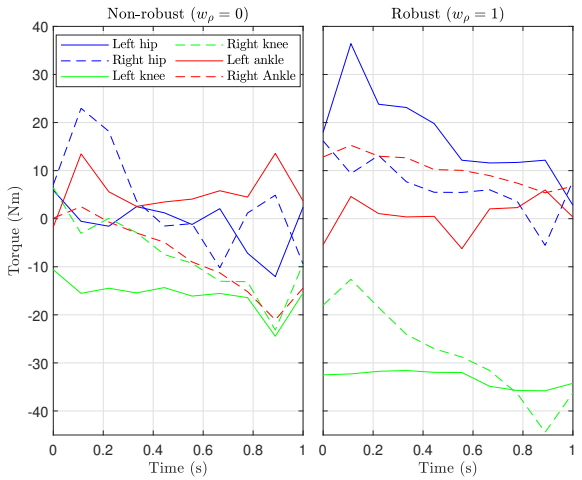


Fig. 6: Torques for each joint during non-robust and robust while the upper body has to move 6 cm forwards.

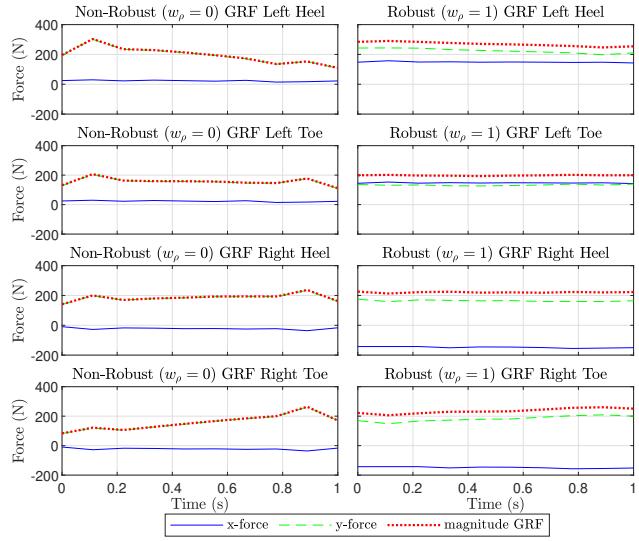


Fig. 7: GRFs over time for each contact point during the standing task for both non-robust and robust optimization.

*2) Human-Like Walking*

Fig. 8 shows the magnitude of the SUF during the walking task. It shows that the magnitudes are very similar during the non-robust and robust trajectories, with the exception that in the robust trajectory, the magnitude of the SUF is on average slightly higher.

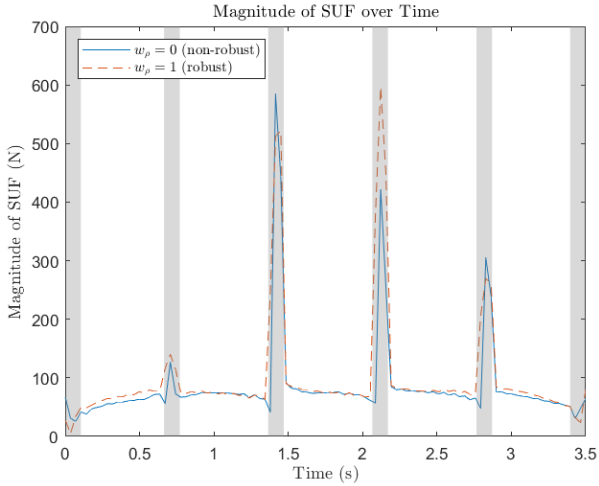


Fig. 8: Magnitude of the SUF during non-robust and robust while walking 2 meters in 5 steps. The grey areas indicate double stance

This similarity is also visible in the trajectories, shown in Fig. 9. The trajectories are almost identical, there are only small differences in foot placement. The GRF and joint torques

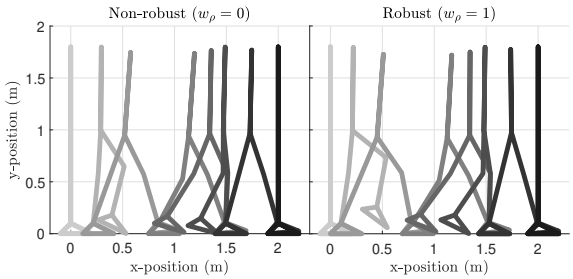


Fig. 9: Snapshots of the configurations during the walking of 2 meters in five steps at different time instances. On the left the non-robust walking is shown on the right the robust walking. The lightest gray shade is the initial position and black is the final position.

also show similarity. The joint torques, shown in Fig. 10, the shape of the trajectories is the same, only during the robust trajectory, the torques are slightly higher. The GRFs follow a similar pattern during non-robust and robust trajectories, with the only exception that during robust optimization there is a higher spike in forces when making or breaking contact. This can be seen in Fig. 11

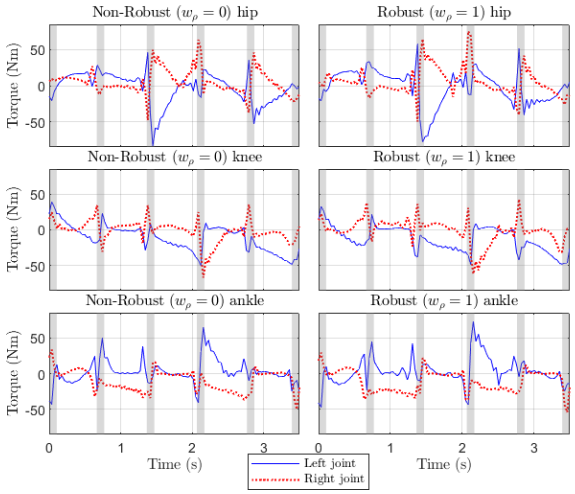


Fig. 10: Torques for each joint during non-robust and robust walking. The gray areas indicate the double stance phases.

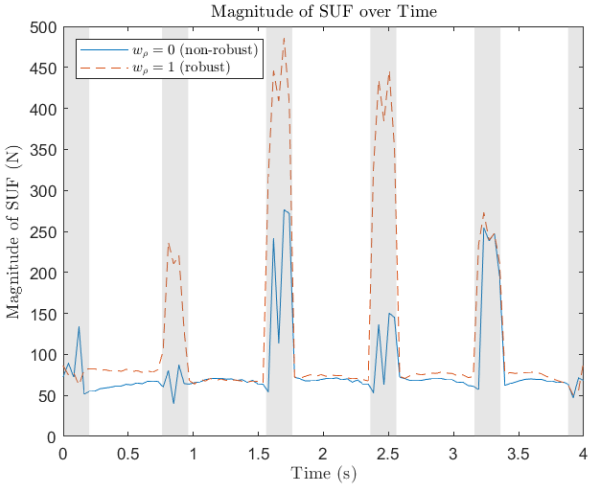


Fig. 12: Magnitude of the SUF during non-robust and robust while walking 1.5 meters in 5 steps. The grey areas indicate double stance

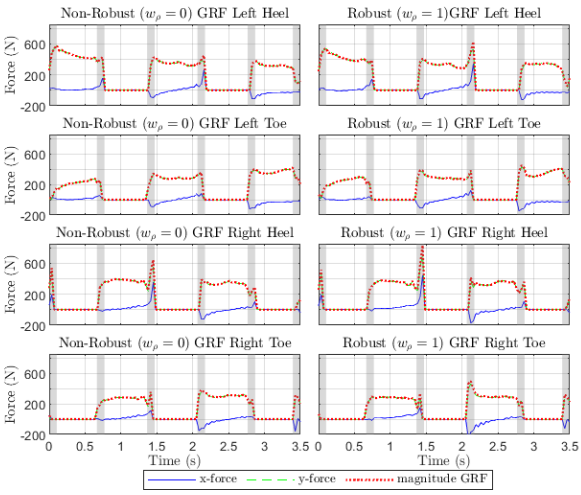


Fig. 11: GRFs over time for each contact point during the walking task for both non-robust and robust optimization. The gray areas indicate the double stance phases.

### 3) Walking Longer Double Stance

Exoskeletons are currently not capable of walking using human contact sequences. They have longer double stance phases when walking. Therefore, it is useful to assess the behavior of the optimization on a more practical contact sequence. Due to the longer double-contact phases, the robot can cover less distance. Fig. 12 show the radius over time with 30% double contact phase. It shows that during the double stance phase, the magnitude of the SUF is higher for the robust trajectory than for the non-robust.

Fig. 13 shows the trajectories for the non-robust and the robust trajectories. The non-robust trajectory takes 5 small steps to cover 1.5 meters. The robust trajectory takes first a small step back to be able to make two larger steps that are more robust.

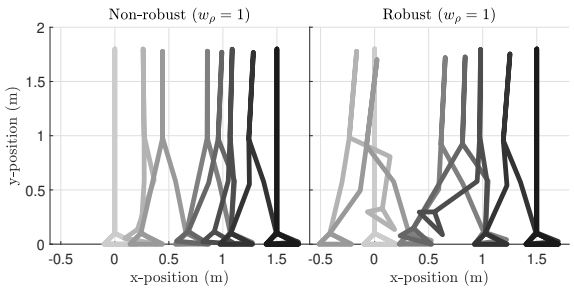


Fig. 13: Snapshots of the configurations during the walking of 1.5 meters in five steps at different time instances. On the left, the non-robust walking is shown on the right the robust walking. The lightest gray shade is the initial position and black is the final position.

## C. Disturbance rejection

This subsection shows the behavior of the system when disturbed. A disturbance force is applied to the robot during two trajectories in the walking task. As stated before, a linear disturbance force is applied to the robot while walking. This disturbance is applied in single stance phase or double stance phase. The instances of the disturbances are shown in Fig. 14. Multiple tests will be performed to assess the difference in the reaction to a disturbance between a non-robust and robust initial trajectory, and the difference between a non-robust and robust reaction to the disturbance. The latter is important when the optimization is implemented in an MPC framework.

Since the SUF itself is defined as the smallest that the robot cannot counteract without changing its trajectory, it is interesting to see the difference in the handling of disturbances

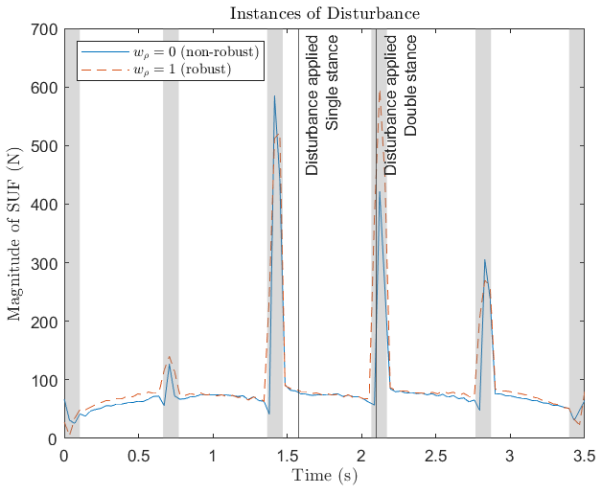


Fig. 14: Magnitude of the SUF during robust and non-robust trajectories with points of disturbance.

larger than the SUF. Note that in this case, the robot has to adjust its trajectory.

To make sure that the force is outside the feasible polytope, the disturbance force is applied in the direction of the SUF. It is tested on what magnitude of force the optimizer can find a feasible solution. The results are shown in Tab. I. To have a fair comparison the forces applied at the non-robust trajectory are in the same direction as the force applied at the robust trajectory. The table shows that in single stance the maximum disturbance that can be handled is slightly larger in single stance and more than double as large in double stance phase. This is similar to the behavior of the SUF which is also higher in double stance.

TABLE I: Maximum perturbation in SUF direction that can be rejected using robust rejection strategy.

| Trajectory | Max perturbation force (N) | |
|---|---|---|
| | Single stance | Double stance |
| Non-robust | 1500 | 1600 |
| Robust | 1600 | 3700 |

The difference in reaction of a non-robust and robust rejection strategy to a disturbance is tested on the robust walking trajectory. A disturbance of 0.9 times the SUF is applied during single stance and double stance. Note that these disturbances are applied in separate simulations. Fig. 15 shows the SUFs over time for each disturbance for both a non-robust and robust rejection strategy. The difference in reaction to a disturbance is in line with the difference between non-robust and robust trajectory generation. The non-robust rejection strategy leads to slightly lower SUFs after the disturbance than the robust rejection strategy. This holds for disturbance during single stance and double stance.

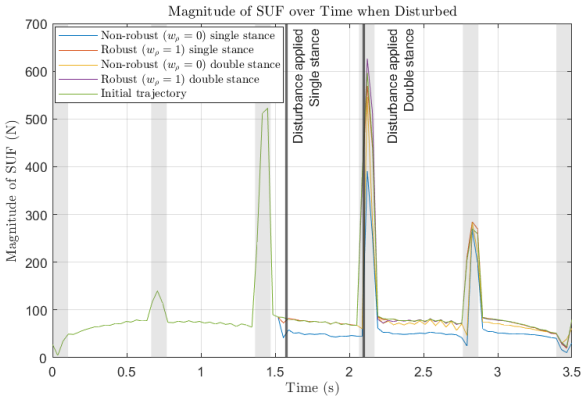


Fig. 15: Magnitude of the SUF during robust and non-robust rejection strategies at two different instances of disturbance.

## V. Discussion

The goal of this work was to implement an optimization algorithm that would generate trajectories that are robust against disturbances on the CoM. The results show that during the standing task, the robustness is greatly increased when optimized for it. During walking the increase in robustness is smaller. The effect of the optimization for robustness is more clear in the walking with a longer double stance phase. When there is a longer double stance phase the optimizer takes a larger step which results in a wider stance, which is more robust. Ferrolho et al. [9] showed that their quadruped adjusted its stance to increase the robustness. This work shows that this also holds for bipedal robots when standing. As an addition, this work shows that there is also an increase in robustness against forces larger than the SUF. In the single contact phase or swing phase, there was only a small increase in SUF. This is expected as in single contact the robot is inherently not robust against disturbances. Orsoline et al. [6] were able to perform a more robust gait, also during the swing phase. However, it was performed on a quadruped so it could adjust it stance of the other three legs while walking to improve robustness. This cannot be done when walking with a bipedal robot.

Besides an increase in the SUF, there is also an increase in disturbance rejection for disturbances greater than the SUF. The results showed, during both single and double stances, improvement in the resistance against perturbations. The results also showed an increase in robustness after encountering a perturbation when optimizing for robustness. This is beneficial when multiple disturbances are encountered sequentially.

However, there are also some limitations. First of all, a planar model is used. This can influence the results of the robustness, especially during double stance and standing. In these scenarios, a wider stance is used to increase robustness. In a three-dimensional scenario, this would probably not be as effective. A wider stance in the anteroposterior direction does not increase robustness in the mediolateral direction and vice versa. Still, it would be beneficial to implement robustness optimization in a 3D model. It will improve the robustness and a slightly more robust stance can be the difference between falling and maintaining balance when disturbed.

If 3D simulations yield positive results, another limitation has to be overcome before it can be implemented in practice. This limitation is the computational time. At this point, the most time-consuming step is generating the initial guess for the robust NLP. This can take up to 5 minutes when planning a trajectory for the walking task. By using an educated guess instead of 2 optimizations, the computational time can decrease tremendously. However, the total algorithm would still be too slow to implement online. The robust optimization also takes several minutes while ideally, the algorithm would run at 100Hz. To be able to implement the optimization online, other methods could be investigated. For instance, learning the behavior of the optimizer to a neural network could be a way to reach online implementation as this can run at the required speed.

Also, some useful extension of the algorithm would be to remove the dependence of the pre-specified foot contact timings. This increases the freedom of the optimizer and therefore it could improve robustness during both planning the trajectory and reacting to a disturbance. Methods as performed by Van Gils can be considered for this [15]. However, this would most likely lead to an increase in computation time.

## VI. Conclusion

This work presents an adjusted method to describe and optimize robustness for bipeds. Robustness is defined as the Chebyshev radius of the polytope defined by the GRF and torque constraints. This radius is mapped to a force on the CoM, which results in the SUF on the COM. This type of optimization has two advantages compared to non-robust optimization. First, a robust trajectory leads to a higher disturbance resistance, which means the system can handle larger disturbances before failing the desired task. Second, incorporating robustness in disturbance handling leads to a higher robustness after a disturbance. This is beneficial when multiple disturbances are encountered.

For future work, the algorithm should be implemented in a model predictive control framework. This way real-world behavior can be simulated and the algorithm can be fully validated. If these results are positive, the computational time should decrease to be able to implement it in an online model predictive control framework.

## References


[1] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, Nov. 2021.

[2] S. Caron, Q.-C. Pham, and Y. Nakamura, *Stability of Surface Contacts for Humanoid Robots: Closed-Form Formulae of the Contact Wrench Cone for Rectangular Support Areas*, Jan. 2015.

[3] S. Samadi, J. Roux, A. Tanguy, S. Caron, and A. Kheddar, "Humanoid Control Under Interchangeable Fixed and Sliding Unilateral Contacts," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, Apr. 2021.

[4] Y. Zheng, S. W. Liao, and K. Yamane, "Humanoid Locomotion Control and Generation Based on Contact Wrench Cones," *International Journal of Humanoid Robotics*, vol. 16, Oct. 2019.

[5] M. G. Navaneeth, A. P. Sudheer, and M. L. Joy, "Contact Wrench Cone-Based Stable Gait Generation and Contact Slip Estimation of a 12-DoF Biped Robot," *Arabian Journal for Science and Engineering*, vol. 47, Dec. 2022.

[6] R. Orsolino, M. Fiocchi, 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, Oct. 2018.

[7] W. Wolfslag, C. McGreavy, G. Xin, C. Tiseo, S. Vijayakumar, and Z. Li, *Optimisation of Body-ground Contact for Augmenting Whole-Body Loco-manipulation of Quadruped Robots*, Feb. 2020.

[8] Y. Ding, C. Li, and H.-W. Park, "Kinodynamic Motion Planning for Multi-Legged Robot Jumping via Mixed-Integer Convex Program," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020.

[9] H. Ferrolho, W. Merkt, V. Ivan, W. Wolfslag, and S. Vijayakumar, "Optimizing Dynamic Trajectories for Robustness to Disturbances Using Polytopic Projections," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA: IEEE, Oct. 2020.

[10] H. Ferrolho, V. Ivan, W. Merkt, I. Havoutis, and S. Vijayakumar, "RoLoMa: Robust Loco-Manipulation for Quadruped Robots with Arms," Mar. 2022.

[11] M. P. Kelly, *Transcription Methods for Trajectory Optimization: A beginners tutorial*, Jul. 2017.

[12] A. Kharb, V. Saini, Y. K. Jain, and S. Dhiman, "A REVIEW OF GAIT CYCLE AND ITS PARAMETERS," 2011.

[13] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2019.

[14] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, Mar. 2006.

[15] H. van Gils, "Simultaneous optimization of contacts and smooth movements for control of humanoid robotics," University of Twente, M.S. thesis, 2023.

# 4 General Discussion and Conclusion

This section will discuss the outcome of this research. First, some shortcomings and possible improvements will be highlighted. Next, the steps to real-world implementation will be discussed. It ends with a conclusion on what extent the goal of the thesis is reached.

## 4.1 Improvements and Additions to the Current Algorithm

Although the algorithm performs well, some issues could improve the performance if resolved. First, it should be investigated whether second-order trapezoidal collocation can be used instead of first-order trapezoidal collocation. According to literature, second-order trapezoidal collocation leads to smaller dynamical errors in similar computation times [10]. The second-order method was implemented in the algorithm, but it led to unwanted behavior in the program. This resulted in the robot moving from position between two collocation points, while the velocity on both collocation points was zero. This is due to the nature of second-order collocation, where a second-order polynomial approximates the velocities (see Appendix B for the exact equations used in second-order trapezoidal collocation). This made it possible that the velocity can be zero on the collocation point, but the integral of this could be non-zero. This resulted in a position change. The problem was tried to be solved using extra constraints on the position, where the foot position was not allowed to move during contact. This led to poor convergence of the optimizer. It should be investigated if revising the contact dynamics and constraints can lead to the successful implementation of second-order trapezoidal collocation.

Also, during the walking task, robustness is higher for no weight on the robustness than for low weight on the robustness. This is contrary to what is expected. It could be explained by the fact that the solver has to solve a 'different' problem, as it uses other objectives and constraints. Besides, another initial guess is used. This could also lead to another local optimum.

A useful extension of the algorithm would be to remove the dependence on the pre-specified foot contact timings. Currently, the used contact timings are based on the contact timings on human contact timings. This is sufficient when planning a trajectory. However, when disturbed, the same contact sequence is used as it would when not disturbed. This could lead to suboptimal behavior, as having contact sooner or later could be beneficial to counteract the disturbance. Removing the dependence on contact timings, the freedom of the optimizer increases and therefore it could improve robustness. Methods as performed by Van Gils can be considered for implementing this in the NLP [11].

## 4.2 Steps to Real World Implementation

To be able to implement the algorithm into a real-world application, the motion plan should be executed. A logical step is to implement this into a model predictive control framework, as this is also based on optimal control. It often uses collocation but with a receding horizon. Using model predictive control, the behavior of the algorithm can be planned while it is run in real-time. If this yields positive results, the model should be extended to a three-dimensional model. The last step would be to decrease the computational time to be able to implement the algorithm online. In the current version, three optimizations are done to obtain a robust trajectory, of which 2 are used to obtain an initial guess for the robust trajectory. Instead of using two separate optimizations, it could be better to use an estimated guess of the trajectory to improve the computation time. Although this would greatly improve the computational time of robust trajectories, it will not be fast enough to be implemented online. A solution to this could be to learn the behavior of the algorithm to a neural network, which then can be implemented in an online control framework.

## 4.3 Conclusion

This thesis contributes by showing that optimizing for robustness can improve the stability of biped robots. The level of robustness against disturbances is defined as the Chebyshev radius of the polytope defined by the GRF and torque constraints. This radius is mapped to a force on the CoM, which results in the SUF

on the COM. The results showed that optimizing for robustness leads to a higher average SUF over the total trajectory in all tested situations. This means that the robot can withstand higher forces on the CoM without replanning the trajectory. This is beneficial for planning trajectories for exoskeletons as the user can generate larger forces without influencing the trajectory of the exoskeleton. This work also showed that optimizing for robustness leads to the rejection of higher disturbances in situations where the optimizer can adjust the trajectory. This could also improve the stability of exoskeletons since the disturbances coming from the user can be larger than the SUF. Last, this work showed that optimizing for robustness when handling a disturbance leads to higher robustness in the rest of the trajectory. This is advantageous when multiple disturbances are encountered sequentially.

# References


[1] *Spinal cord injury*, en. [Online]. Available: `https://www.who.int/news-room/fact-sheets/detail/spinal-cord-injury` (visited on 11/30/2023).

[2] M. Wyndaele and J.-J. Wyndaele, "Incidence, prevalence and epidemiology of spinal cord injury: What learns a worldwide literature survey?", en, *Spinal Cord*, vol. 44, no. 9, Sep. 2006.

[3] C. L. Flemmer and R. C. Flemmer, "A review of manual wheelchairs", *Disability and Rehabilitation. Assistive Technology*, vol. 11, no. 3, 2016.

[4] R. B. van Dijsseldonk, I. J. W. van Nes, A. C. H. Geurts, and N. L. W. Keijsers, "Exoskeleton home and community use in people with complete spinal cord injury", en, *Scientific Reports*, vol. 10, no. 1, Sep. 2020.

[5] C. Hu, S. Xie, L. Gao, S. Lu, and J. Li, "An overview on bipedal gait control methods", *IET Collaborative Intelligent Manufacturing*, vol. 5, no. 3, Sep. 2023.

[6] 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", en, *IEEE Robotics and Automation Letters*, vol. 3, no. 4, Oct. 2018.

[7] M. Kelly, "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation", en, *SIAM Review*, vol. 59, no. 4, Jan. 2017.

[8] S. Stahl and C. Stenson, *Introduction to topology and geometry* (Pure and applied mathematics), en, 2nd edition. Hoboken, New Jersey: Wiley, 2013.

[9] L. Zhao and W.-S. Lu, "Chebyshev Centre of a Polyhedron",

[10] S. Moreno-Martın, L. Ros, and E. Celaya, "Collocation Methods for Second Order Systems",

[11] H. van Gils, "Simultaneous optimization of contacts and smooth movements for control of humanoid robotics", University of Twente, M.S. thesis, 2023.

[12] N. I. M. Gould, D. Orban, A. Sartenaer, and P. L. Toint, "Superlinear Convergence of Primal-Dual Interior Point Algorithms for Nonlinear Programming", *SIAM Journal on Optimization*, vol. 11, no. 4, Jan. 2001.

[13] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming", *Mathematical Programming*, vol. 106, no. 1, Mar. 2006.

[14] N. Andrei, *Modern Numerical Nonlinear Optimization* (Springer Optimization and Its Applications). Cham: Springer International Publishing, 2022, vol. 195.

# A First-Order Trapezoidal Collocation Constraint Derivation

In first-order collocation, the integration is approximated by a second-order polynomial [7]:

$$x(\tau) = a + b\tau + c\tau^2 \tag{1a}$$
$$\dot{x}(\tau) = b + 2c\tau. \tag{1b}$$

Three unknown coefficients need to be found, so three known conditions are used:

$$x(0) = x_k \tag{2a}$$
$$\dot{x}(0) = \dot{x}_k \tag{2b}$$
$$\dot{x}(h_k) = \dot{x}_{k+1}. \tag{2c}$$

Here $h_k$ is the timestep between the collocation points. By substituting Eqn. (2) into Eqn. (1), the following expressions for the coefficients are obtained:

$$a = x_k \tag{3a}$$
$$b = \dot{x}_k \tag{3b}$$
$$c = \frac{1}{2h_k}(\dot{x}_{k+1} - \dot{x}_k). \tag{3c}$$

Substituting these coefficients into the interpolation polynomials gives, note that $\dot{x}_k = f_k$ and $\dot{x}_{k+1} = f_{k+1}$:

$$x(\tau) = x_k + f_k\tau + \frac{\tau^2}{2h_k}\left(f_{k+1} - f_k\right), \tag{4a}$$
$$\dot{x}(\tau) = f_k + \frac{\tau}{h_k}\left(f_{k+1} - f_k\right) \tag{4b}$$

To find the collocation constraints, the interpolation polynomial is evaluated at $x_{k+1} = x(h_k)$:

$$x_{k+1} = x_k + f_k h_k + \frac{h_k^2}{2h_k}\left(f_{k+1} - f_k\right), \tag{5a}$$
$$x_{k+1} = x_k + f_k h_k + \frac{h_k}{2} f_{k+1} - \frac{h_k}{2} f_k, \tag{5b}$$
$$x_{k+1} = x_k + \frac{h_k}{2}\left(f_k + f_{k+1}\right). \tag{5c}$$

# B Second-Order Trapezoidal Collocation Constraint Derivation

For second-order trapezoidal collocation, the interpolating polynomial of $q(\tau)$ is of order three [10]. So for a given interval, the polynomials for position, velocity, and acceleration are given by:

$$q(\tau) = a + b\tau + c\tau^2 + d\tau^3 \tag{6a}$$
$$\dot{q}(\tau) = b + 2c\tau + 3d\tau^2 \tag{6b}$$
$$\ddot{q}(\tau) = 2c + 6d\tau. \tag{6c}$$

To find the coefficients $a, b, c, d$, four conditions need to be imposed:

$$q(0) = q_k \tag{7a}$$
$$\dot{q}(0) = \dot{q}_k \tag{7b}$$
$$\ddot{q}(0) = \ddot{q}_k \tag{7c}$$
$$\ddot{q}(h_k) = \ddot{q}_{k+1}. \tag{7d}$$

Here $h_k$ is the timestep between the collocation points. By substituting Eqn. (7) into Eqn. (6), the following expressions for the coefficients are obtained:

$$a = q_k \tag{8a}$$
$$b = \dot{q}_k \tag{8b}$$
$$c = \frac{1}{2}\ddot{q}_k \tag{8c}$$
$$d = \frac{1}{6h_k}(\ddot{q}_{k+1} - \ddot{q}_k). \tag{8d}$$

Substituting these coefficients into the interpolation polynomials gives:

$$q(\tau) = q_k + \dot{q}_k\tau + \frac{\tau^2}{2}\ddot{q}_k + \frac{\tau^3}{6h_k}(\ddot{q}_{k+1} - \ddot{q}_k) \tag{9a}$$
$$\dot{q}(\tau) = \dot{q}_k + \ddot{q}_k\tau + \frac{\tau^2}{2h_k}(\ddot{q}_{k+1} - \ddot{q}_k) \tag{9b}$$
$$\ddot{q}(\tau) = \ddot{q}_k + \frac{\tau}{h_k}(\ddot{q}_{k+1} - \ddot{q}_k). \tag{9c}$$

To find the collocation constraints, evaluate $q_{k+1} = q(h_k)$ and $\dot{q}_{k+1} = \dot{q}(h_k)$. When simplified this gives:

$$q_{k+1} = q_k + \dot{q}_k h_k + \frac{h_k^2}{6}(\ddot{q}_{k+1} + 2\ddot{q}_k) \tag{10a}$$
$$\dot{q}_{k+1} = \dot{q}_k + \frac{h_k}{2}(\ddot{q}_{k+1} + \ddot{q}_k). \tag{10b}$$

# C Interior Point Method for Nonlinear Programming

In this section, a brief description will be given about the mathematical idea behind the working of interior point solvers for nonlinear programs. Interior point solvers are made to solve problems with the following structure:

$$\underset{\mathbf{z}}{\min} \quad \mathbf{J}(\mathbf{z}) \quad \text{s.t.} \tag{11a}$$
$$\mathbf{c}(\mathbf{z}) = \mathbf{0} \tag{11b}$$
$$\mathbf{z} \geq 0. \tag{11c}$$

To be able to solve the nonlinear problem as given in sec: 2.1, the constraints need to be rewritten. This is done using a slack variable, which is here notated as $\mathbf{s}$. $\mathbf{s}$ is defined as:

$$\mathbf{s} \equiv \mathbf{g}(\mathbf{z}) - \mathbf{b} \tag{12}$$

This gives:

$$\left.\begin{array}{r} \mathbf{f}(\mathbf{z}) = \mathbf{0} \\ \mathbf{g}(\mathbf{z}) - \mathbf{b} - \mathbf{s} = \mathbf{0} \end{array}\right\} \quad \mathbf{c}(\mathbf{z}) = \mathbf{0} \tag{13a}$$
$$\left.\begin{array}{r} \mathbf{z} \geq \mathbf{0} \\ \mathbf{s} \geq \mathbf{0} \end{array}\right\} \quad \mathbf{z} \geq \mathbf{0}, \tag{13b}$$

which is the standard form used for the interior point method.

The next step is to write the inequality constraints on the decision variables into the objective function. This is done using a barrier function:

$$\underset{\mathbf{z}}{\min} \quad \mathbf{J}(\mathbf{z}) - \mu \sum_{i=1}^{n} \ln(z_i) \tag{14a}$$
$$\text{s.t.} \quad \mathbf{c}(\mathbf{z}) = \mathbf{0}. \tag{14b}$$

A natural logarithm is used as a barrier function since this is not defined for inputs smaller than or equal to zero. However, the value increases rapidly when the input of the barrier function approaches zero. This leads to solutions that lie further for the constraints, while this might not be the optimal solution to the initial problem. To overcome this the variable $\mu$ is introduced. For small $\mu$ the barrier function gets "steeper", therefore a small $\mu$ is preferred. To find a solution the NLP is not solved directly, but a sequence of barrier problems with decreasing positive $\mu$ values [12]. The optimal solution is found when the solution satisfies the so-called primal-dual equations [13]:

$$\nabla \mathbf{J}(\mathbf{z}) + \nabla \mathbf{c}(\mathbf{z})\boldsymbol{\lambda} - \mathbf{y} = 0 \tag{15a}$$

$$\mathbf{c}(\mathbf{z}) = \mathbf{0} \tag{15b}$$

$$\mathbf{ZYe} - \mu \mathbf{e} = 0, \tag{15c}$$

where $\lambda$ are the Lagrange multipliers, $y_i = \frac{\mu}{x_i}$, $\mathbf{Z} = diag(\mathbf{z})$, $\mathbf{Y} = diag(\mathbf{y})$ and $\mathbf{e}$ is a row vector of ones to make the dimensions work. For $\mu = 0$ and $\mathbf{y}, \mathbf{z} \geq 0$, the equations of Eqn. (15) are called the Karush–Kuhn–Tucker (KKT) conditions. To solve the posed barrier problem in Eqn. (14), a damped Newton's method is applied to the primal-dual equations in Eqn. (15) For more details on the damped Newton's method, see sec: C.1. Applying Newton's method on Eqn. (15), the following equation is obtained:

$$\begin{bmatrix} \mathbf{W}_k & \mathbf{A}_k & -\mathbf{I} \\ \mathbf{A}_k^T & 0 & 0 \\ \mathbf{Y}_k & 0 & \mathbf{Z}_k \end{bmatrix} \begin{pmatrix} \mathbf{d}_k^z \\ \mathbf{d}_k^\lambda \\ \mathbf{d}_k^y \end{pmatrix} = - \begin{pmatrix} \nabla \mathbf{f}\left(\mathbf{z}_k\right) + \mathbf{A_k}\boldsymbol{\lambda}_k - \mathbf{y}_k \\ \mathbf{c}\left(\mathbf{z}_k\right) \\ \mathbf{Z}_k\mathbf{Y}_k\mathbf{e} - \mu_j\mathbf{e} \end{pmatrix}, \tag{16}$$

where index j denotes the 'outer loop' of decreasing $\mu$, k denotes the index of the 'inner loop' of Newton's method, $\mathbf{W}_k$ is the Hessian of the Lagrangian function, $\mathbf{A}_k = \nabla \mathbf{c}(\mathbf{z})$ and $\mathbf{d}_k^z, \mathbf{d}_k^\lambda, \mathbf{d}_k^y$ are the search directions. The next step is to update the current values of the decision variables, this is done using:

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k \mathbf{d}_k^z \tag{17a}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha_k \mathbf{d}_k^\lambda \tag{17b}$$

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \alpha_k^y \mathbf{d}_k^y. \tag{17c}$$

Here $\alpha$ is the stepsize, more detailed information about how this stepsize is obtained, is described in sec: C.1

## C.1 Newton's Method

Newton's method is an iterative method that is used to find the solution of a set of nonlinear algebraic equations [14]. Consider a set of continuous differentiable nonlinear equations:

$$\mathbf{F}(\mathbf{x}) = 0. \tag{18}$$

Newton's method is interested in finding a point $\mathbf{x}^*$ such that $\mathbf{F}(\mathbf{x}^*) = 0$. Let $\mathbf{x}_0$ be initial guess for $\mathbf{x}^*$. This value can be substituted in Eqn. (18) to see the result of the estimation. The outcome is the error:

$$\boldsymbol{\epsilon}_e = \left|\mathbf{F}(\mathbf{x}_0)\right|. \tag{19}$$

If $\boldsymbol{\epsilon}_e$ is smaller or equal to a user-defined tolerance, the solution is found. If not, a more accurate estimate should be found. To obtain a better estimate a first-order Taylor series around the current estimate is used

$$\mathbf{F}(\mathbf{x}_{i+1})|_{\mathbf{x}_{i+1} \approx \mathbf{x}_i} = \mathbf{F}(\mathbf{x}_i) + \mathbf{J}(\mathbf{x}_i)(\mathbf{x}_{i+1} - \mathbf{x}_i) = 0, \tag{20}$$

where $\mathbf{J}(\mathbf{x})$ is the Jacobian of $\mathbf{F}(\mathbf{x})$. Since the approximation is a linear function, the new estimate can be determined analytically:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{J}(\mathbf{x}_i)^{-1}\mathbf{F}(\mathbf{x}_i). \tag{21}$$

For the new estimate, the error is checked. If it is within the tolerance, the estimate is accepted as the solution. When the tolerance is not met, the difference with the previous estimate is checked:

$$\boldsymbol{\epsilon}_s = |\mathbf{x}_{i+1} - \mathbf{x}_i| \,. \tag{22}$$

When the difference is smaller than the user-defined tolerance, the algorithm is stopped. This is done to prevent the algorithm from doing many iterations without improving the solution significantly. Otherwise, the algorithm continues with the next iteration. Newton's method only guarantees local convergence. When the initial guess is far away from the solution, it is not guaranteed that the Hessian of $\mathbf{F}(\mathbf{x}_i)$ is positive definite. This problem is solved using a line search. This type of Newton's method is called the damped Newton's method [14]. By taking a second-order Taylor expansion, the descent direction is given by:

$$\mathbf{d}_k = -\nabla^2\mathbf{F}(\mathbf{x}_k)^{-1}\nabla\mathbf{F}(\mathbf{x}_k). \tag{23}$$

The next iterate is then found by:

$$\mathbf{x}_{i+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k, \tag{24}$$

where $\alpha_k$ is the stepsize obtained by the line search. This is done using:

$$\alpha_k \min \quad \mathbf{F}(\mathbf{x}_k + \alpha_k\mathbf{d}_k), \tag{25}$$

with $\alpha_k$. The same stopping criteria are applied for the damped Newton's method as for the 'normal' Newton's method.