Model-based motion optimization for quadruped robots with an actuated articulated torso.

M.M.W. Kockelkoren

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Model-based motion optimization for quadruped robots with an actuated articulated torso.

Master of Science Thesis

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

M.M.W. Kockelkoren

Student number: 5425727
Project duration: March 1, 2023 - January 17, 2024

Thesis committee:	Dr.ir. S. (Shengling) Shi	TU Delft, supervisor
	Dr. M. (Manuel) Mazo Espinosa	TU Delft, supervisor
	Dr. C. (Cosimo) Della Santina	TU Delft
	L. (Luyao) Zhang	TU Delft

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

Cover:	Spot® - The Agile Mobile Robot by Boston Dynamics CC BY-SA 3.0 NL (Modified)
Style:	TU Delft MSc Report Style

Abstract

In recent years, the deployment of ground-based mobile robots has gained more and more interest in various domains. In contrast to other types of mobile robots, legged robots can traverse irregular terrains, climb stairs, and step over obstacles. However, these unique properties intensify the energy demand and require highly advanced perception methods, actuator designs, and motion control algorithms. The most significant challenges in legged robotics lie in robustness, energy efficiency, and agility.

In recent years, the integration of an articulated torso or active spine, inspired by the body motion of high-performance mammals like the cheetah, has shown promising results. Various studies observed higher maximum velocities and lower energy consumption compared to a rigid torso. However, in these studies, the compliant systems were typically controlled using basic control strategies. In recent years, the development of highly dynamic model-based motion optimization strategies has greatly enhanced the overall performance of various legged robots. Therefore, a model-based motion optimization scheme is developed specifically for articulated quadruped robots. This scheme fully exploits the additional degrees of freedom of the torso to enhance the dynamic performance of the legged robot further.

Table of Contents

List of Figures

"Motion is created by the destruction of balance; that is, by the destruction of equality of weight, for nothing can move by itself which does not leave its state of balance, and that thing moves most rapidly which is furthest removed from equilibrium."

— *Leonardo da Vinci*

Chapter 1

Introduction

Over the past several decades, mobile robotics has experienced a significant surge in various sectors, taking on multiple tasks. Initially, these robots were designed for simple activities within controlled environments. However, modern mobile robots can autonomously execute complex tasks in unstructured and challenging terrains. Their ability to navigate different spaces significantly extends their operational area and, thus, the variety of tasks they can achieve. Commonly, mobile robots utilize wheels, tracks, or legs for movement on the ground or propellers for aerial movement. Wheels and tracks offer efficient and rapid movement on flat terrains but struggle on uneven, complex surfaces. On the other hand, legged robots have the unique ability to adjust their contact configurations to suit their surroundings, enabling them to navigate obstacles, climb stairs, and traverse rugged terrains. This versatility renders them suitable for diverse applications such as inspection, transportation, object manipulation, and search and rescue missions, as illustrated in Figure 1-1.

Despite their advantages in maneuvering over complex terrains, legged robots face several limitations compared to wheeled robots. First, legged robots generally consume more energy than wheeled robots. Lifting and moving legs requires more power, especially over flat surfaces where wheels are inherently more efficient. Secondly, wheeled robots typically achieve higher speeds due to their wheels' continuous contact and rolling motion. Legged robots, on the other hand, often move slower because of the more complex and controlled movements required for each step. Thirdly, legged robots have a more complex mechanical structure. The multiple joints and actuators in their legs make them more prone to mechanical failures and require more maintenance compared to the more straightforward design of wheeled robots. Finally, while legged robots can traverse highly irregular terrains, maintaining stability is challenging and requires complex control systems to avoid losing balance.

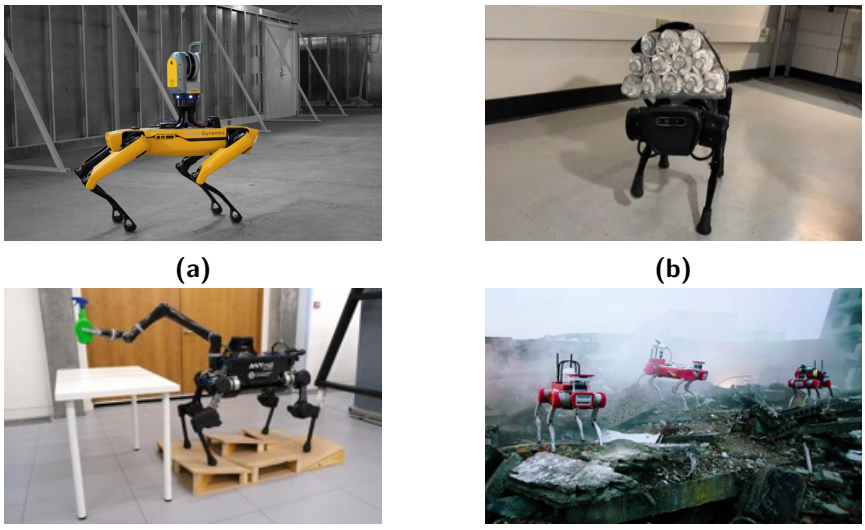


Figure 1-1: Examples of real-world application of multi-legged robots: (a) inspection robot [1], (b) transportation robot [2], (c) manipulation robot [3], (d) search and rescue robot [4].

1-1 Problem statement

In recent years, due to their unique abilities, much research has been dedicated to enhancing the overall performance of quadruped robots. Several studies [5][6][7][8] have investigated the beneficial effects of integrating a spinal joint in the design of the torso and observed faster-running motions, lower energy consumption, and better stability. Despite these convincing benefits, recent advancements in legged robotics have typically disregarded the recommendations discussed in these studies.

One central research area in legged robotics is the development of efficient and stable motion planning and control algorithms. Despite significant advancements in robotics, current motion planning algorithms for quadruped robots often fall short in optimally leveraging articulated mechanisms, particularly in complex and dynamic environments. Traditional motion planning techniques typically focus on stability and basic locomotion, neglecting the potential mobility and energy efficiency enhancements an articulated design can offer. This limitation becomes especially evident in scenarios involving uneven terrain, obstacle navigation, and tasks requiring high agility and precision.

Current dynamic models used in motion planning either suffer from the high dimensionality standard for multi-legged robots or struggle to integrate the capabilities of an articulated spine or torso fully. This typically leads to suboptimal performance in terms of adaptability, speed, and energy consumption. Therefore, the development of articulated-legged robots can only catch up with ordinary robots when the advanced motion planning frameworks are also extended to articulated robots.

1-2 Research objectives

The global objective of this thesis is to extend the operational capabilities of quadruped robots in general, making them more viable for a wider range of applications and setting a new standard for robotic mobility and efficiency. Therefore, the challenge addressed in this thesis is the development of an advanced motion planning framework designed explicitly for quadruped robots with articulated spines. This framework aims to enhance the robot's adaptability to complex terrains, improve energy efficiency, and ensure robust performance in dynamic environments.

The most important aspect of motion optimization for any system is the representation of the dynamics. The level of complexity of the selected representation typically defines the performance of the planning algorithm in terms of accuracy with respect to the actual system and the computational cost required to find a solution. Therefore, one primary objective of this research is the derivation of a low-dimensional but accurate dynamic representation that can leverage the articulation of the torso to improve its performance.

The motion planning frameworks proposed for ordinary quadruped robots are versatile and can typically achieve high performance. Therefore, instead of proposing a new motion optimization framework, this study proposes a convenient method to adapt existing frameworks to leverage the articulation of the torso. This allows articulated robots to level the playing field with legged robots and take advantage of innovations in model-based motion planning for ordinary-legged robots.

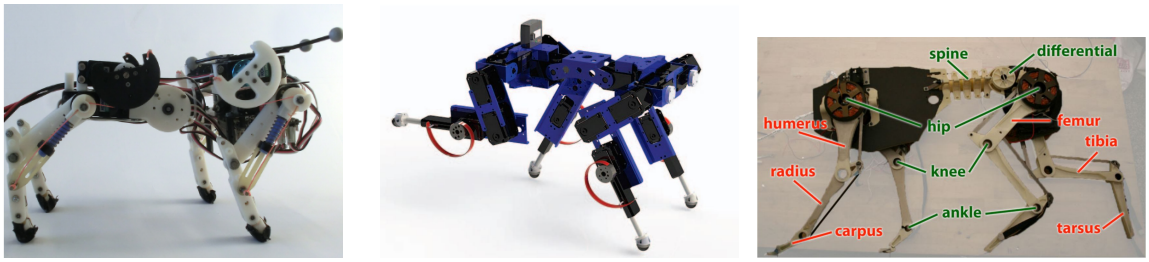
Finally, this framework will be tested by adapting an established motion optimization algorithm to our framework. The potential benefits and flaws of the proposed framework can be identified by analyzing the optimized trajectories and comparing these to the original framework. Ultimately, this study aims to validate the feasibility of the dynamic representation and the optimized trajectory.

1-3 Related work

Articulated robots designs

The articulated spine is an active topic in legged robotics, and much attention is given to improving the design to benefit performance. This has led to many different proposed design frameworks. Foremost, [6] investigated the optimal spine morphology for a single spinal joint (rigid, revolute, prismatic). By performing a large-scale Monte-Carlo optimization, the revolute and especially the prismatic spinal joint improved the ability to accelerate in most experiments compared to the rigid torso. Additionally, more Degrees of Freedom (DOFs) can be added as demonstrated in the designs in Figure 1-2.

Typically, the integration of an actuated torso can increase the power consumption. Therefore, in many studies, parallel compliance is added to the legs [9][10], the spinal joint [11], or both[7]. A much lower energy consumption can be achieved by tuning the stiffness of the compliant elements towards a specific task. Notably, the potential energy in the compliant elements can also have detrimental effects on the performance of different behaviors [12].



(a) (b) (c)

Figure 1-2: Flexible spine designs: (a) 1DOF Bobcat [7] b) 2DOF Dynabot [9], (c) MIT Cheetah [10].

Motion planning algorithms

Several motion planning algorithms for ordinary quadruped robots have been proposed in recent years that have improved the performance of multi-legged robots significantly.

One of the main inspirations of this study is the model-based Trajectory Optimization (TO) approach proposed in [13]. This TO algorithm is specifically designed for legged robots and finds a 6D base trajectory, foothold positions, an appropriate gait schedule, swing-leg motions, and contact forces. By employing phase-based parameterization of the feet' motion and contact forces, the discrete nature of the contact-swing phases was eliminated. It resulted in an Nonlinear Programming (NLP) problem using only continuous decision variables. To improve the computational complexity of the NLP problem, a Single Rigid Body Dynamics (SRBD) model was employed to enforce the system dynamics on the optimization problem. This proposed method is implemented in an open-source C++ library called Trajectory Optimizer for Walking Robots (TOWR), which can generate highly dynamic motion plans for various types of legged robots on different types of terrains.

In the literature, a large set of motion optimization schemes that aim to analyze the optimized behavior of the spine can be found. Most of these studies were performed using a planar robot model and studied the behavior with the bound gait [5]. Some studies integrated parallel compliance in the legs [10], the spinal joint, or both. In [6], a large-scale Monte-Carlo optimization scheme was used to find an optimal spine morphology (rigid, revolute, prismatic) in terms of acceleration. With the two-dimensional Rigid Body Dynamics (RBD) model, it was demonstrated that in the majority of the experiments, the prismatic spine was optimal in terms of acceleration/deceleration and contact force magnitude. In [14], deep Reinforcement Learning was used to learn active spine behaviors for two parallel spinal joints. In all these studies, it was observed that the forward velocity increased, the overall energy consumption decreased, the stride length increased and the bound frequency decreased compared to the same robot with a rigid spine.

Modeling frameworks

In the literature, there are many different studies that present a dynamic representation of a multi-legged robot. These derivations vary significantly in terms of model complexity and accuracy. Surprisingly, the number of studies that derive a dynamic representation specifically

for quadruped robots with an articulated torso is limited. In [15], a Rigid Body Dynamics (RBD) model was derived using the Euler-Lagrange method for a quadruped with two revolute perpendicular torso joints to simulate the dynamic behavior of the robot. In [11], a planar model RBD model was derived for the Bobcat robot in Figure 1-2a with a parallel compliant revolute joint and each leg modeled as a compliant prismatic link.

Finding an appropriate model for trajectory optimization is a tedious task in general and requires thoughtful considerations of the robot design, the robot task, and the computational hardware. While the RBD model is considered accurate when all internal links can be regarded as rigid, the model is considered highly nonlinear and is generally unsuitable for optimization purposes. An analysis of the Centroidal point's dynamics defines a lower-dimensional representation of the Center of Mass (CoM) dynamics. However, this Centroidal Dynamics (CD) model is still highly nonlinear due to the large number of DOFs. A well-known abstraction is the SRBD model to vastly reduce the model complexity. This model generally neglects the leg-joint positions and velocities to compute the Centroidal dynamics. Finally, the dynamic representation can be further abstracted to reduce the computational complexity of the model at the expense of model accuracy.

In [16], the centroidal momentum and dynamics are derived by evaluating the centroidal properties of the torso and an abstraction of each individual leg. This leads to a decrease in the number of dimensions needed to describe the system's dynamics.

1-4 Contributions

The main contribution presented in this report is the derivation of a TO algorithm that can generate an efficient motion plan for a legged robot with one or more articulated spinal joints. The main contribution can be summarized as follows:

Modeling framework for articulated quadrupeds

- Derived an analytic description of the Centroidal Dynamics (CD) model for a quadruped robot with an articulated torso.
- Proposed a low-dimensional and dynamically accurate CD-based model abstraction for quadruped robots with an articulated torso that is convenient to describe in motion optimization algorithms.
- Derived a direct symbolic derivation of the CD/SRBD dynamic representation for quadruped robots with actuated torso joints in MATLAB.
- Derived the inverse kinematic relations of the quadruped with an articulated torso.

Trajectory optimization for articulated quadrupeds

- Derivation of the Triple Rigid Body Dynamics (TRBD) model for a quadruped robot with one or two spinal joints.

- Integration of the TRBD model into the dynamic constraint of the TOWR library.
- Derivation of an analytic expression of the dynamic constraints Jacobians of the TRBD model to increase computational efficiency.
- Derivation of a new kinematic constraint considering the torso joint position.
- Construction of an articulated simulation model to visualize the generated trajectories in Rviz or Gazebo.

Validation of trajectory optimization

- Validated the accuracy of the analytically derived TRBD model against a numeric derivation.
- Discovered a computation error in the CD derivation of the Pinocchio library. The error was reported to the developers and has been corrected.
- Compared the accuracy of the low-dimensional TRBD model to the high-dimensional CD model.
- Analyzed the dynamic consistency of the generated trajectory computed for the articulated torso.

1-5 Outline

This research is structured in the following way. In Chapter 2, a basic theoretical foundation is provided to understand the basic physics behind legged robots in general. Additionally, the concept of the articulated torso is briefly discussed, followed by a general introduction to motion planning. Finally, a short overview of the nomenclature used in this study is provided. Chapter 3 discusses an appropriate modeling framework by first examining established dynamic representations used to describe legged systems. Ultimately, a new dynamic representation is derived and compared to these ordinary representations in the articulated spine and motion optimization context. In Chapter 4, an adaptation to the Trajectory Optimizer for Walking Robots (TOWR) algorithm is proposed that takes full advantage of the articulation of the torso. This is followed by a discussion of the generated trajectories in Chapter 5. Finally, we will briefly discuss the most significant observations of this research as well as the limitations in Chapter 7. Ultimately, we will discuss the main conclusions and recommendations in Chapter 8.

Chapter 2

Theoretical Foundation

Despite the numerous benefits and potential uses of multi-legged robots, their widespread adoption in various industries still lags behind that of wheeled robots. As discussed, legged robots face many challenges in aspects like energy consumption, limited velocities, a demanding mechanical structure, and a complex control architecture. In the literature review performed before this thesis, three primary challenges hindering the broad application of legged robots were identified: energy efficiency, stability, and performance. Numerous studies discussed various proposals in research areas like mechanical design, actuation, or control, that proved beneficial. Two independent proposals seemed to have a positive impact on all three challenges: the integration of spinal joints and efficient motion planning algorithms. To this day, the development of efficient motion planning algorithms is still a very active research area within the subject of multi-legged robots. Similarly, the design and potential performance of multi-legged robots with one or more spinal joints is an active independent research area as well. However, to this day, no efficient trajectory optimization scheme has been proposed that comes close to the optimization schemes used for legged robots with a rigid torso.

2-1 Basic physics behind legged-robots

Robots, in general, can be categorized into two groups, namely fixed-base robots and floating-base robots. Fixed-base robots are typically rigidly fixed to the environment, while floating-base robots can move freely through space. Evidently, wheeled and legged robots are considered floating-base robots. The floating-base systems generally cannot actuate the position and orientation of the base directly through an actuator but only indirectly by external forces generated by their environment. These external forces can be aerodynamic forces in aerial robots or contact forces in land-based robots. These typically complex interactions with the environment usually introduce uncertainties and necessitate a good understanding of the surroundings. Therefore, sophisticated sensory equipment is crucial for the good operation of these robots.

Legged robots, in general, are incredibly complex systems due to their large number of dimensions, interaction with the environment, different styles of locomotion, and intricate kinematic and dynamic relationships. A legged robot typically consists of a floating base and a number of legs. Each leg is equipped with a number of joints, typically actuated with an electric motor that exerts a torque on the joint. By changing the configuration of its legs, a legged system is able to alter the locations where the external forces act on the system. Furthermore, in legged robots, the external forces used for locomotion are the reaction forces generated by exerting a force on the ground surface. This set of forces, in combination with their contact positions, generates an indirect net force and moment on the floating base. This allows the floating base to move and rotate in a specific direction.

2-1-1 Gait schedule

Another critical aspect of multi-legged robots is the gait schedule, which is visualized in Figure 2-1. This timing schedule is the time and duration at which a foot is in contact with the surface. Figure 2-1 shows that there are various possible combinations for four-legged robots. The gait schedule is a highly important aspect of locomotion in legged systems, as each gait is optimal for different behaviors in terms of efficiency, velocity, and stability. For example, the walking gait offers more stability and is also more efficient at low velocities; however, at higher velocities, the bound gait becomes more efficient.

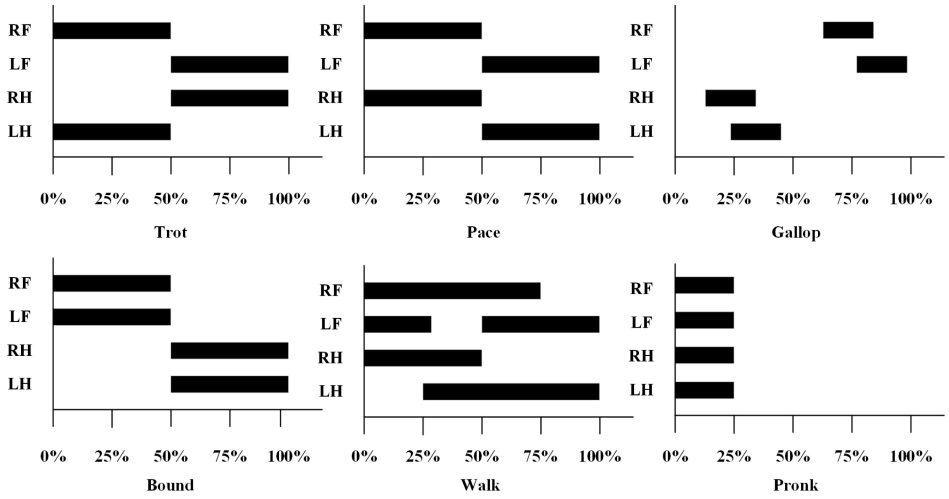


Figure 2-1: Various common gait schedules for quadruped robots, with the contact-phase (black) and flight-phase (white) [17].

2-1-2 Stability and balance

One key element of the physical properties of the legged robot is stability. This stability is also one of its main concerns. Therefore, we will briefly discuss two basic concepts regarding stability. When considering a four-legged robot, the table-like structure typically provides static stability when the Center of Mass (CoM) projected on the ground surface lies inside the contact area spanned by the feet. However, when a leg is lifted off the ground, the contact

area reduces to a trapezoid, causing the CoM to fall outside this region. The quadruped robot will then tip over and ultimately fall to the ground. This principle is often used for low velocities, where the robot's CoM is kept within the contact region spanned by the feet.

For larger velocities, the system may still remain stable while the CoM is outside the region of static stability. This type of stability is called dynamic stability and refers to the ability to maintain balance while in motion. Unlike static stability, where an object remains stable when stationary, dynamic stability is about the ability to maintain balance during movement. While the momentum of the system during locomotion has a positive impact on balance as well, a well-planned set of contact forces during ground contact can actuate the base to keep balance.

2-2 Kinematic relations

When all the links of a system are considered to be fully rigid, a kinematic model of a robot can be constructed that takes advantage of the geometry of the robot to find the kinematic chain of the system. This kinematic relation between joints can be used to obtain the position and orientation of a certain frame with respect to another. This kinematic relation is also known as Forward Kinematics (FK). The current states of all Degrees of Freedom (DOFs) are captured in the generalized state vector $\mathbf{q} \in \mathbb{R}^n$. By using the kinematic relation between frames in combination with the current state vector, the position of each frame can be determined with respect to another. For example, as visualized in Figure 2-2a, the end-effector position $\mathbf{r} + ee$ can be computed directly by entering the full state vector $\mathbf{q}$ in the FK relation.

Figure 2-2: Schematic representation of the application for a) Forward Kinematics (FK), b) Inverse Kinematics (IK).

Alternatively, the geometric relation between joints can also be exploited to convert a desired frame position $\mathbf{r}$ into the desired state vector $\mathbf{q}$. The state vector then computes the required state vector that positions a certain frame in the desired position. This conversion is known as IK. A schematic representation can be observed in Figure 2-2b, where the full state vector is obtained that places the end-effector in a particular position $\mathbf{r}_{ee}$. In the following subsections, both kinematic relations will be discussed.

2-3 Dynamic representations

For model-based motion optimization algorithms, an appropriate mathematical representation of the system dynamics is crucial for the algorithm's performance. This dynamic constraint enforces that the generated trajectory is dynamically feasible. In the literature, there are many different dynamic representations available for ordinary quadruped robots, each with varying

complexity and accuracy. Three models discussed in [13] are visualized in Figure 2-3. The Rigid Body Dynamics (RBD) model assumes all links to be fully rigid and provides a dynamic representation for all states of the system. The Centroidal Dynamics (CD) model assumes that all joints are fully actuated, such that the joints can follow any generated trajectory. This assumption eliminates the need for a representation of the joint dynamics. The Single Rigid Body Dynamics (SRBD) model fully neglects the joint dynamics and observes the whole system as a single floating rigid body with contact points and forces. Finally, the Linear Inverted Pendulum (LIP) model is a linearization of the system dynamics and neglects the angular dynamics of the system. It models the robot as a virtual leg connecting the CoM and the Center of Pressure (CoP). With the exemption of the latter model, all models are discussed further in Chapter 3. Selecting a simplified model can misrepresent the system dynamics and, therefore an infeasible generated trajectory. However, full-body models are typically complex and potentially increase the computational effort.

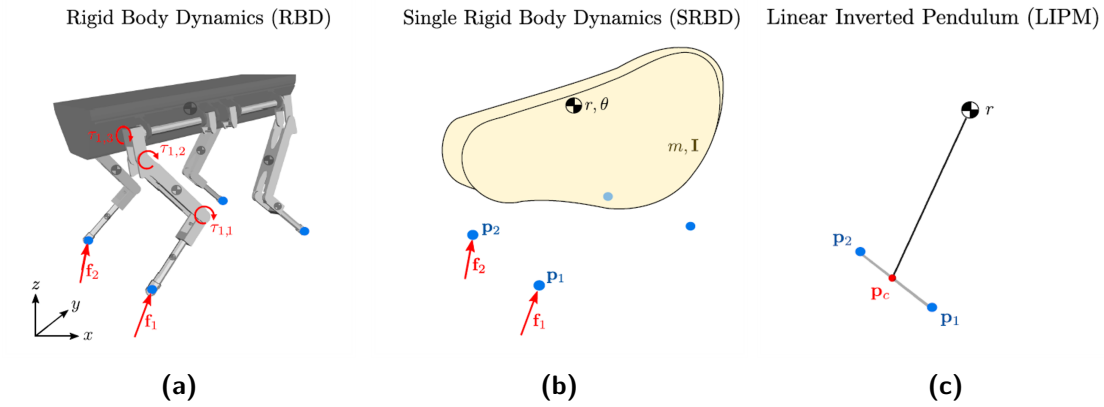


Figure 2-3: Visual representation of dynamic models, from [13]: (a) RBD/CD model, (b) SRBD model, (c) LIP model.

2-4 Articulated torso

The principle of the articulated torso originated from the study of the bodily motion of fast land animals like the cheetah. These animals benefit significantly from the flexibility of the torso in terms of energy consumption, stability, and maximum velocity. As can be observed in Figure 2-4, the cheetah alternates between a gathered and extended torso during the flight phases. The flexible spine seen in mammals allows a longer distance covered between each step (stride length)[18][8]. Furthermore, the change in mass distribution allows for more self-stabilization by adjusting the angle and angular velocity of the leg [19]. In ordinary quadruped robots, the main function of the torso is to store hardware components needed for locomotion or for task execution. However, these biological observations have led to the proposition to revise the design of the torso and integrate various types of articulation.

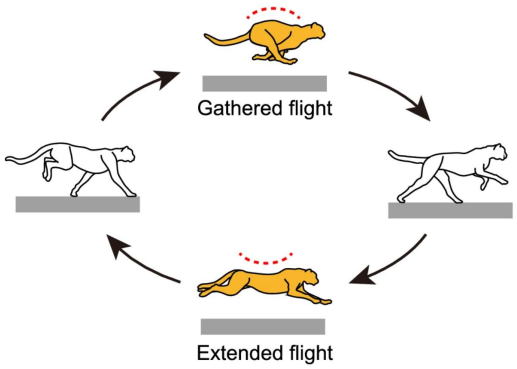


Figure 2-4: Cheetah galloping involves two types of flights through spine movement: gathered and extended, from [20].

2-5 Motion planning and control

Motion planning and control are intricate subjects within the context of legged robotics. The motion-control algorithm of a mobile robot can typically be subdivided into three main groups, shown in Figure 2-5. The first group is the sensory unit, which uses its sensory hardware to estimate the system's internal states and map its environment. The motion planning unit uses this information to generate a finite-time motion plan to travel to a desired point in space or to execute a certain task. The tracking unit is responsible for computing the required control inputs to track this plan and counteract any disturbances. As evident from Figure 2-5, all these elements are highly dependent on each other.

Motion planning is a complex and essential area of study in robotics, focusing on the development of methods and algorithms that enable robots to move effectively in varying environments. Especially in mobile robotics, motion planning is a crucial task that allows the robot to move through its surroundings while avoiding obstacles. Unlike wheeled robots, legged robots must deal with a higher degree of freedom, complex dynamics, and keeping balance, making their motion planning more challenging.

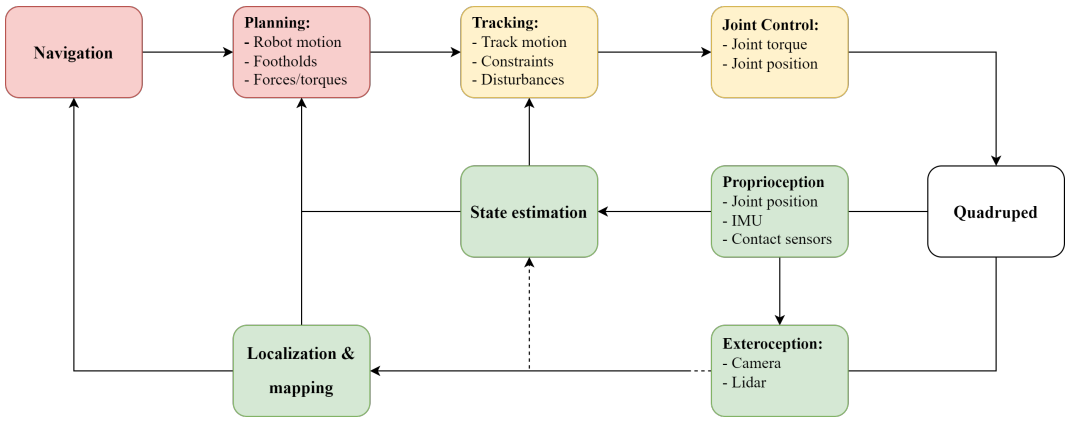


Figure 2-5: Common motion-control architecture for mobile robots, subdivided into three main components: planning (red), control (yellow), and sensing (green).

Typically, motion planning algorithms plan a motion trajectory over a finite time horizon. In general, motion planning algorithms can be divided into four main methods: graph-based methods, sampling-based methods, optimization-based methods, and learning-based methods. The graph-based method involves creating a map of the environment and using algorithms like A* or Dijkstra's to find an optimal path. It's suitable for static environments but less so for dynamic or unknown terrains. Sampling-based methods like Rapidly Exploring Random Trees (RRT) are used to explore unknown or dynamic environments. For optimization-based methods, the motion of the robot is formulated as an optimization problem that finds an optimal solution by considering factors like trajectory feasibility, energy efficiency, and velocity. Furthermore, the optimization problem allows for the consideration of system or task-specific constraints. Finally and more recently, learning-based approaches like Reinforcement Learning (RL) are being explored for mobile robotics. These methods enable robots to learn optimal behavior through trial and error in a simulated or real environment.

For legged robotics, the complexity of the system typically demands advanced planning algorithms like optimization or learning-based approaches. Learning-based approaches are highly dependent on their training data and can, therefore, become unpredictable for situations they did not encounter during the training phase. On the other hand, optimization-based approaches are highly dependent on the derived set of constraints and typically suffer from the large number of dimensions and the nonlinear nature of the system. These often enforce large burdens on the computational hardware. Typically, the computational burden is reduced by using abstractions of the constraints and reducing the number of dimensions as discussed in Section 2-3. This typically affects the feasibility of the generated trajectory.

Furthermore, while the gait schedule is highly important in motion planning, it typically also imposes a significant computational burden on the motion optimization algorithm when including it in the trajectory optimization. Therefore, in typical motion optimization schemes, these two tasks are typically decoupled. Through independent gait optimization algorithms, an appropriate gait schedule is selected prior to the optimization.

2-5-1 Formulation of trajectory optimization

Trajectory optimization algorithms are typically formulated mathematically as Equation (2-1). Here, the continuous trajectory is typically represented by the state trajectory $\mathbf{x}(t)$ and input trajectory $\mathbf{u}(t)$ for a finite time-interval $t \in [t_0, t_f]$.

$$\begin{array}{rl} \min_{\mathbf{u}(\cdot)} & \ell_f\left(\mathbf{x}\left(t_f\right)\right) + \int_{t_0}^{t_f} \ell(\mathbf{x}(t), \mathbf{u}(t)) dt \\ \text{subject to} & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad \forall t \in [t_0, t_f] \\ & \mathbf{x}\left(t_0\right) = \mathbf{x}_0. \end{array} \tag{2-1}$$

Furthermore, the algorithm aims to minimize an objective function $\ell(\mathbf{x}(t), \mathbf{u}(t))$, that describes the desired behavior. Furthermore, the optimization problem contains a set of constraints on the state trajectory or the input trajectory. For trajectory optimization, a critical constraint is the dynamic constraint $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$, that describes the dynamic behavior of the system. Other constraints typically describe a start $\mathbf{x}_0$ and final state $\mathbf{x}_\mathrm{f}$, collision avoidance, and other system-specific constraints.

Direct collocation method

The formulation in Equation (2-1), describes the trajectory optimization formulation in continuous time. In order to formulate this Trajectory Optimization (TO) for a numeric solver, this TO problem needs to be parameterized to a finite number of decision variables. In [21], several methods are described to parameterize these decision variables. Sequential methods like direct transcription and direct shooting discretize the continuous trajectory and constraints to solve for the discretized trajectory. In contrast, simultaneous methods like the direct collocation method represent the trajectories as piece-wise polynomial functions, as shown in Figure 2-6. For a cubic polynomial, provided in Equation (A-15), the continuous trajectory between two break points $t \in [t_{k-1}, t_k]$ can be expressed using four coefficients. These coefficients are fully defined using the expressions for $\mathbf{x}(t_{k-1}), \mathbf{x}(t_k)$ and the expressions for their time-derivatives $\dot{\mathbf{x}}(t_{k-1}), \dot{\mathbf{x}}(t_k)$. Furthermore, the direct collocation solver simultaneously varies the input and output trajectories while enforcing the dynamic constraints at the so-called collocation points.

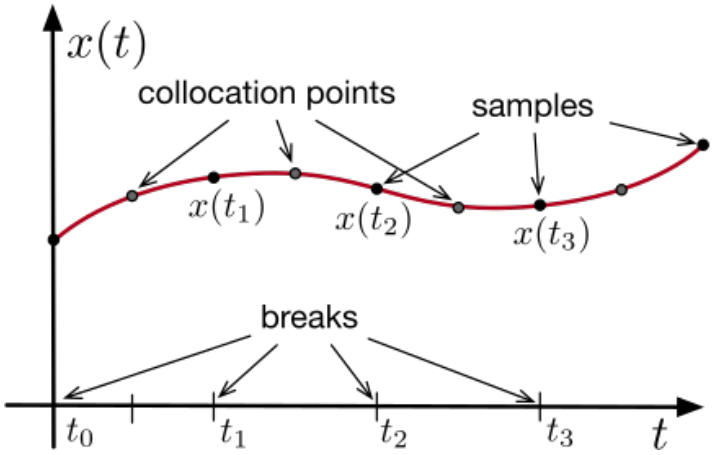


Figure 2-6: Cubic spline parameters used in the direct collocation method, from [21].

Phase-based parameterization

In legged systems, the discontinuous nature of the contact dynamics is caused by the change in dynamics when a foot makes or breaks contact with the ground surface. For this reason, the TO with contact-dependent constraints is a complex task from a computational perspective. The hybrid nature of the contact constraints is a common problem in legged motion planning. In much of the literature, the motion optimization problem is decoupled in a gait and footstep planner that specifies the contact schedule and contact locations prior to the motion optimization. The constraints are then enforced only on these specific time intervals. However, as stated in [13], this approach decreases computational complexity but degenerates the range of achievable motions and agility on difficult terrains.

Another common way, proposed in [13], is the use of phase-based parameterization that differentiates between the swing phase and the stance phase, visualized in Figure 2-7. The contact-dependent decision variables are parameterized using polynomials between the phase

time intervals. The order of the polynomial and the set of constraints is switched between the stance and the swing phase.

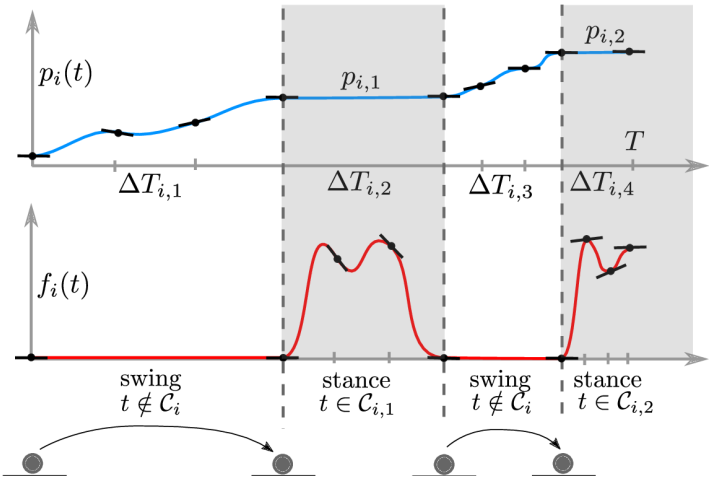


Figure 2-7: Phase-based parameterization of the end-effector position $\mathbf{p}_i(t)$ and contact force $\mathbf{f}_i(t)$, from [13].

2-6 Nomenclature

Before diving into the complicated kinematic and dynamic relations, it is important to describe the notation used for all parameters. In this section, we will elaborate on the notation of the most important parameters used in the following sections. First, it must be noted that matrices are typically denoted by a bold capital letter ($\mathbf{A} \in \mathbb{R}^{n \times m}$). Furthermore, vectors are typically denoted by a bold lower-case letter ($\mathbf{a} \in \mathbb{R}^n$). Finally, scalars are typically represented by lower-case letters ($a \in \mathbb{R}$). Furthermore, when describing a coordinate frame, a capital calligraphic letter is used $\mathcal{A}$. When we discuss the origin of that frame or a point, a non-bold capital letter is used A.

Furthermore, a matrix or vector filled with zeros is typically represented by a bold $\mathbf{0}$. In contrast, an identity matrix, a zero matrix with ones on the diagonal, will be represented by $\mathbf{1}$. Finally, the cross-product between two three-dimensional vectors is typically denoted by $\times$. However, often it is more convenient to denote the cross multiplication using a skew-symmetric matrix $S(.) \in \mathbb{R}^{3\times3}$. For example, when considering the cross product between vector $\mathbf{a} \in \mathbb{R}^3$ and $\mathbf{b} \in \mathbb{R}^3$, it can be denoted using the following two notations:

$$\mathbf{a} \times \mathbf{b} = \mathbf{S}(\mathbf{a})\mathbf{b} \qquad \text{with}: \quad \mathbf{S}(\mathbf{a}) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{2-2}$$

2-6-1 Relevant coordinate frames

Finally, there are a number of coordinate frames that are often used throughout this thesis, so it might be convenient to discuss these first. All these coordinate frames are visualized in

Figure 2-8. First, we denote the fixed world coordinate frame as the Inertial frame $\mathcal{I}$. Second, the center of the middle section of the torso is denoted by the Base frame $\mathcal{B}$. Furthermore, the front and hind sections of the torso are denoted by the Front $\mathcal{F}$ and Hind $\mathcal{H}$ frames, respectively. These frames are placed along the rotation axis of the torso joints and are aligned with the front and hind sections of the torso. Finally, the end-effector frames $\mathcal{EE}$ are connected to the most distal part of each leg and are named based on their relative location to the base frame. For example, the frame belonging to the right-front end-effector is called $\mathcal{EE}_{RF}$.

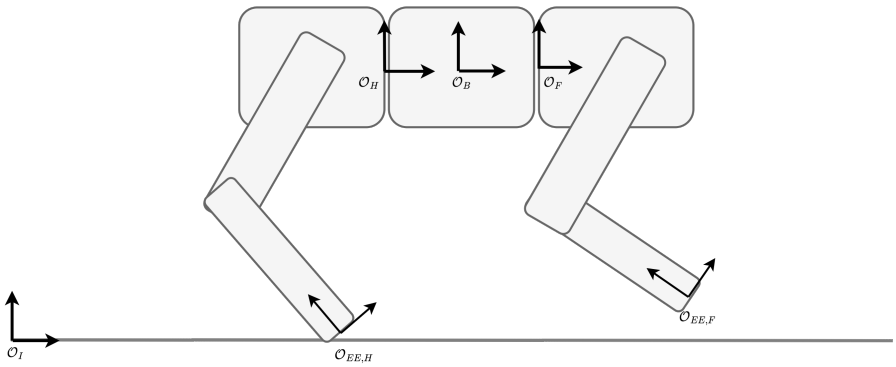


Figure 2-8: Relevant coordinate frames.

2-6-2 Motion parameterization

Position

A position vector is a three-dimensional vector used to describe the translation from one point to another. The position vector always has a reference frame. For example, the position vector from point A to point B, with respect to frame $\mathcal{C}$ is denoted by ${}_{\mathcal{C}}\mathbf{r}_{AB} \in \mathbb{R}^3$. Typically, when the origin on the reference frame is located in the starting point, we denote the position vector without a reference frame (e.g. $\mathbf{r}_{AB} = {}_{\mathcal{A}}\mathbf{r}_{AB}$). Furthermore, if the reference frame $\mathcal{A}$ is the world or inertial frame $\mathcal{I}$ the position vector is typically represented with $\mathbf{r} \in \mathbb{R}^3$ in combination with a subscript of the endpoint (e.g. $\mathbf{r}_A = {}_{\mathcal{I}}\mathbf{r}_{IA}$).

Rotation

Rotation matrices are an important aspect of modeling and allow us to describe the orientation of a certain frame with respect to another. In contrast to [22], we will use $\mathbf{R}_{\mathcal{AB}} \in \mathbb{R}^{3\times3}$ to denote a rotation from frame $\mathcal{B}$ to frame $\mathcal{A}$. Rotation matrices can be made up of multiple individual rotations by multiplying them in the order of rotation.

$$\mathbf{R}_{\mathcal{AC}} = \mathbf{R}_{\mathcal{AB}}\mathbf{R}_{\mathcal{BC}} \tag{2-3}$$

Moreover, another useful property of the rotation matrix is that the inverse or transpose of a rotation matrix represents the inverse rotation. For example, the transpose of the rotation

matrix from frame $\mathcal{B}$ to $\mathcal{A}$ is the rotation matrix from frame $\mathcal{A}$ to frame $\mathcal{B}$.

$$\mathbf{R}_{\mathcal{BA}} = \mathbf{R}_{\mathcal{AB}}^{\intercal} = \mathbf{R}_{\mathcal{AB}}^{-1} \tag{2-4}$$

Finally, rotation matrices can transform any type of three-dimensional motion or force-type vector to another frame. For example, the position vector from point A to B can be rotated to obtain the position vector with respect to frame $\mathcal{C}$.

$${}_{\mathcal{C}}\mathbf{r}_{AB} = \mathbf{R}_{\mathcal{CA}} \cdot {}_{\mathcal{A}}\mathbf{r}_{AB} \tag{2-5}$$

Coordinate transformation

To express a combination of linear and angular transformation, a transformation matrix $\mathbf{T} \in \mathbb{R}^{4\times4}$ is used. For example, the transformation between between frame $\mathcal{A}$ and frame $\mathcal{B}$ consists of a combination of rotations $\mathbf{R}_{\mathcal{AB}}$ and translations ${}_{\mathcal{A}}\mathbf{R}_{\mathcal{AB}}$. The rigid transformation matrix between frame $\mathcal{A}$ and frame $\mathcal{B}$ and the opposite relation are constructed as follows.

$$\mathbf{T}_{\mathrm{AB}} = \begin{bmatrix} \mathbf{R}_{\mathcal{AB}} & {}_{\mathcal{A}}\mathbf{r}_{AB} \\ \mathbf{0} & 1 \end{bmatrix}, \quad \mathbf{T}_{\mathrm{BA}} = \mathbf{T}_{\mathrm{AB}}^{-1} = \begin{bmatrix} \mathbf{R}_{\mathcal{AB}}^{\intercal} & {}_{\mathcal{B}}\mathbf{r}_{BA} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2-6}$$

Furthermore, transformation matrices can be multiplied to obtain the transformation matrix for a set of sequential transformations. This allows for a convenient and structured approach to derive a composite transformation matrix, which describes the transformation.

$$\mathbf{T}_{\mathrm{AC}} = \mathbf{T}_{\mathrm{AB}}\mathbf{T}_{\mathrm{BC}} \tag{2-7}$$

2-6-3 Velocity

Another key concept is the linear and angular velocity of a rigid body. The linear velocity of point B with respect to point A will be represented by the time-derivative of the position vector ${}_{\mathcal{A}}\dot{\mathbf{r}}_{AB} \in \mathbb{R}^3$. The angular velocity of point B with respect to point A is denoted by ${}_{\mathcal{A}}\boldsymbol{\omega}_{AB}$.

In rigid-body dynamics, the absolute velocity is often used to describe the velocity of a moving body and the velocity of all points inside that body. For example, if we have a body with linear velocity $\dot{\mathbf{r}}_B$ with respect to a fixed frame and a point P within that body, we can express the spatial velocity of that point using the spatial linear velocity of B and the position vector between B and A.

$${}_{\mathcal{A}}\dot{\boldsymbol{r}}_{\mathrm{IP}} = {}_{\mathcal{A}}\dot{\mathbf{r}}_{IB} + {}_{\mathcal{A}}\boldsymbol{\omega}_{\mathrm{IB}} \times {}_{\mathcal{A}}\mathbf{r}_{\mathrm{BP}} = {}_{\mathcal{A}}\boldsymbol{v}_{\mathrm{P}} \tag{2-8}$$

The absolute velocity is a useful tool to represent the velocity of any point in a body by the linear and angular velocity of the body and the position vector to the point. This is typically captured in the spatial velocity $\mathbf{v} \in \mathbb{R}^6$, which captures the absolute linear velocity $\boldsymbol{v} \in \mathbb{R}^3$ and absolute angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$.

$${}_{\mathcal{A}}\mathbf{v}_B = \begin{bmatrix} {}_{\mathcal{A}}\boldsymbol{\omega}_{\mathrm{B}} \\ {}_{\mathcal{A}}\boldsymbol{v}_{\mathrm{B}} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathcal{AI}} \cdot \boldsymbol{\omega}_{\mathrm{IB}} \\ \mathbf{R}_{\mathcal{AI}}(\dot{\mathbf{r}}_{\mathrm{IB}} + \boldsymbol{\omega}_{\mathrm{IB}} \times \mathbf{r}_{\mathrm{BP}}) \end{bmatrix} \tag{2-9}$$

Using the spatial velocity instead of the ordinary velocity allows a more convenient transformation of the absolute velocity to reference frames.

Furthermore, the spatial acceleration $\mathbf{a}_{\mathrm{IB}} \in \mathbb{R}^6$ of a rigid body is typically determined as follows:

$$\mathbf{a}_B = \begin{bmatrix} \dot{\boldsymbol{\omega}}_{\mathrm{B}} \\ \boldsymbol{a}_{\mathrm{B}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega}_{\mathrm{IB}} \\ \ddot{\mathbf{r}}_{\mathrm{IB}} - \boldsymbol{\omega}_{\mathrm{IB}} \times \dot{\mathbf{r}}_{\mathrm{IB}} \end{bmatrix}. \tag{2-10}$$

2-6-4 Spatial Jacobian

An important concept in differential kinematics and multi-body dynamics is the Jacobian matrix ${}_{\mathcal{A}}\mathbf{J} \in \mathbb{R}^{6\times n}$, which maps the joint rates $\dot{q}$ to the spatial velocity of a frame with respect to frame $\mathcal{A}$. It must be noted that this Geometric Jacobian has a reference frame and thus maps the joint rates to the spatial velocity with respect to that frame.

$${}_{\mathcal{I}}\mathbf{v}_{\mathrm{Q}} = \begin{bmatrix} \boldsymbol{v}_{IB} \\ \boldsymbol{\omega}_{\mathrm{IB}} \end{bmatrix} = \begin{bmatrix} {}_{\mathcal{I}}\mathbf{J}_{\mathrm{Q,P}}(\mathbf{q}) \\ {}_{\mathcal{I}}\mathbf{J}_{\mathrm{Q,R}}(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}} = {}_{\mathcal{I}}\mathbf{J}_{\mathrm{Q}}(\mathbf{q})\dot{\mathbf{q}} \tag{2-11}$$

The linear spatial velocity of an arbitrary point Q with respect to the inertial frame can be derived by taking the time-derivative of the expression for $\mathbf{r}_{IQ}$. This results in an expression for the absolute linear velocity with respect to the inertial frame as a function of the generalized velocity.

$$\begin{aligned} \dot{\mathbf{r}}_{IQ} &= \dot{\mathbf{r}}_{IB} + \dot{\mathbf{R}}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\mathbf{r}_{BQ} + \mathbf{R}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\dot{\mathbf{r}}_{BQ} && \text{(2-12a)} \\ &= \dot{\mathbf{r}}_{IB} - \mathbf{R}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\mathbf{r}_{BQ} \times \boldsymbol{\omega}_{\mathrm{IB}} + \mathbf{R}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\mathbf{J}_v(\mathbf{q}_j) \cdot \dot{\mathbf{q}}_j, && \text{(2-12b)} \\ &= \underbrace{\begin{bmatrix} \mathbb{I}_{3\times 3} & \mathbf{S}(\mathbf{R}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\mathbf{r}_{BQ}) & \mathbf{R}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\mathbf{J}_v(\mathbf{q}_j)\cdot \end{bmatrix}}_{{}_{\mathcal{I}}\mathbf{J}_P} \cdot \dot{\mathbf{q}}, && \text{(2-12c)} \end{aligned}$$

where the analytical Jacobian is provided by ${}_{\mathcal{B}}\mathbf{J}_v = \frac{\partial \mathbf{r}_{BQ}}{\partial \mathbf{q}_j}$. For the angular spatial velocity of an arbitrary point on the floating-base system, a similar expression for the spatial rotational Jacobian ${}_{\mathcal{I}}\mathbf{J}_R$ can be derived.

$$\begin{aligned} {}_{\mathcal{I}}\boldsymbol{\omega}_{\mathrm{IQ}} &= {}_{\mathcal{I}}\boldsymbol{\omega}_{\mathrm{IB}} + {}_{\mathcal{I}}\boldsymbol{\omega}_{\mathrm{BQ}} && \text{(2-13a)} \\ &= \underbrace{\begin{bmatrix} \mathbf{0}_{3\times 3} & \mathbb{I}_{3\times 3} & \mathbf{R}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\mathbf{J}_w(\mathbf{q}_j) \end{bmatrix}}_{{}_{\mathcal{I}}\mathbf{J}_R} \cdot \dot{\mathbf{q}}, && \text{(2-13b)} \end{aligned}$$

where ${}_{\mathcal{B}}\mathbf{J}_w$ denotes the rotational Jacobian matrix of point Q with respect to the base frame. Evidently, the derivation of the spatial Jacobian matrices is highly dependent on the selected parameterization of $\dot{\mathbf{q}}$.

2-6-5 Spatial transformations

To transform a spatial vector to another reference frame, a spatial transformation matrix ${}^{\mathrm{B}}\mathbf{X}_{\mathrm{A}} \in \mathbb{R}^{6\times 6}$ can be used. This transformation matrix projects a spatial vector represented in frame $\mathcal{A}$ to another frame $\mathcal{B}$. It must be noted that force-type vectors are transformed using

a force-type spatial transformation matrix ${}^{\mathrm{B}}\mathbf{X}_{\mathrm{A}}$. The motion-type and force-type transformation matrices are related in the following way:

$$ {}^{\mathrm{B}}\mathbf{X}_{\mathrm{A}}^{*} = {}^{\mathrm{B}}\mathbf{X}_{\mathrm{A}}^{-\intercal} = {}^{\mathrm{A}}\mathbf{X}_{\mathrm{B}}^{\intercal} \tag{2-14} $$

The spatial transformation matrices from frame A to frame B can be constructed by using the translation $\mathbf{r}_{\mathrm{AB}}$ and the rotation $\mathbf{R}_{\mathcal{BA}}$ matrix from frame $\mathcal{A}$ to frame $\mathcal{B}$ with the relation in Equation (2-15).

$$ {}^{B}\mathbf{X}_{\mathcal{A}} = \begin{bmatrix} \mathbf{R}_{\mathcal{BA}} & \mathbf{0} \\ \mathbf{S}(\mathbf{r}_{\mathrm{AB}})\mathbf{R}_{\mathcal{BA}} & \mathbf{R}_{\mathcal{BA}} \end{bmatrix}, \quad {}^{B}\mathbf{X}_{A}^{*} = \begin{bmatrix} \mathbf{R}_{\mathcal{BA}} & \mathbf{R}_{\mathcal{BA}}\mathbf{S}(\mathbf{r}_{\mathrm{AB}}) \\ \mathbf{0} & \mathbf{R}_{\mathcal{BA}} \end{bmatrix} \tag{2-15} $$

Chapter 3

Multi-legged Robot: Modeling Framework

3-1 Introduction

In the context of model-based trajectory optimization, the representation of the dynamic behavior of the robot is a fundamental aspect that has great impact on the feasibility and update frequency of the generated trajectory. Therefore, the careful selection of an appropriate dynamic model is a crucial step in trajectory generation. It is, therefore, important to understand the dynamic behavior of the system itself and also how each model is capable of representing that behavior. In general, there are numerous dynamic representations for legged robots with varying accuracies, complexities, and applications. Selecting a dynamic model with a high level of accuracy generally improves the feasibility of the trajectory. However, typically, it has a detrimental effect on the computation speed. Selecting a dynamic model that neglects certain aspects of the dynamic behavior might improve the computation speed but might also result in infeasible trajectories.

This chapter will start with a short introduction to the parameterization of generalized states of the system. This is followed by a short derivation of the kinematic relations between the frames of the quadruped robot with the articulated torso. This is followed by an elaboration on the most conventional dynamic representations: Rigid Body Dynamics (RBD), Centroidal Dynamics (CD), Single Rigid Body Dynamics (SRBD). Furthermore, we will discuss how these dynamic representations fit in the context of trajectory generation for quadruped robots with an articulated torso. In this context, we conclude that the conventional models are not sufficient, and therefore, a new dynamic representation is presented.

3-2 Parameterization of generalized coordinates

Before diving into the kinematic and dynamic relations, it is important to understand the type of system, the Degrees of Freedom (DOFs), and how these are typically parameterized.

First of all, as discussed before, a legged robot is not only a mobile robot but also belongs to a floating-base type of system. This implies that the state vector of the robot contains at least six DOFs for the position and orientation of the floating base.

Position and rotation

These six DOFs generally belong to the Euclidean space $\mathbf{q}_\mathrm{b} \in SE(3) = \mathbb{R}^3 \times SO(3)$, which describes both the position and orientation. The position coordinates of the floating-base are typically parameterized in Cartesian space such that $\mathbf{q}_{\mathrm{b_P}} \in \mathbb{R}^3$. The rotation of the base frame is represented in the rotation group $\mathbf{q}_{\mathrm{b_R}} \in SO(3)$. The rotation can be parameterized in numerous ways using Euler-angles, Angle-Axis, and Unit Quaternions, as described in [22]. The Euler angles only require three independent variables to represent a rotation in space, while the latter two require four independent variables. The generalized variable for the orientation of the torso is denoted by $\boldsymbol{\theta}_\mathrm{IB} \in \mathbb{R}^3$ when using Euler-angles.

Furthermore, while an ordinary quadruped robot with a single torso section and three leg joints per leg introduces $n_\mathrm{lj} = 12$ DOFs, the torso-joints add another n_tj DOFs to the state vector. The states belonging to all internal joints in the robot can be combined into a single state vector $\mathbf{q}_j \in \mathbb{R}^{n_\mathrm{lj}+n_\mathrm{tj}}$. The internal joints can be categorized into n_tj torso joints and n_lj leg joints.

$$\mathbf{q} = \begin{pmatrix} \mathbf{q}_\mathrm{b} \\ \mathbf{q}_\mathrm{j} \end{pmatrix} \in SE(3) \times \mathbb{R}^{n_j} \tag{3-1}$$

$$\mathbf{q}_\mathrm{b} = \begin{pmatrix} \mathbf{r}_\mathrm{IB} \\ \boldsymbol{\theta}_\mathrm{IB} \end{pmatrix} \in \mathbb{R}^3 \times SO(3), \quad \mathbf{q}_j = \begin{pmatrix} \boldsymbol{\varphi}_\mathrm{tj} \\ \boldsymbol{\varphi}_\mathrm{lj} \end{pmatrix} \in \mathbb{R}^{n_\mathrm{lj}+n_\mathrm{tj}} \tag{3-2}$$

The angular position vector of the torso joints is given by $\boldsymbol{\varphi}_\mathrm{tj} = [\varphi_\mathrm{f}, \varphi_\mathrm{h}]^\top$, while the angular position vector of a single leg joint is given by $\boldsymbol{\varphi}_\mathrm{lj,side} = [\varphi_\mathrm{haa,side}, \varphi_\mathrm{hfe,side}, \varphi_\mathrm{kfe,side}]^\intercal$. This order corresponds to the Hip Abduction-Adduction (HAA), HIP Flexion-Extension (HFE), Knee Flexion-Extension (KFE) joint, typically used in legged robotics. A specific leg is typically indicated in the subscript with two letters after the comma, with the corresponding side like: Left-Front (LF), Right-Front (RF), Left-Hind (LH), Right-Hind (RH).

Velocity and acceleration

To describe the velocity of each frame of the robot, we need to parameterize the velocity of the floating-base frame with respect to the Inertial frame.

$$\dot{\mathbf{q}} = \begin{pmatrix} {}_\mathcal{B}\boldsymbol{v}_\mathrm{IB} \\ {}_\mathcal{B}\boldsymbol{\omega}_\mathrm{IB} \\ \dot{\boldsymbol{\varphi}}_\mathrm{j} \end{pmatrix} \in \mathbb{R}^{6+n_j} = \mathbb{R}^{n_u} \quad \ddot{\mathbf{q}} = \begin{pmatrix} {}_\mathcal{B}\boldsymbol{a}_\mathrm{IB} \\ {}_\mathcal{B}\dot{\boldsymbol{\omega}}_\mathrm{IB} \\ \ddot{\boldsymbol{\varphi}}_\mathrm{j} \end{pmatrix} \in \mathbb{R}^{6+n_j} \tag{3-3}$$

As discussed in Section 2-6, the velocity of the torso is identical to the time-derivative of the position vector $\dot{\mathbf{r}}_\mathrm{B}$. Similarly, the generalized velocity of the internal joints can be described using the time-derivative of the joint positions $\dot{\boldsymbol{\varphi}}_\mathrm{j} = \frac{d\boldsymbol{\varphi}_\mathrm{j}}{dt}$.

The generalized Euler-rates $\dot{\boldsymbol{\theta}}$ are not equal to the angular velocity of the floating-base $\boldsymbol{\omega}_\mathrm{IB}$, but requires a projection matrix $\mathbf{E}_\mathrm{R} \in \mathbb{R}^{3\times3}$. This projection matrix depends on the selected

parameterization of the orientation and maps the time-derivative of the generalized orientation coordinate vector $\dot{\boldsymbol{\theta}}_{\mathrm{IB}}$ to the absolute angular velocity $\boldsymbol{\omega}_{\mathrm{IB}}$. Similarly, the acceleration vector $\ddot{\mathbf{q}}$ is typically described using the angular acceleration $\dot{\boldsymbol{\omega}}_{\mathrm{IB}} \in \mathbb{R}^3$. Again, the parameterized vector of the orientation is used to describe the absolute angular acceleration of the base. In Appendix A-1, the derivation of this mapping matrix and the conversion between the Euler rates and angular velocities are described.

3-3 Kinematic model

The general definitions of the kinematic relations of a robot have been discussed in Section 2-2. Below we will elaborate on a few kinematic relations that are important for this study.

3-3-1 Forward kinematic relations

As discussed, the Forward Kinematics model uses the geometric relation between frames to estimate the position of a certain frame. The Forward Kinematics (FK) model can be constructed by following the kinematic chain of the joints and substituting the joint variables with the joint states. In Figure 3-1, a three-dimensional schematic visualization is provided for the kinematic chain of the RF side of a quadruped robot, where a coordinate frame origin $\mathcal{O}_i$ is placed at every rotational joint. The joint position of each joint is represented by $\varphi \in \mathbb{R}$.

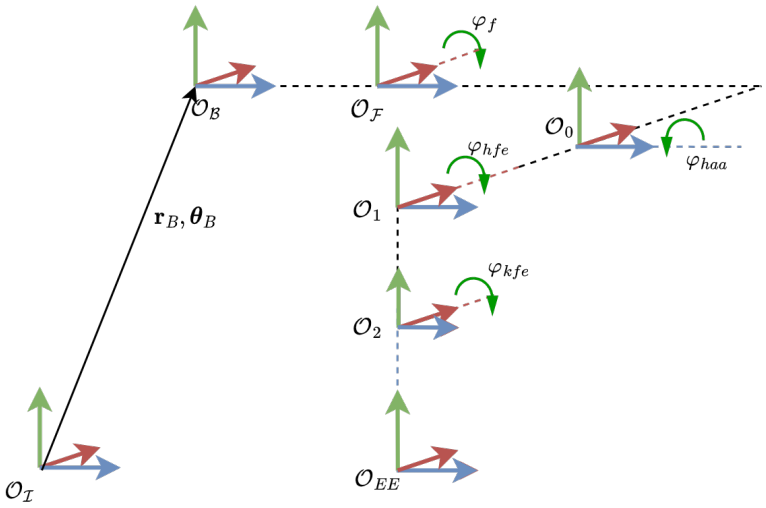


Figure 3-1: The joint frames of the RF side of a quadruped robot, with x=blue, y= red, z= green.

In order to establish the kinematic relation between the Inertial frame and the end-effector frame, first, the kinematic relation from one joint frame to the subsequent joint frame needs to be established. This allows for a convenient and structured approach to derive the transformation matrix $\mathbf{T}_{\mathcal{IEE}}$, which describes the net transformation from the inertial frame to the end-effector frame dependent on the joint positions $\boldsymbol{\varphi}$.

$$\boldsymbol{T}_{\mathrm{IEE}}(\mathbf{q}) = \boldsymbol{T}_{\mathrm{IB}}(\mathbf{r}_{\mathrm{IB}}, \boldsymbol{\theta}_{\mathrm{IB}})\boldsymbol{T}_{\mathrm{BF}}(\varphi_f)\boldsymbol{T}_{\mathrm{F0}}(\varphi_{\mathrm{haa}})\boldsymbol{T}_{01}(\varphi_{\mathrm{hfe}})\boldsymbol{T}_{12}(\varphi_{\mathrm{kfe}})\boldsymbol{T}_{\mathrm{2EE}} \tag{3-4}$$

This transformation matrix then provides information about the absolute position and orientation of the end-effector frame.

Differential kinematics

An important concept in differential kinematics and multi-body dynamics is the Jacobian matrix ${}_{\mathcal{A}}\mathbf{J} \in \mathbb{R}^{6\times n}$, which maps the joint rates to the spatial velocity of a frame with respect to frame $\mathcal{A}$. It must be noted that this Geometric Jacobian has a frame of reference and thus maps the joint rates to the spatial velocity with respect to that frame.

$$\mathbf{v}_{\mathrm{e}} = \mathbf{J}_{\mathrm{e}}(\mathbf{q})\dot{\mathbf{q}} \tag{3-5}$$

Furthermore, in order to derive the spatial acceleration of the end-effector $\mathbf{a}_{\mathrm{e}} \in \mathbb{R}^6$, the time-derivative of Equation (3-5) can be derived as:

$$\mathbf{a}_{\mathrm{e}} = \mathbf{J}_{\mathrm{e}}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{\mathrm{e}}(\mathbf{q})\dot{\mathbf{q}}, \tag{3-6}$$

3-3-2 Inverse kinematic relations

Inverse Kinematics (IK) is especially useful for motion planners that optimize end-effector trajectories instead of joint trajectories. IK converts the end-effector trajectories to joint-trajectories, that can then be tracked by individual joint controllers. With respect to the FK model, the Inverse Kinematics (IK) model of a quadruped robot is rather complicated to derive. Furthermore, the IK problem can be solved both analytically and numerically. However, while numeric derivations are much easier to compute, analytic derivations are much more efficient but require a thorough understanding of the kinematic relations.

For ordinary multi-legged systems, the inverse kinematic relations are typically derived from the base frame to the end-effector frame. The analytic approach uses geometric relations to find the three DOFs that move the end-effector in the desired location. These geometric relations are described in the [23].

For multi-legged robots with articulated torsos, the IK problem becomes even more complicated as the number of DOFs increase and two end-effectors are dependent on the spinal joint's DOF. However, when the state of the torso-joint is predefined, the IK problem reduces back to the ordinary IK. The full articulated IK solution then has the following structure:

$$\boldsymbol{\varphi}_{\mathrm{lj}} = \boldsymbol{f}_{\mathrm{IK}}(\mathbf{r}_{\mathrm{IB}}, \boldsymbol{\theta}_{\mathrm{IB}}, \boldsymbol{\varphi}_{\mathrm{tj}}), \tag{3-7}$$

where $\boldsymbol{f}_{\mathrm{IK}}()$ is a nonlinear function that computes the leg-joint angles.

Inverse differential kinematics

Besides leg-joint positions, the IK relations also needs to convert end-effector velocities and accelerations to individual joint velocities and accelerations. As discussed in Section 3-3-2, the trajectories of the base $\mathbf{r}_{\mathrm{b}}$, the end-effectors $\mathbf{r}_{\mathrm{ee}}(t)$, leg-joint angles $\boldsymbol{\varphi}_{\mathrm{tj}}$ and the spatial Jacobians $\mathbf{J}$ are commonly used to approximate the leg-joint velocities and accelerations. This relation can generally be inverted to compute the inverse relation.

When the matrix $\mathbf{J}$ is square and invertible, an efficient way to compute the solution is by using $\dot{\mathbf{q}} = \mathbf{J}^{-1}\mathbf{v}$. This is typically the case for square and non-redundant manipulators. For non-square and redundant manipulators, the pseudo-inverse of the Jacobian matrix $\mathbf{J}^{+} = (\mathbf{J}^{\intercal}\mathbf{J})^{-1}\mathbf{J}^{\intercal} \in \mathbb{R}^{6\times n}$ can minimize inverse problem by multiplying both sides by $\mathbf{J}^{\intercal}$.

$$\dot{\mathbf{q}} = \mathbf{J}_{\mathrm{e}}^{+}(\mathbf{q})\mathbf{v}_{\mathrm{e}}, \tag{3-8a}$$

$$\dot{\mathbf{q}} = \mathbf{J}_{\mathrm{e}}^{+}(\mathbf{q})(\mathbf{a}_{\mathrm{e}} - \dot{\mathbf{J}}_{\mathrm{e}}^{+}(\mathbf{q})\dot{\mathbf{q}}), \tag{3-8b}$$

This method is typically called the least-square method for inverse kinematics, as the computation of $\dot{\mathbf{q}}$ results in the following least-squares minimization:

$$\min ||\mathbf{J}^{\intercal}\mathbf{v} - \mathbf{J}^{\intercal}\mathbf{J}\dot{\mathbf{q}}||_2 \tag{3-9}$$

For articulated systems the differential IK method can be applied similarly. However, in this case the relative end-effector velocity with respect to the torso-joint velocity $\mathbf{v}_{\mathrm{jee}} = \mathbf{v}_{\mathrm{ee}} - \mathbf{v}_{\mathrm{tj}}$ and its relative spatial Jacobian is used $\mathbf{J}_{\mathrm{jee}} = \mathbf{J}_{\mathrm{ee}} - \mathbf{J}_{\mathrm{tj}}$.

$$\dot{\boldsymbol{\varphi}}_{\mathrm{lj}} = \mathbf{J}_{\mathrm{jee}}^{+}\mathbf{v}_{\mathrm{jee}} \tag{3-10a}$$

$$\ddot{\boldsymbol{\varphi}}_{\mathrm{lj}} = \mathbf{J}_{\mathrm{jee}}^{+}(\mathbf{a}_{\mathrm{jee}} - \dot{\mathbf{J}}_{\mathrm{jee}}\dot{\mathbf{q}}) \tag{3-10b}$$

It is important to note that the Jacobian matrix can become rank-deficient when a robot configuration $\mathbf{q}$ is singular. This singularity will result in infeasible joint velocities in the singular direction. However, the effect of the kinematic singularity is also experienced near singular configurations. Common methods to detect the singularity are listed in [24]. A more detailed description of the kinematic singularity and a method to avoid kinematic singular behavior is provided in Appendix A-4.

3-4 Conventional dynamic models

The system dynamics describe the (non-)linear relations between parameters of a system. Deriving these dynamic relations can help to provide insight into the underlying dependencies and interplay in complex systems and their subsystems. Moreover, dynamic models can be helpful for prediction and forward simulation. Finally, deriving a dynamic model can be extremely useful for controller design, especially for highly complex systems. The dynamic description can then be exploited to generate feasible trajectories and control actions or even to optimize the system design or behavior.

In literature, there is a vast amount of different types of dynamic models and derivations. The selection of the appropriate model highly depends on the application and available hardware. If a highly accurate model is preferred, the model becomes more complicated to derive and understand the dynamic relations. This may be fine for prediction and forward simulation, but often, it can be beneficial to consider predefining some model assumptions that soften the computational burden and make the model less complicated to understand. Especially for optimization and control objectives, an abstraction of a complex dynamic model is typically highly recommended. However, when abstracting a model, it is critical to understand what impact the modeling assumption has on the accuracy of your model. A trajectory or control

input computed with an inaccurate model might be infeasible in the actual system and lead to instability. In contrast, the controller or motion generator might not be able to compute the control actions or trajectories at the desired control frequency when the model is too complex.

In the case of a legged robot in general, the dynamic model is considered highly complex due to the highly nonlinear interplay between the large number of joints and the hybrid nature of interaction with the environment. In addition, the legged system is also equipped with a floating base, which implies that the six DOFs of the torso are considered unactuated. These DOFs can only be actuated indirectly by the interaction with the environment, the foot-ground interactions. With these dynamic characteristics of legged robots in general, one needs to take good consideration of the application and requirements when selecting an appropriate dynamic model. In the following section, a close look is given to the three most conventional dynamic models used for quadrupeds with a rigid torso. This analysis will help to gain more understanding about the perks and flaws of these models in the context of trajectory generation for quadrupeds with an articulated torso.

3-4-1 Rigid body dynamics

The RBD model can be derived using various principles from classical mechanics as described in [22]. The most well-known methods are the Newton-Euler method, based on the principle of virtual work, and the Euler-Lagrange method, which is primarily based on the conservation of energy.

The floating-base system is generally expressed in terms of the generalized coordinates $\mathbf{q} \in \mathbb{R}^n$, the generalized velocities $\dot{\mathbf{q}} \in \mathbb{R}^{n_q}$, the generalized torques $\boldsymbol{\tau} \in \mathbb{R}^{n_\tau}$ and the external forces acting on the system $\mathbf{F}_c \in \mathbb{R}^{3n_c}$. The dynamic model of floating-base systems like a quadruped robot is typically represented in the following form:

Rigid Body Dynamics (RBD) model

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{b}(\dot{\mathbf{q}}, \mathbf{q}) + \mathbf{g}(\mathbf{q}) = \mathbf{S}^\intercal \boldsymbol{\tau} + \mathbf{J}_c^\intercal \mathbf{F}_c, \tag{3-11}$$

with the Mass matrix $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n_q \times n_q}$, the Coriolis and Centrifugal vector $\mathbf{b}(\dot{\mathbf{q}}, \mathbf{q}) \in \mathbb{R}^{n_q}$, the Gravitational vector $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^{n_q \times n_q}$, the selection matrix of the actuated joints $\mathbf{S} \in \mathbb{R}^{n_\tau \times n_q}$, the contact Jacobian matrix $\mathbf{J}_c \in \mathbb{R}^{n_c \times n_q}$ of the location where the forces are applied.

A complete derivation of this model and its matrices can be found in Appendix A-2. For articulated robots, the RBD model can be derived using an identical approach to ordinary multi-body systems. Ultimately, the complexity of the RBD representation suffers significantly from the DOFs introduced by the torso joints.

3-4-2 Centroidal dynamics

As can be observed from the formulation of the rigid body dynamics model in Equation (A-9), the dynamic representation of the floating-base system is highly nonlinear and contains six

un-actuated DOFs. For applications such as model-based trajectory optimization, such complexities might impose a large computational burden on the motion planners. In such applications, an abstraction of the dynamic model of floating-base systems can provide a reduction in computational complexity.

The momentum-based approach is fundamental in rigid-body dynamics and is primarily based on the Newton-Euler law of rigid-body motions. This law states that the time-derivative of the spatial momentum of a body $\mathbf{h} \in \mathbb{R}^6$ is equal to the net external wrenches $\mathbf{w} \in \mathbb{R}^6$ acting on the body. By applying this law, we can derive an Equation of Motion (EoM) for a rigid body based on the body's spatial acceleration $\mathbf{a}$, spatial velocity $\mathbf{v}$ and its spatial Inertia tensor $\bar{\mathbf{I}}$.

$$\mathbf{w} = \frac{d}{dt}\mathbf{h} = \frac{d}{dt}(\bar{\mathbf{I}}\mathbf{v}) = \bar{\mathbf{I}}\mathbf{a} + \dot{\bar{\mathbf{I}}}\mathbf{v} = \bar{\mathbf{I}}\mathbf{a} + \mathbf{v} \times \bar{\mathbf{I}}\mathbf{v} \tag{3-12}$$

The Centroidal Dynamics (CD) model is a well-established approach that uses this method to derive the dynamics of a floating-base system based on the conservation of the net centroidal momentum of all the bodies. The approach evaluates the linear and angular spatial momentum of the centroidal point $\mathcal{O}_{\mathrm{G}}$, which lies in the composite Center of Mass (CoM) of the whole system. The following derivations are primarily based on the derivation presented in [25].

Spatial momentum

In order to find the net centroidal spatial momentum, first, the spatial momentum on each link needs to be described in a mutual reference frame. For convenience, we will use the Inertial frame $\mathcal{I}$ as our reference frame. The spatial momentum of body i is denoted by $\mathbf{h}_i \in \mathbb{R}^6$, and consists of the linear momentum $\mathbf{l}_i \in \mathbb{R}^3$ and the angular momentum $\mathbf{k}_i \in \mathbb{R}^3$. The spatial momentum of body i, is dependent on the spatial Inertia tensor $\bar{\mathbf{I}}_i \in \mathbb{R}^{6\times6}$ and its spatial velocity vector $\mathbf{v}_i \in \mathbb{R}^6$, both represented with respect to a common reference frame. Typically, the spatial momentum is computed about its own local coordinate frame [26].

$$\mathbf{h}_i = \begin{bmatrix} \mathbf{k}_i \\ \mathbf{l}_i \end{bmatrix} = \begin{bmatrix} \mathbf{I}_i + m_i\mathbf{S}(\mathbf{r}_{c,i})\mathbf{S}(\mathbf{r}_{c,i})^\intercal & m_i\mathbf{S}(\mathbf{r}_{c,i}) \\ m_i\mathbf{S}(\mathbf{r}_{c,i})^\intercal & m_i\mathbb{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_i \\ \boldsymbol{v}_i \end{bmatrix} = \bar{\mathbf{I}}_i\mathbf{v}_i \tag{3-13}$$

Here $\bar{\mathbf{I}}_i \in \mathbb{R}^{6\times6}$ is the spatial inertia for body i, $\mathbf{I}_i \in \mathbb{R}^{3\times3}$ is the standard Cartesian Moment of Inertia (MoI) tensor of body i around its CoM, and m_i its mass. Furthermore, $\mathbf{r}_{c,i} \in \mathbb{R}^3$ is the position vector to the CoM frame of the body. It is important to note that when the CoM frame is selected as the reference frame, the off-diagonal blocks reduce to zero.

Instead of expressing the spatial momentum in the spatial velocity, it is more convenient to express each body's momentum in terms of the generalized velocity vector $\dot{\mathbf{q}} \in \mathbb{R}^n$. In the context of spatial momentum, the spatial Jacobian matrix $\mathbf{J}_i \in \mathbb{R}^{6\times n}$ conveniently maps the generalized velocity vector to the spatial velocity of link i. In Section 2-6-4, the derivation of the spatial Jacobian was discussed.

$$\mathbf{h}_i = \bar{\mathbf{I}}_i\mathbf{J}_i\dot{\mathbf{q}} = \mathbf{A}_i\dot{\mathbf{q}} \tag{3-14}$$

Furthermore, we can define a momentum matrix for each link $\mathbf{A}_i \in \mathbb{R}^{6\times n}$, that maps the generalized velocity to the spatial momentum of link i.

Centroidal momentum

The spatial momentum $\mathbf{h}_i$ of each link is described in its own coordinate system. However, for analysis or control, it is more convenient to express the spatial momenta of all links into a common frame. In Centroidal Dynamics (CD), the instantaneous CoM or the centroid of the system $\mathcal{G}$ is selected as the common reference frame. This frame is positioned in the composite CoM frame $\mathcal{C}$, but with its axes aligned with the inertial frame $\mathcal{I}$. As the spatial momentum vector is considered to be a force-type vector, the momentum of each link needs to be transformed into the Centroidal frame using a force-type transformation matrix ${}^{G}\mathbf{X}_i^*$, as is visualized in Figure 3-2.

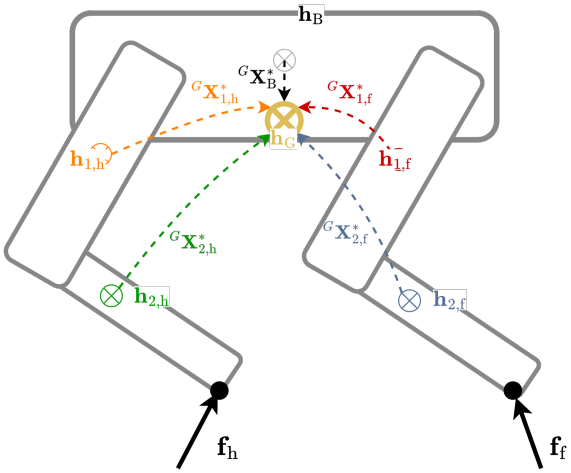


Figure 3-2: Schematic visualization of the spatial transformations from each link to the Centroidal frame $\mathcal{G}$.

Typically, the derivation of the force-type transformation matrix exploits the relation in Equation (2-14) by using the FK relations from the centroidal frame $\mathcal{G}$ to the link frame i. By summing each individual link momenta expressed in the common frame, we obtain the total net momentum in the centroidal frame, called the centroidal momentum $\mathbf{h}_\mathrm{G} \in \mathbb{R}^6$. A convenient way to derive the centroidal momentum is the derivation of the Centroidal Momentum Matrix (CMM) $\mathbf{A}_\mathrm{G} \in \mathbb{R}^{6\times n}$. As shown in the derivation in Equation (3-15), the CMM maps the generalized velocity vector $\dot{\mathbf{q}}$ to the centroidal momentum $\mathbf{h}_\mathrm{G}$.

$$\begin{aligned}\mathbf{h}_\mathrm{G} &= \sum_i^N {}^{\mathrm{G}}\mathbf{X}_i^* \mathbf{h}_i = \sum_i^N {}^{i}\mathbf{X}_\mathrm{G}^\top \mathbf{h}_i \\ &= \sum_i^N {}^{i}\mathbf{X}_\mathrm{G}^\top \bar{\mathbf{I}}_i \mathbf{J}_i \dot{\mathbf{q}} = \mathbf{A}_\mathrm{G}\dot{\mathbf{q}}\end{aligned} \tag{3-15}$$

Centroidal dynamics of a legged system

As mentioned by [26], dynamic whole-body controllers typically require second-order Centroidal Dynamics. This dynamic formulation is derived by using the Newton-Euler formulation $\mathbf{w} = \frac{d}{dt}\mathbf{h}$, which states that the time-derivative of the momentum in a frame that

coincides with the CoM, is equal to the net external spatial wrench $\mathbf{w}_{\mathrm{G}} = [\mathbf{n}_{\mathrm{G}}^{\intercal}, \boldsymbol{f}_{\mathrm{G}}^{\intercal}]^{\intercal} \in \mathbb{R}^6$ acting on system centroid, as described below.

$$\frac{d}{dt}\mathbf{h}_{\mathrm{G}} = \mathbf{w}_{\mathrm{G}} \tag{3-16a}$$

$$\frac{d}{dt}\left(\mathbf{A}_{\mathrm{G}}\dot{\mathbf{q}}\right) = \mathbf{A}_{\mathrm{G}}\ddot{\mathbf{q}} + \dot{\mathbf{A}}_{\mathrm{G}}\dot{\mathbf{q}} = \mathbf{w}_{\mathrm{G}} \tag{3-16b}$$

The net external wrench acting on the system centroid $\mathbf{w}_{\mathrm{G}}$, consists of all the external moments $\mathbf{n}_{\mathrm{G}}$ and forces $\boldsymbol{f}_{\mathrm{G}}$ acting on the centroid and can be constructed by transforming each local external wrench $\mathbf{w}_i$ onto the centroidal frame using the force-type spatial transformation matrix in Equation (2-15). The total wrench on the centroid for a legged system is described in Equation (3-17).

$$\mathbf{w}_{\mathrm{G}} = \mathbf{w}_{\mathrm{g}} + \sum_{i}^{n_c} {}^{i}\mathbf{X}_{\mathrm{G}}^{\intercal}\mathbf{w}_{\mathrm{c,i}} = \begin{bmatrix} \sum_i \boldsymbol{f}_{\mathrm{c,i}} \times (\mathbf{r}_G - \mathbf{p}_{\mathrm{c,i}}) \\ \boldsymbol{f}_{\mathrm{g}} + \sum_i \boldsymbol{f}_{\mathrm{c,i}} \end{bmatrix} \tag{3-17}$$

In legged systems, these external wrenches typically consist of the wrenches exerted on the feet $\mathbf{f}_{\mathrm{c,i}}$ and the wrenches due to the gravitational force $\mathbf{f}_{\mathrm{g}}$. The full dynamics of the centroid point are then described as follows:

Centroidal Dynamics model

$$\mathbf{A}_{\mathrm{G}}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{A}}_{\mathrm{G}}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{w}_{\mathrm{G}} \tag{3-18}$$

in which $\mathbf{r}_{\mathrm{G}}$ describes the absolute position of the centroid in the world frame, and $\mathbf{p}_{\mathrm{c,i}}$ the absolute position of the contact point in the world frame.

3-4-3 Single rigid body dynamics

While the CD model reduces the model complexity by disregarding the joint dynamics, the matrices $\mathbf{A}_{\mathrm{G}}(q), \dot{\mathbf{A}}_{\mathrm{G}}(q)$ are still highly nonlinear. This complexity might still introduce computational burdens on several motion planners. In order to reduce the model complexity even further, the Single Rigid Body Dynamics (SRBD) can provide a convenient abstraction of the CD model.

As discussed in [25], a new spatial velocity vector is introduced called the "average composite velocity" $\mathbf{v}_C$, which relates to the system velocity vector with a system transformation matrix through $\mathbf{v} = \mathbf{X}_C\mathbf{v}_C$. Furthermore, the composite full-body inertia is computed by transforming all link spatial inertias to a composite CoM-frame. This inertia tensor $\bar{\mathbf{I}}_C \in \mathbb{R}^{6\times6}$ is called the Composite Rigid Body Inertia (CRBI) matrix. This CRBI captures the total spatial inertia of all links combined into a single body at the composite CoM frame.

$$\bar{\mathbf{I}}_{\mathrm{C}} = \mathbf{X}_C^{\intercal}\bar{\mathbf{I}}_{\mathrm{C}}\mathbf{X}_C = \begin{bmatrix} \mathbf{I}_C(\mathbf{q}) & \mathbf{0} \\ \mathbf{0} & \sum m_i \end{bmatrix} \tag{3-19}$$

By interpreting the robot as a single rigid body with all joints fixed in a nominal position, the CD derivation can be applied to this single body. This results in the following expression for the centroidal momentum of the composite rigid body.

$$\mathbf{h}_{\mathrm{G}} = \mathbf{X}_{\mathrm{G}}^{\intercal}\bar{\mathbf{I}}_{\mathrm{C}}\mathbf{v}_{\mathrm{C}} = \mathbf{R}_{\mathcal{IB}}(\boldsymbol{\theta})\bar{\mathbf{I}}_{\mathrm{C}}(\boldsymbol{\varphi})\mathbf{R}_{\mathcal{BI}}(\boldsymbol{\theta})_{\mathcal{I}}\mathbf{v}_C \tag{3-20}$$

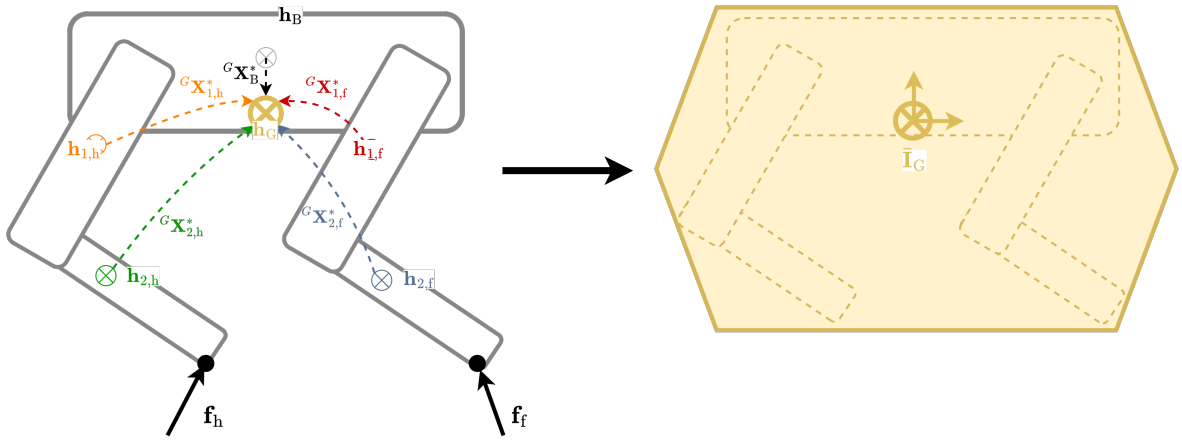


Figure 3-3: Visualisation of the SRBD model abstraction compared to the CD model.

Here, $\mathbf{v}_C$ is the spatial velocity of the composite CoM frame. This velocity typically depends on the joint velocities $\boldsymbol{\varphi}$ as well. However, in SRBD, the impact of joint velocities on the centroidal momentum is typically neglected. Instead, the velocity of the base is used $\mathbf{v}_C \approx \mathbf{v}_B$. Furthermore, the impact of the joint positions on the full-body MoI is typically also neglected, such that $\mathbf{I}_C(\boldsymbol{\varphi}(t)) \approx \mathbf{I}_C(\boldsymbol{\varphi}_{nom})$. By using the Newton-Euler law of rigid-body motions as in Equation (3-12), a second-order SRBD model is obtained:

Single Rigid Body Dynamics (SRBD) model

$$\mathbf{I}(\boldsymbol{\theta})\dot{\boldsymbol{\omega}}_{IB} + \boldsymbol{\omega}_{IB} \times \mathbf{I}(\boldsymbol{\theta})\boldsymbol{\omega}_{IB} = \sum_{i=1}^{n_i} \boldsymbol{f}_{c,i} \times (\mathbf{r}_{IB} - \mathbf{r}_{c,i}) \tag{3-21a}$$

$$m\ddot{\boldsymbol{r}} = m\boldsymbol{g} + \sum_{i=1}^{n_i} \boldsymbol{f}_{c,i} \tag{3-21b}$$

The assumptions can introduce large discrepancies between the actual system dynamics and the model. These assumptions are generally related to the leg-base mass distribution. For instance, the assumption that the momentum produced by the leg-joint velocities can only be justified when the mass or inertia of the legs is low compared to the mass of the torso. Furthermore, the fluctuation of the composite CoM position also depends on the mass distribution of the legs. Therefore, we must be careful when using the SRBD derivation when dealing with robots with a high leg-base distribution.

3-4-4 Conventional models for articulated torso

In this section, all three conventional models discussed above are evaluated in terms of the complexity, the number of dimensions, and their ability to incorporate the articulation of the torso.

As discussed, the RBD model is the most accurate of all models presented in Section 3-4. It derives an EoM for all n states and thus describes all dynamic relations in the system.

However, for high-dimensional and nonlinear systems, the model typically becomes impossible to evaluate or use as a dynamic constraint. A quadruped with an articulated torso can easily be modeled using the RBD model as shown in [15]. However, the computation time can drastically increase.

With respect to the Rigid Body Dynamics (RBD) model, the CD model eliminates n_j joint EoMs and only evaluates the six-dimensional change in spatial momentum of the CoM. By neglecting the joint dynamics, the number of equations needed to be solved is vastly reduced. In the context of motion generation, the exclusion of the joint dynamics implies that the control problem needs to be decoupled and that a separate controller is required to determine the required actuator torques. The CD derivation provides a good representation of the impact of the joints on the base. However, similar to the RBD approach, this method still results in a highly nonlinear dynamic representation due to all nonlinear dependencies in the system. Therefore, the computational time can also become a cumbersome issue for articulated robots.

When investigating whether the SRBD model abstraction is suitable for a quadruped robot with an articulated torso, we can quickly conclude that this type of model is not suitable for this particular type of robot. When decoupling the torso into smaller segments, the SRBD models would stay identical. Furthermore, a Trajectory Optimization (TO) algorithm would not be able to exploit the additional DOFs of the articulated torso. However, the approach of neglecting the momentum produced by low inertia links is an interesting concept that can be exploited for the derivation of a low-dimensional dynamic model for articulated quadruped robots. Evidently, in order to develop a computationally efficient TO algorithm, a dynamic representation is needed that is computationally uncomplicated but also represents the dynamics of the articulated torso accurately.

3-5 Dynamic model for articulated robots

As mentioned in the previous sections, the task of finding a low-dimensional dynamic model for an ordinary quadruped robot can be rather challenging for numerous reasons. Unfortunately, the integration of additional DOFs in the base frame increases the nonlinearity of the system behavior. This suggests that a model abstraction might be even more imperative. Evidently, it is crucial to find an optimal level of model reduction that minimizes the dimensionality and maximizes the model accuracy. Therefore, a good understanding of the impact of each assumption on the complexity and accuracy of the model is essential.

3-5-1 Triple rigid body dynamics

The main objective of this study is to find an accurate and low-dimensional modeling approach that captures the dynamics of the articulated torso and does not impose a large computational burden on the trajectory optimization task. By examining the derivation of other conventional models, an alternative approach was derived: the Triple Rigid Body Dynamics (TRBD). Similar to the SRBD model abstraction, this approach assumes that the leg dynamics have no significant effect on the momentum of the centroid.

Derivation

In general, this model evaluates the change in centroidal momentum while using the constant composite inertia $\tilde{\mathbf{I}}_{\text{side}}$ for the front and hind segments of the robot. As visualized in Figure 3-4, the spatial inertia of each individual link is projected to a common frame for the front and hind legs. This abstraction eliminates the dependency on the leg-joint positions and velocities, thus vastly reducing the number of dimensions and nonlinearity of the dynamic representation.

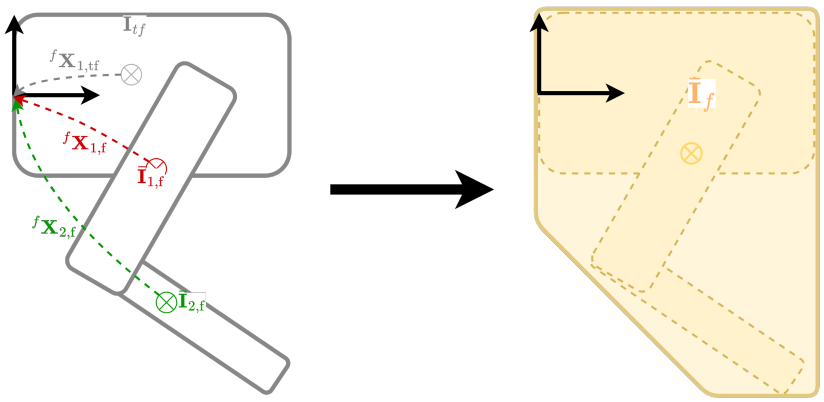


Figure 3-4: Schematic representation of a lumped front side abstraction.

The composite spatial Inertia tensor $\bar{\mathbf{I}}_{\text{side}}$ is the perceived spatial inertia of the lumped mass at the section of the robot. The leg-joint positions are assumed to be constant and equal to the nominal positions, similar to the SRBD approach.

$$\bar{\mathbf{I}}_{\text{f}}(\boldsymbol{\varphi}_{\text{lj,f}}(t)) \approx \tilde{\mathbf{I}}_{\text{f}}(\boldsymbol{\varphi}(t)_{\text{lj,f,nom}}) := \sum_{i=0}^{7} {}^{\text{i}}\mathbf{X}_{\text{f}}^{\text{T}} \bar{\mathbf{I}}_{\text{i}} {}^{\text{i}}\mathbf{X}_{\text{f}} \tag{3-22}$$

With this abstraction, various segments of the robot can be coupled into a single mass. The number of bodies that need to be evaluated is then drastically reduced. This abstraction of the leg dynamics results in a CD model that represents three individual bodies: the base, the front section, and the hind section of the torso. Similar to the CD approach, first, the Centroidal Momentum $\mathbf{h}_{\text{G}}$ is derived as shown in Equation 3-23. However, eliminating the leg joints enables a derivation of the centroidal momentum with much less effort than the CD model. For convenience of notation, the transformation matrix projecting all momentum to the centroidal frame ${}^{\text{G}}\mathbf{X}^{*}$ is decomposed into two transformation matrices. The transformation matrix ${}^{\text{C}}\mathbf{X}^{*}(\boldsymbol{\varphi}_{\text{tj}})$ projects all link momentum onto the CoM frame aligned with the body frame, while the transformation matrix ${}^{\text{G}}\mathbf{X}_{C}(\boldsymbol{\theta})$ aligns the coordinate frame with the Inertial frame.

The derivation of the centroidal momentum starts by projecting the momentum of all three bodies onto a common frame. The most straightforward choice would be the CoM frame directly or projecting all link momentum onto the base frame and then projecting it onto the CoM frame, as done so in Equation (3-23). Similar to the CD derivation, we can describe each link momentum in terms of the generalized velocity vector using a Jacobian Matrix.

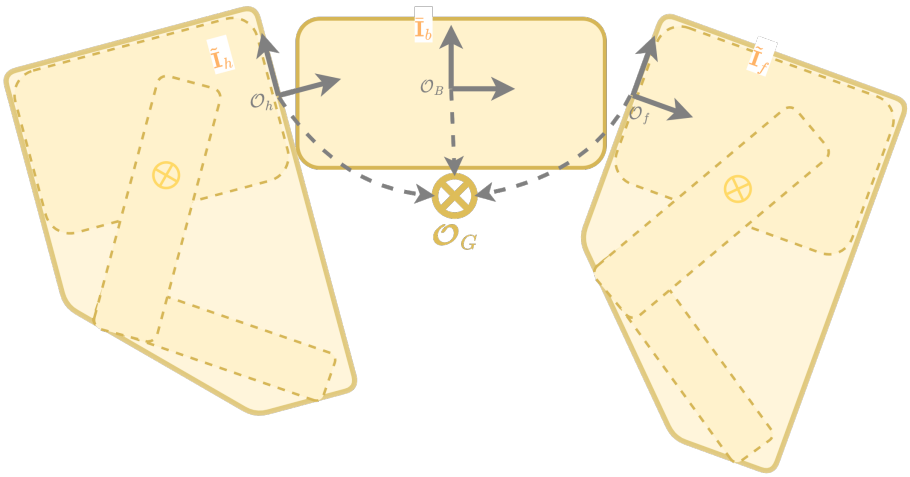


Figure 3-5: Schematic representation of the TRBD model and its three composite bodies: $\mathcal{H}$ (hind), $\mathcal{B}$ (middle), $\mathcal{F}$ (front).

Ultimately, we can describe the CMM $\mathbf{A}_{\mathrm{G}}$ as a product of the spatial rotation matrix ${}^{\mathrm{G}}\mathbf{X}^{*}_{\mathrm{C}}(\boldsymbol{\theta})$ and the CoM Momentum Matrix $\mathbf{A}_{\mathrm{C}}$.

$$\mathbf{h}_G = {}^{G}\mathbf{X}^{*}_{\mathrm{B}}(\mathbf{h}_{\mathrm{B}} + {}^{\mathrm{B}}\mathbf{X}^{*}_{\mathrm{F}}\mathbf{h}_{\mathrm{F}} + {}^{\mathrm{B}}\mathbf{X}^{*}_{\mathrm{H}}\mathbf{h}_{\mathrm{H}}) \tag{3-23a}$$

$$\mathbf{h}_G = {}^{\mathrm{G}}\mathbf{X}^{*}_{\mathrm{B}}(\bar{\mathbf{I}}_{\mathrm{B}}\mathbf{v}_{\mathrm{B}} + {}^{\mathrm{B}}\mathbf{X}^{*}_{\mathrm{F}}\bar{\mathbf{I}}_{\mathrm{F}}\mathbf{v}_{\mathrm{F}} + {}^{\mathrm{B}}\mathbf{X}^{*}_{\mathrm{H}}\bar{\mathbf{I}}_{\mathrm{H}}\mathbf{v}_{\mathrm{H}}) \tag{3-23b}$$

$$= {}^{\mathrm{G}}\mathbf{X}^{*}_{\mathrm{B}}\left(\bar{\mathbf{I}}_{\mathrm{B}}\mathbf{J}_{\mathrm{B}} + {}^{\mathrm{B}}\mathbf{X}^{*}_{\mathrm{F}}\bar{\mathbf{I}}_{\mathrm{F}}\mathbf{J}_{\mathrm{F}} + {}^{\mathrm{B}}\mathbf{X}^{*}_{\mathrm{H}}\bar{\mathbf{I}}_{\mathrm{H}}\mathbf{J}_{\mathrm{H}}\right)\dot{\mathbf{q}} \tag{3-23c}$$

$$= {}^{\mathrm{G}}\mathbf{X}^{*}_{\mathrm{C}}(\boldsymbol{\theta})\underbrace{{}^{\mathrm{C}}\mathbf{X}^{*}_{\mathrm{B}}(\boldsymbol{\varphi}_{\mathrm{tj}})\left(\bar{\mathbf{I}}_{\mathrm{B}}\mathbf{J}_{\mathrm{B}} + {}^{\mathrm{B}}\mathbf{X}^{*}_{\mathrm{F}}(\varphi_{\mathrm{f}})\bar{\mathbf{I}}_{\mathrm{F}}\mathbf{J}_{\mathrm{F}}(\varphi_{\mathrm{f}}) + {}^{\mathrm{B}}\mathbf{X}^{*}_{\mathrm{H}}(\varphi_{\mathrm{h}})\bar{\mathbf{I}}_{\mathrm{H}}\mathbf{J}_{\mathrm{H}}(\varphi_{\mathrm{h}})\right)}_{\mathbf{A}_{\mathrm{C}}(\boldsymbol{\varphi}_{\mathrm{tj}})}\dot{\mathbf{q}} \tag{3-23d}$$

With respect to the CD model, the description of the spatial Jacobian $\mathbf{J}$ and the spatial transformation matrices are more convenient to derive. The CMM matrix $\mathbf{A}_{\mathrm{C}}$ then provides the relationship between the joint rates and the net momentum in the CoM frame. The decomposition of transformation matrix ${}^{\mathrm{G}}\mathbf{X}^{*}$ provides a more convenient representation and simplifies the analytical derivation of the rate of centroidal momentum, $\dot{\mathbf{h}}_{\mathrm{G}}$.

For clarity we will denote this spatial transformation that is purely based on the rotation from the CoM frame aligned with the base from to the centroidal frame ${}^{C}\mathbf{X}^{\intercal}_{\mathrm{G}}$ and its time-derivative ${}^{C}\dot{\mathbf{X}}^{\intercal}_{\mathrm{G}}$, by $\bar{\mathbf{R}}$ and $\dot{\bar{\mathbf{R}}}$ respectively. These angular transformation matrices are provided as follows:

$$\bar{\mathbf{R}}(\boldsymbol{\theta}) = \begin{bmatrix} \mathbf{R}_{\mathcal{IB}}(\boldsymbol{\theta}) & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\mathcal{IB}}(\boldsymbol{\theta}) \end{bmatrix}, \quad \dot{\bar{\mathbf{R}}}(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{\omega}_{IB} \times \mathbf{R}_{\mathcal{IB}}(\boldsymbol{\theta}) & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\omega}_{IB} \times \mathbf{R}_{\mathcal{IB}}(\boldsymbol{\theta}) \end{bmatrix} \tag{3-24}$$

By applying Equation (3-16), we can derive the time-derivative of the centroidal momentum and find the abstracted EoM for the centroidal momentum of the articulated robot. With the parameterization described in Equation (3-3), the TRBD model is described in Equation (3-25). Here, the external wrench on the centroid $\mathbf{w}_{\mathrm{G}}$ is described as a function of the contact forces and contact positions as shown in Equation (3-17).

Triple Rigid Body Dynamics model

$$\begin{aligned} &\mathbf{A}_{\mathrm{G}}\ddot{\mathbf{q}} + \dot{\mathbf{A}}_{\mathrm{G}}\dot{\mathbf{q}} = \mathbf{w}_{\mathrm{G}} \\ \text{with}: \quad &\mathbf{A}_{\mathrm{G}} = \bar{\mathbf{R}}(\boldsymbol{\theta})\mathbf{A}_{\mathrm{C}}(\boldsymbol{\varphi}_{\mathrm{tj}}) \\ &\dot{\mathbf{A}}_{\mathrm{G}} = \dot{\bar{\mathbf{R}}}(\boldsymbol{\theta}, \boldsymbol{\omega})\mathbf{A}_{C}(\boldsymbol{\varphi}_{\mathrm{tj}}) + \bar{\mathbf{R}}(\boldsymbol{\theta})\dot{\mathbf{A}}_{C}(\dot{\boldsymbol{\varphi}}_{\mathrm{tj}}, \boldsymbol{\varphi}_{\mathrm{tj}}) \end{aligned} \tag{3-25}$$

Comparison with ordinary dynamic representations

In essence, the derivation of the TRBD model in Equation 3-25 is identical to the CD formulation in Equation (3-18). Moreover, as described in Equation (3-26), when the leg-joint positions are fixed in the nominal position, the CD model reduces to the TRBD model. However, the elimination of the dependency on the leg joints in the TRBD model vastly reduces the number of dimensions of the model. Instead of evaluating the spatial momentum of $n_{tb} + 4n_{lb}$ individual links, this representation only requires the evaluation of n_{tb} links. Furthermore, the abstraction leads to the elimination of $4n_l b$ columns of the CMM.

$$\mathbf{A}_{C}\left(\boldsymbol{\varphi}\right) \approx \tilde{\mathbf{A}}_{C}\left(\boldsymbol{\varphi}_{\mathrm{tj}}\right) \tag{3-26}$$

The proposed modeling approach is closely related to the leg centroidal model proposed in [16], where the centroidal momentum and dynamics are generated by evaluating the centroidal properties of the torso and abstraction of each individual leg. However, [16] focuses more on the centroidal properties of the legs, especially for systems that do not adhere to the massless limb assumption. Our approach is most useful for legged robots with an articulated torso that adheres to the massless limb assumption.

Similar to the abstraction of the SRBD representation, a CRBI matrix approximates the inertia of a composition of individual links. However, the TRBD model applies the same approach but with more consideration of the momentum produced by the torso-joint dynamics. In contrast to the SRBD model, the CRBI matrix $\bar{\mathbf{I}}_{\mathrm{C}}$ of the TRBD model is also dependent on the torso joint positions $\boldsymbol{\varphi}_{\mathrm{tj}}$. While the SRBD model is extremely useful for legged systems with a rigid torso that does adhere to the massless-limb assumption, it is not a valid representation for legged systems with an articulated torso. In general, for such systems, the momentum produced by the rotation of the torso joint is significant and cannot be abstracted.

Chapter 4

Trajectory Optimization for Articulated Robots

The following chapter demonstrates how a Trajectory Optimization (TO) can be adapted to generate articulated torso trajectories. Conveniently, a number of open-source motion optimization frameworks are available for legged systems, such as the Trajectory Optimizer for Walking Robots (TOWR) Library [13] and Optimal Control for Switched Systems (OCS2) Library [27]. These C++ libraries solve an Nonlinear Programming (NLP) to optimize the motion of a legged robot. The TOWR library is a TO algorithm, that searches for a feasible torso and end-effector trajectories to travel from an initial torso position to a final torso position. As this toolbox allows gait optimization and can compute rather complex maneuvers, the library is mainly used for offline TO. In contrast, the OCS2 toolbox optimizes the planned motion of the legged robot by predefining the switching times and contact locations. This algorithm is, therefore, capable of optimizing the motion at a much faster rate and can therefore be used in real time. For this study the TOWR library is adapted to generate trajectories for an articulated robot.

4-1 Trajectory optimization formulation

The NLP formulation derived for the articulated quadruped robot is presented in Equation (4-2). It might be noticed that the formulation for ordinary legged robots, presented in [13, Figure 5.2], is very similar to the articulated NLP problem in Equation (4-2), with the exception of the lines in blue and the exclusion of the phase-durations ΔT_i. The optimization of the phase durations allows each end-effector to alter its contact schedule. When all four end-effectors are allowed to freely modify their contact schedule, the NLP can optimize its own gait schedule. However, as these decision variables and constraints are not directly dependent on the articulation of the torso, it is excluded from this discussion for simplicity. In order to fully comprehend the behavior of the articulated NLP problem and how it differs from the ordinary NLP problem, we will also elaborate on the ordinary problem formulation. Furthermore, the NLP described in Equation 4-2 terminates when it has found a feasible trajectory

that satisfies all constraints. However, several other motion optimization formulations supply an objective function to the NLP, with the aim of minimizing that function. Typical objective functions aim to minimize energy consumption (joint torques, contact forces) or other performance measures. However, as is argued in [13], this typically increases the computational burden and leads to large computation times. To keep the algorithm time-efficient, the NLP will also terminate when it has found a feasible trajectory that satisfies all constraints.

4-1-1 Phase based parameterization

The set of continuous decision parameters of the NLP problem in Equation (4-1) are the position vector of the middle segment of the body frame $\mathbf{r}_{\text{ib}}(t)$ and ZYX (yaw-pitch-roll) orientation $\boldsymbol{\theta}_{\text{IB}}(t)$, the foothold positions $\mathbf{p}_c(t)$ and contact forces $\mathbf{f}_c(t)$ for every foot. Furthermore, two continuous decision variables are included that represent the front and hind torso joint angles $\boldsymbol{\varphi}_{\text{tj}}(t) = [\varphi_{\text{f}}, \varphi_{\text{h}}]^\intercal \in \mathbb{R}^2$. We will use $\mathbf{x}(t)$ to denote the optimization variables, excluding the phase durations.

$$\mathbf{x}(t) = [\mathbf{r}_{\text{ib}}^\intercal(t), \boldsymbol{\theta}_{\text{IB}}^\intercal(t), \mathbf{f}_c^\intercal(t), \mathbf{p}_{\text{c}}^\intercal(t), \varphi_{\text{f}}^\intercal(t), \varphi_{\text{h}}^\intercal(t)]^\intercal \tag{4-1}$$

In the TO problem, the orientation of the base is represented using optimized XYZ (roll-pitch-yaw) Euler-angles. These optimized Euler-angles describe any orientation by only three variables $\boldsymbol{\theta} : \{\phi, \theta, \psi\} \in \mathbb{R}^3$. The definition of the rotation matrix $\mathbf{R}_{\mathcal{IB}}(\phi, \theta, \psi)$ is described in Appendix A-1. Furthermore, as discussed in Section 3-2, in order to obtain the absolute base angular velocity $\boldsymbol{\omega}_{\text{IB}}$ and acceleration $\dot{\boldsymbol{\omega}}_{\text{IB}}$ used in Section 3-5, an Euler mapping matrix $\mathbf{E}_{\text{R}}(\boldsymbol{\theta})$ is used like in Equation (A-2). The derivation of this Euler mapping matrix is discussed in Appendix A-1.

As discussed in Section 2-5-1, the continuous trajectory must be parameterized to a finite set of decision variables. Therefore, each trajectory is parameterized by a set of cubic polynomials in accordance with the direct collocation method discussed in Section 2-5-1. Furthermore, as discussed in Section 2-5-1, the discontinuous nature of the contact dynamics is captured using the phase-based parameterization presented in [13]. By including the set of contact durations $\Delta\mathbf{T}$ to the set of decision variables, the solver can generate new gait patterns.

Problem formulation

The formulation in Equation (4-2) describes the general description of the proposed NLP for the quadruped robot with additional Degrees of Freedom (DOFs) in the torso. One of the primary objectives of this study was to limit the number of modifications needed. As can be seen below, only the three lines in blue need revision to incorporate the articulation of the torso. As can be seen below, the aim of the NLP is to find a dynamically feasible motion in the user-allocated time-interval $t \in [t_0, t_f]$, in terms of its dynamic optimization variables

$\mathbf{x}(t)$.

$$
\begin{aligned}
\text{find} \quad & \mathbf{r}(t) \in \mathbb{R}^3, \quad \forall t \in [t_0, t_f] \\
& \boldsymbol{\theta}(t) \in \mathbb{R}^3 \\
& \boldsymbol{\varphi}_{tj}(t) \in \mathbb{R}^{n_{tj}} && \text{(Torso joint variables)} \\
& \text{for every foot } i: \\
& \quad \mathbf{p}_i(t) \in \mathbb{R}^3 \\
& \quad \mathbf{f}_i(t) \in \mathbb{R}^3 \\
\text{s.t.} \quad & [\mathbf{r}, \boldsymbol{\theta}](t_0) = [\mathbf{r}_0, \boldsymbol{\theta}_0] \\
& \mathbf{r}(t_N) = \mathbf{r}_f \\
& [\ddot{\mathbf{r}}, \dot{\boldsymbol{\omega}}]^\intercal = \mathbf{F}\left(\mathbf{r}, \boldsymbol{\theta}, \boldsymbol{\varphi}_{\text{tj}}, \mathbf{p}_1, ..., \mathbf{f}_1, ...\right) && \text{(Dynamic constraint)} \\
& |\ddot{\boldsymbol{\varphi}}_{\text{tj}}(t)| \leq \ddot{\boldsymbol{\varphi}}_{\text{tj,lim}}, \quad |\dot{\boldsymbol{\varphi}}_{\text{tj}}(t)| \leq \dot{\boldsymbol{\varphi}}_{\text{tj,lim}} && \text{(Torso joint limits)} \\
& \boldsymbol{\varphi}_{\text{tj,min}} \leq \boldsymbol{\varphi}_{\text{tj}}(t) \leq \boldsymbol{\varphi}_{\text{tj,max}} \\
& \text{for every foot } i: \\
& \quad \mathbf{p}_i(t) \in \mathcal{R}_i(\mathbf{r}, \boldsymbol{\theta}, \boldsymbol{\varphi}_{tj}) && \text{(Kinematic constraint)} \\
& \quad \text{if foot } i \text{ in } \textbf{contact } (t \in \mathcal{C}_i): \\
& \quad\quad \dot{\mathbf{p}}_i(t) = \mathbf{0} \\
& \quad\quad p_i^z(t) = h_{terrain}(\mathbf{p}_i^{xy}) \\
& \quad\quad \mathbf{f}_i(t) \cdot \mathbf{n}(\mathbf{p}_i^{xy}) \geq 0 \\
& \quad\quad \mathbf{f}_i(t) \in \mathcal{F}(\mu, \mathbf{n}, \mathbf{p}_i^{xy}) \\
& \quad \text{if foot } i \text{ in } \textbf{air } (t \notin \mathcal{C}_i): \\
& \quad\quad \mathbf{f}_i(t) = \mathbf{0}
\end{aligned}
\tag{4-2}
$$

The NLP above is dependent on a set of motion constraints. These constraints are described in detail in [13] but will be briefly discussed here. The first two constraints in Equation (4-2) describe the initial and final pose of the base frame. These initial and final poses are arbitrary and are selected by the user before the optimization. Furthermore, the third constraint is called the dynamic constraint and enforces the dynamic feasibility of the base acceleration. Furthermore, for every foot, there is the kinematic constraint, which specifies the reachable space $\mathcal{R}_i$ of each foot i. This constraint is further discussed in Section 4-2-1.

Finally, there are a number of contact-dependent constraints that are quite intuitive, and some are also apparent from Figure 2-7. Namely, when a foot is in contact with the ground $t \in \mathcal{C}_i$, the velocity of the end-effector needs to be zero. Furthermore, the vertical position of the foot p_i^z is equal to the height of the terrain at the horizontal position of the foot $\mathbf{p}_i^{xy}$. Moreover, the terrain surface cannot generate a pulling force on the foot. This implies that the product between the contact force and the normal of the ground surface at the horizontal position of the foot can only be positive. Finally, a slip-constraint is introduced that specifies that the contact force stays within the friction cone $\mathcal{F}$. This constraint is dependent on the friction coefficient μ of the terrain and on the normal of the ground surface and constrains the force to be consistent with the no-slip constraint. The final foot constraint is that when the foot is in the swing phase, the surface cannot generate any contact force on the foot.

4-2 Adaptations for articulated torso

In the previous sections, the general overview of the torso-articulated framework adapted from [13] is presented. However, in the following section, we will elaborate on the required modifications needed to leverage the articulation of the torso. First of all, the decision variables

representing the torso joints φ_{tj} are parameterized using a cubic spline parameterization as discussed in Section 4-1-1. First, the most intuitive constraint is the inclusion of any actuator limits like joint position, velocity, or acceleration.

4-2-1 Kinematic constraint

In TO problems, the kinematic constraint evaluates whether a foot position $\mathbf{p}_i$ lies within the possible configuration space $\mathcal{R}_i$ of that foot. Typically, the Inverse Kinematics (IK) model is used to compute the necessary joint positions and to check whether the joint positions exceed the joint limits.

$$\boldsymbol{f}_{\text{IK}}(\mathbf{r}, \boldsymbol{\theta}, \mathbf{p}_i) \leq \boldsymbol{\varphi}_{j,lim} \tag{4-3}$$

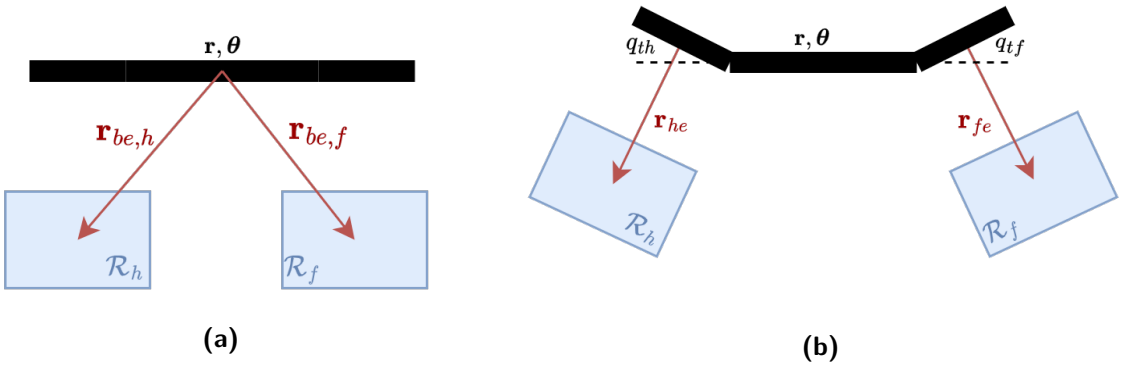


Figure 4-1: Schematic representation of kinematic constraint for legged robots with a) rigid torso, b) articulated torso. In the newly proposed kinematic constraint, the torso joint positions q_{tj} are considered while defining the Range of Motion (ROM) $\mathcal{R}$.

However, as observed in Section 3-3-2, the derivation of the IK model can be quite cumbersome and the IK model is also highly non-linear. The implementation of the inverse kinematics model as a kinematic constraint would intensify the computational demand of the NLP problem. To loosen the computational demand, [13] proposes a kinematic constraint abstraction by describing the ROM $\mathcal{R}_i(\mathbf{r}, \boldsymbol{\theta}) \in \mathbb{R}^3$ in Cartesian Space. The leg's configuration space is then reduced to a three-dimensional box with a constant position relative to the robot's torso position and orientation, as shown in Figure 4-1a. Therefore, the three-dimensional box is expressed in the body frame, with a constant center and constant boundaries. The kinematic constraint then evaluates whether the current position of the foot with respect to the base $\mathbf{r}_{bee}$ lies inside the ROM. In [13], this constraint is denoted as:

$$\mathbf{p}_i(t) \in \mathcal{R}_i(\mathbf{r}, \boldsymbol{\theta}) \Leftrightarrow |\mathbf{r}_{\text{be}}(\mathbf{r}, \boldsymbol{\theta}) - \mathbf{r}_{\text{be,nom}}| < \mathbf{b}, \tag{4-4}$$

where, $\mathbf{b} \in \mathbb{R}^3$ are the boundaries in each direction and $\mathbf{r}_{\text{be,nom}}$ is the position of the center of the three-dimensional box.

For quadruped robots with an articulated torso, this abstraction of the leg's kinematic limits would be justifiable as well. However, as discussed in Section 2-4, the spine's flexibility allows the cheetah to increase the reachable space of its feet. To leverage this biological principle

in this TO, the ROM of the end-effector needs to consider the torso joint deflection as well. Therefore, the end-effector positions are evaluated with respect to its parent torso-joint frame $\mathbf{r}_{je}$ using the following Forward Kinematics (FK) relations.

$$\mathbf{r}_{\mathrm{je}} = \mathbf{R}_{\mathcal{JB}}(\boldsymbol{\varphi}_{\mathrm{tj}})(\mathbf{R}_{\mathcal{BI}}(\boldsymbol{\theta})(\mathbf{p}_{\mathrm{i}} - \mathbf{r}_{\mathrm{ib}}) - \mathbf{r}_{\mathrm{bj}}) \tag{4-5a}$$

$$\mathbf{r}_{\mathrm{je,nom}} = \mathbf{R}_{\mathcal{JB}}(\boldsymbol{\varphi}_{\mathrm{tj}})(\mathbf{r}_{\mathrm{be,nom}} - \mathbf{r}_{\mathrm{bj}}) \tag{4-5b}$$

The kinematic constraint $\boldsymbol{g}_{\mathrm{k}}$ in Equation (4-4) can easily be adapted to the articulated formulation.

Kinematic Constraint Abstraction for the Articulated Torso

$$\mathbf{p}_i(t) \in \mathcal{R}_i(\mathbf{r}, \boldsymbol{\theta}, \boldsymbol{\varphi}) \iff \mathbf{g}_{\mathrm{k}} : |\mathbf{r}_{\mathrm{jee}}(\mathbf{r}, \boldsymbol{\theta}, \boldsymbol{\varphi}) - \mathbf{r}_{\mathrm{jee,nom}}(\boldsymbol{\varphi})| - \mathbf{b} < \mathbf{0} \tag{4-6}$$

Here, the rotation matrix and the translation vector from the base frame $\mathcal{B}$ to the torso-joint frame are denoted by $\mathbf{R}_{\mathcal{JB}}$ and $\mathbf{r}_{\mathrm{bj}}$ respectively.

4-2-2 Dynamic constraint

As was concluded in Section 3-4-3, the Single Rigid Body Dynamics (SRBD) model used as the dynamic constraint in [13], cannot leverage the articulation of the torso to find a feasible trajectory. In contrast, the dynamic representation derived in Section 3-5 does allow to find feasible torso joint trajectories. The integration of this model into the NLP requires an analytic derivation of the Centroidal Momentum Matrix (CMM) matrix and its time derivative. However, it is also essential for a correct implementation to consider the selected parameterization for $\ddot{\mathbf{q}}, \dot{\mathbf{q}}$ and their relationship to the decision variables $\mathbf{x}(t)$ in Equation (4-1). The relationships in Equation (4-7), can be derived from Equation (2-9)-(2-10) and the Euler-angle parameterization discussed in Appendix A-1.

$$\ddot{\mathbf{q}} = \begin{bmatrix} {}_{\mathcal{B}}\dot{\boldsymbol{\omega}}_{IB} \\ {}_{\mathcal{B}}\boldsymbol{a}_{IB} \\ \ddot{\boldsymbol{\varphi}}_{\mathrm{tj}} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathcal{BI}}(\boldsymbol{\theta})(\mathbf{E}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \dot{\mathbf{E}}(\dot{\boldsymbol{\theta}}, \boldsymbol{\theta})\dot{\boldsymbol{\theta}}) \\ \mathbf{R}_{\mathcal{BI}}(\boldsymbol{\theta})(\ddot{\mathbf{r}}_{\mathrm{IB}} + \dot{\mathbf{r}}_{\mathrm{IB}} \times \boldsymbol{\omega}_{\mathrm{IB}}) \\ \ddot{\boldsymbol{\varphi}}_{\mathrm{tj}} \end{bmatrix}, \dot{\mathbf{q}} = \begin{bmatrix} {}_{\mathcal{B}}\boldsymbol{\omega}_{IB} \\ {}_{\mathcal{B}}\boldsymbol{v}_{IB} \\ \dot{\boldsymbol{\varphi}}_{\mathrm{tj}} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathcal{BI}}(\boldsymbol{\theta})(\mathbf{E}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}) \\ \mathbf{R}_{\mathcal{BI}}(\boldsymbol{\theta})\dot{\mathbf{r}}_{\mathrm{IB}} \\ \dot{\boldsymbol{\varphi}}_{\mathrm{tj}} \end{bmatrix} \tag{4-7}$$

For the derivation where the CMM-matrices are only dependent on the torso-joint trajectories $\mathbf{A}_{\mathrm{C}}(\boldsymbol{\varphi}_{\mathrm{tj}})$, the relation in Equation (4-7) satisfies the desired dynamics. Ultimately, the dynamic constraint $\boldsymbol{g}_{\mathrm{d}}(\mathbf{x}(t))$ can be described as:

Dynamic Constraint Abstraction for the Articulated Torso

$$\begin{aligned} &\boldsymbol{g}_{\mathrm{d}}(\mathbf{x}(t)) = \mathbf{A}\ddot{\mathbf{q}} + \dot{\mathbf{A}}\dot{\mathbf{q}} - \mathbf{w}_{\mathrm{G}} = 0 \\ &\quad \text{with:} \\ &\qquad \mathbf{A} = \bar{\mathbf{R}}(\boldsymbol{\theta})\mathbf{A}_C(\boldsymbol{\varphi}_{\mathrm{tj}}) \\ &\qquad \dot{\mathbf{A}} = \dot{\bar{\mathbf{R}}}(\boldsymbol{\theta}, \boldsymbol{\omega})\mathbf{A}_C(\dot{\boldsymbol{\varphi}}_{\mathrm{tj}}) + \bar{\mathbf{R}}(\boldsymbol{\theta})\dot{\mathbf{A}}_C(\dot{\boldsymbol{\varphi}}_{\mathrm{tj}}, \boldsymbol{\varphi}_{\mathrm{tj}}) \end{aligned} \tag{4-8}$$

4-2-3 Constraint jacobians

Furthermore, the optimization is dependent on the gradient of the objective function $\nabla f(x)$ and the Jacobian of the constraints $\nabla g(x) = \frac{\partial g(x)}{\partial x}$ with respect to the optimization variables $x(t)$. While most solvers can approximate all gradients and Jacobians numerically, most often the Jacobians are provided to speed up the optimization. Modifying the constraints of the TO algorithm also requires an updated derivation of the constraint Jacobians. Therefore, all constraints are revised based on the new decision variables $\boldsymbol{\varphi}_{\text{tj}}$ and the new constraints.

Chapter 5

Results

The generated trajectory of the articulated robot consists of the base-frame trajectory $\mathbf{r}_{\text{ib}}(t), \boldsymbol{\theta}_{\text{ib}}$, the torso-joint trajectories $\boldsymbol{\varphi}_{\text{tj}}(t)$, the contact-forces $\mathbf{f}_{\text{i}}(t)$ and the end-effector trajectories $\mathbf{p}_{\text{i}}(t)$. Each of these trajectories will be evaluated in the following section. The Trajectory Optimization (TO) can generate trajectories for different terrains, gait schemes, and agile maneuvers. However, while these trajectories might be exciting to explore, the trajectories presented in this report are generated using a flat terrain, a fixed end-time $T_{\text{final}} = 3.0\text{s}$, a straight initial trajectory ($r_{\text{start,y}} = r_{\text{final,y}}$) and a fixed gait-schedule.

Furthermore, while the Trajectory Optimizer for Walking Robots (TOWR) Nonlinear Programming (NLP) problem also allows gait-timing optimization, the gait patterns are fixed for the following discussion. Furthermore, the trajectories generated by the TO provide insight into how the integration of the torso joints affects the dynamics of the quadruped robot. However, to truly investigate the benefits that can be achieved with the integration of the spinal joints, it can be much more relevant to solve the NLP for rigid-torso quadruped robots as well. This allows for a much more meaningful differentiation between a rigid and articulated torso in terms of convergence, stability, and performance. Therefore, the trajectories presented in the following subsections are simulated with identical robots, where the torso is separated into 3 sections.

5-1 Experimental setup

The trajectory optimization algorithm is tested on an HP Zbook G3 running Ubuntu 20.04, with an Intel(R) Core(TM) i7-6700HQ@2.60GHz CPU and a 2.59 GHz processor. Furthermore, the complete GitLab repository can be found in [1], with the Unified Robotics Description Format (URDF) description containing all relevant kinematic and inertial properties of the robot used for the experiments.

Furthermore, the TO interface in Figure 5-1 shows the various optimization inputs the user can define by pressing a corresponding button on the keyboard. In addition to [13], a new option

[1] GitLab: https://gitlab.com/kockelkorenmichiel/compliant_spot.git

called "flexible torso" was added, which bounds the torso joints to their nominal position. The TO was tested for various terrains, gait sequences, and final distances and can be viewed here.[2]

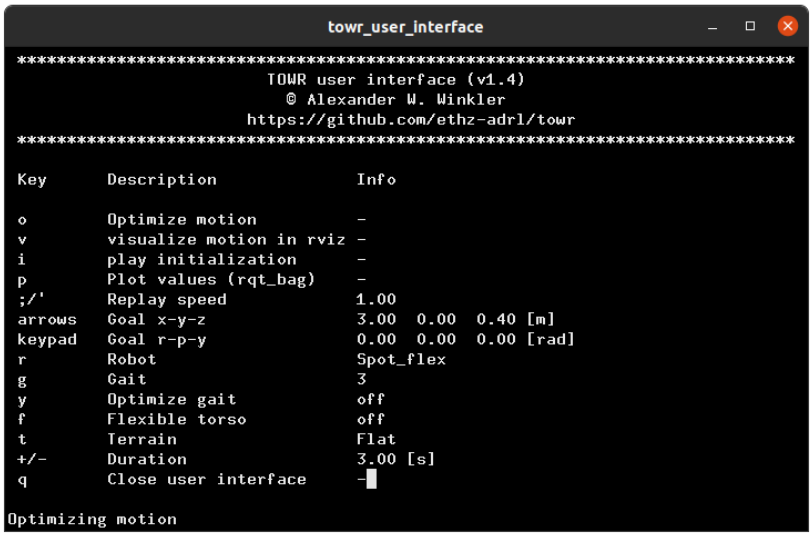


Figure 5-1: Towr user interface with the option to bound the torso angles by pressing the 'f'-button.

Finally, the solver used by the TO was the "MUMPS"-solver [28]. However, other nonlinear solvers that are considered to be faster are available. In general, the TO found a solution until four meters within a total computation time of 14 seconds for the articulated robot.

5-2 Trajectory feasibility

In the following section, the feasibility of the trajectory will be evaluated for varying distances and gait types. This feasibility is analyzed by counting the number of iterations needed to find a feasible solution. If no solution is found within 100 seconds, the optimization is terminated. Intuitively, the number of iterations needed to find a feasible trajectory could potentially suffer from the nonlinearity introduced by the torso joints. However, we can observe that the rigid torso's dynamic constraint only allows the base's actuation during the contact phase. In contrast, the velocity and acceleration of the torso joints can potentially generate momentum on the base frame during the flight phase.

In Figure 5-2, the number of iterations required to find a feasible trajectory is visualized for both the rigid torso and the articulated torso for identical initial conditions. As can be observed, the integration of the Degree of Freedom (DOF) in the torso allows the TO to find a feasible trajectory in much fewer iterations than for the rigid torso for all gait types and overall distances. Furthermore, while the TO algorithm cannot find a feasible trajectory for the rigid torso for a final distance of 5 seconds or higher, the TO even finds a feasible trajectory for the articulated torso for a final distance of 7 meters.

[2]Videos: https://www.youtube.com/playlist?list=PLmZ4ovuqQhMDMwtWewSW37cCfZYxINFxL

From Figure 5-2, it can be observed by the number of iterations, that the bound and gallop gait exploit the DOF of the torso-joints the most compared to the other gait schedules. This phenomenon can be expected from the gait schedules shown in Figure 2-1, front or hind legs make contact with the surface alternately. However, the pace, walk, and trot gait also seem to benefit from the DOFs in the torso in terms of feasibility.

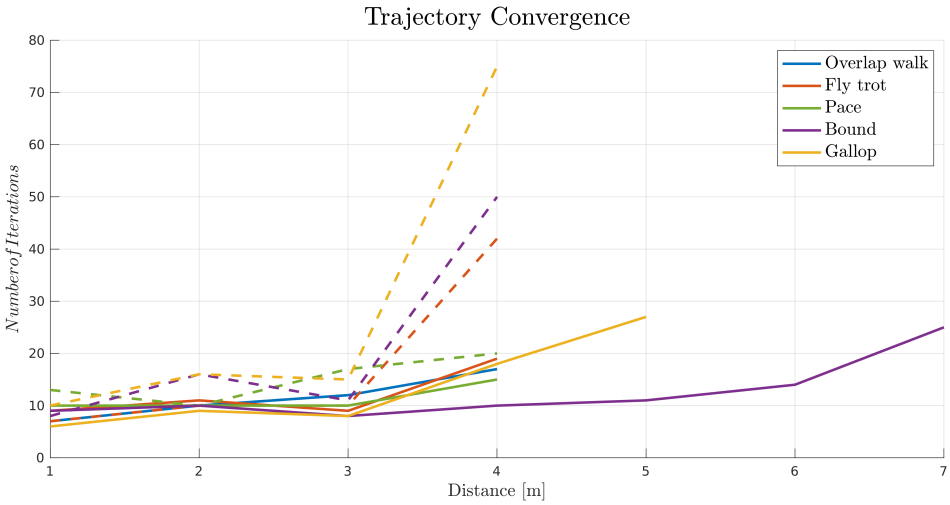


Figure 5-2: TO feasibility for rigid torso (dashed-line) and articulated torso (solid-line) over varying final distances $\mathbf{r}_{\text{final}}$, with $T_{\text{final}} = 3$ and varying gait types.

5-3 Example trajectory

An example trajectory will be discussed as an indication of how the algorithm performs and how the articulation of the torso affects the performance. For this analysis, only a straight-line trajectory was selected. As the bound gait seems to benefit the most from the torso joints, the example trajectory is generated with the bound gait. Furthermore, to allow comparison to the rigid torso with identical initial conditions, a final distance is selected that is feasible for both systems ($\mathbf{r}_{\text{final,x}} = 2.5$). In Section 2-4, it was discussed how the articulation of the torso helps the torso to increase stride length and reduce contact forces to reach a higher maximum velocity. Therefore, we will analyze the base trajectory, a single end-effector trajectory, contact forces, and the torso joint trajectories of the articulated torso. The trajectories can be visualized in the data visualization tool Rviz[29], shown in Figure 5-3.

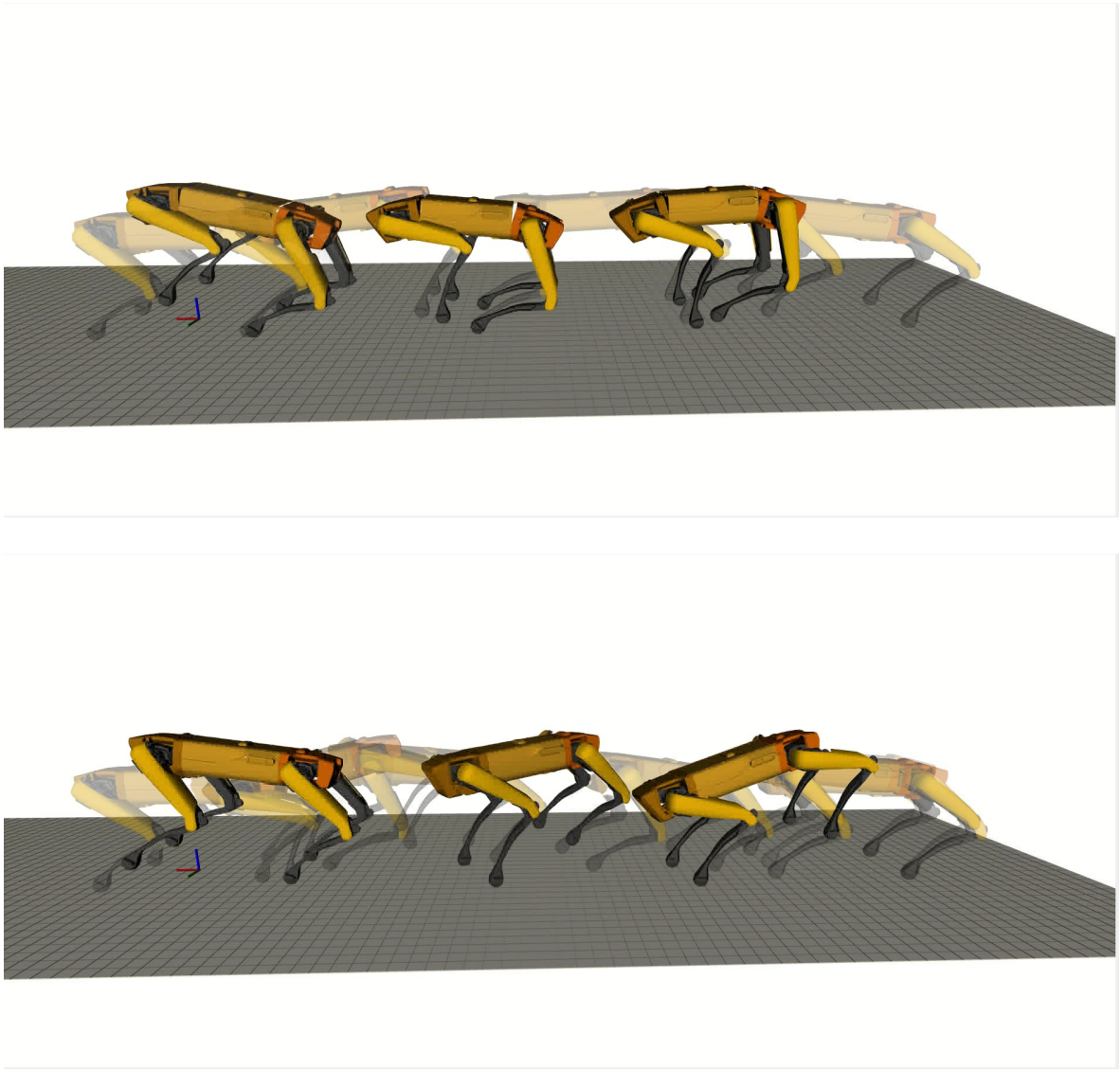


Figure 5-3: Trajectory visualized in Rviz, with an enabled (top) and disabled (bottom) torso articulation.

5-3-1 Base trajectory

The linear base position is provided in Figure 5-4. First, we can observe that the trajectories in the horizontal directions are much smoother for the articulated torso than for the rigid torso. Furthermore, the rigid robot seems to move laterally as well.

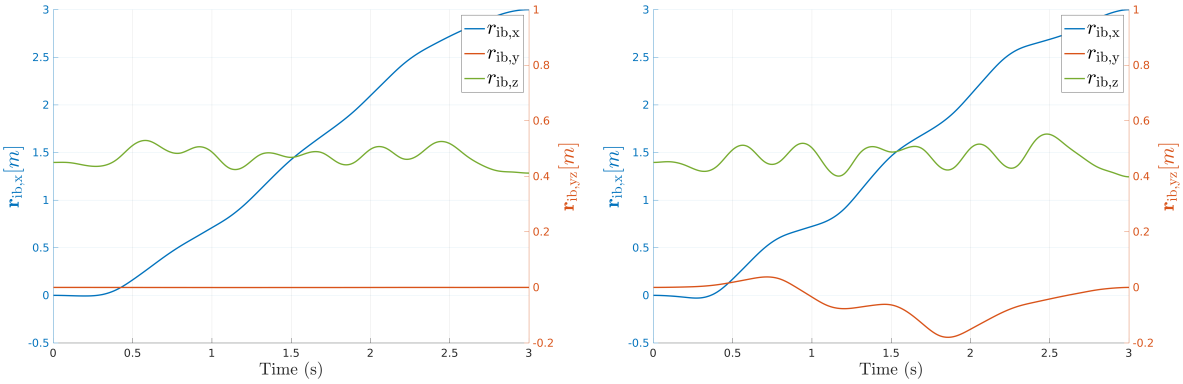


Figure 5-4: Linear base position $\mathbf{r_{IB}}(t)$, with $\mathbf{r}_{yz}$ plotted on the left right vertical axis, a) Enabled torso articulation, b) Disabled torso articulation.

As it is commonly argued that the articulated torso results in higher maximum velocities, the articulated base was initially expected to reach a higher forward velocity. However, the linear velocity is visualized in Figure 5-5 and shows that the articulated torso reaches a lower maximum velocity. Nevertheless, the articulated torso moves in a more streamlined forward motion, with a more constant velocity. The velocity of the rigid torso has much higher fluctuations in all directions.

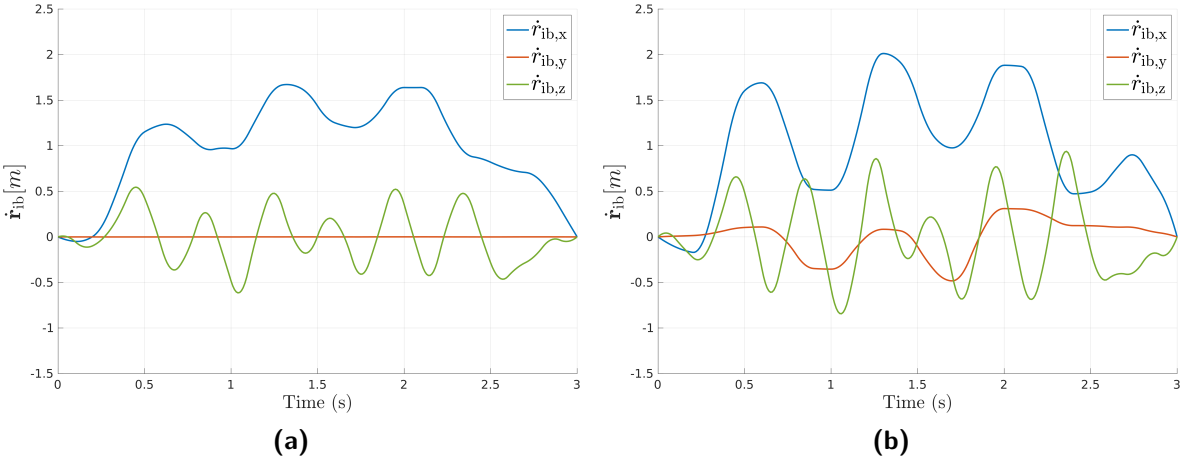
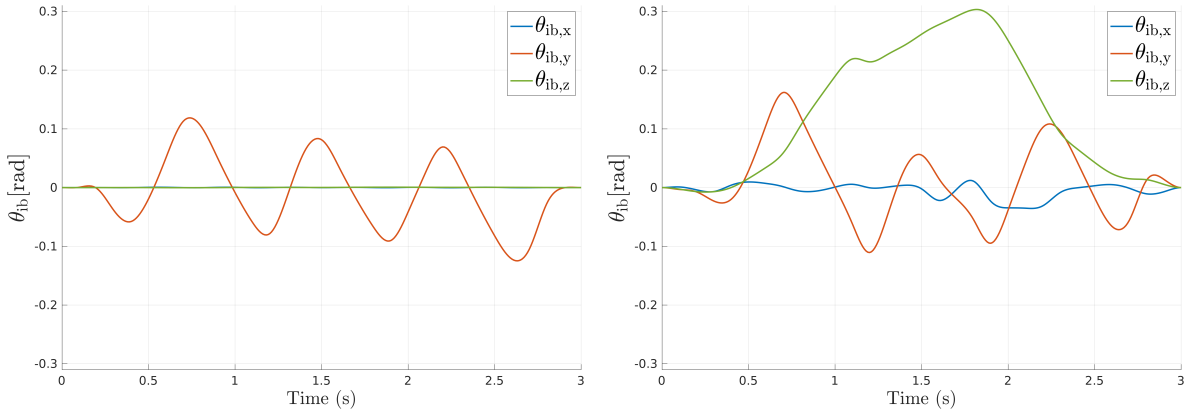


Figure 5-5: Linear base velocity $\dot{\mathbf{r}}_{\mathbf{IB}}(t)$ a) Enabled torso articulation, b) Disabled torso articulation.

This lateral movement of the rigid system can also be observed in Figure 5-5, where a strong yaw rotation (around z-axis) can be seen. Furthermore, most interestingly, a strongly reduced base rotation can be observed for the articulated system. Here, only a pitch rotation (around y-axis) is observed, and all other rotations are negligible. Also, the pitch rotation of the articulated base is reduced in comparison with the rigid system.

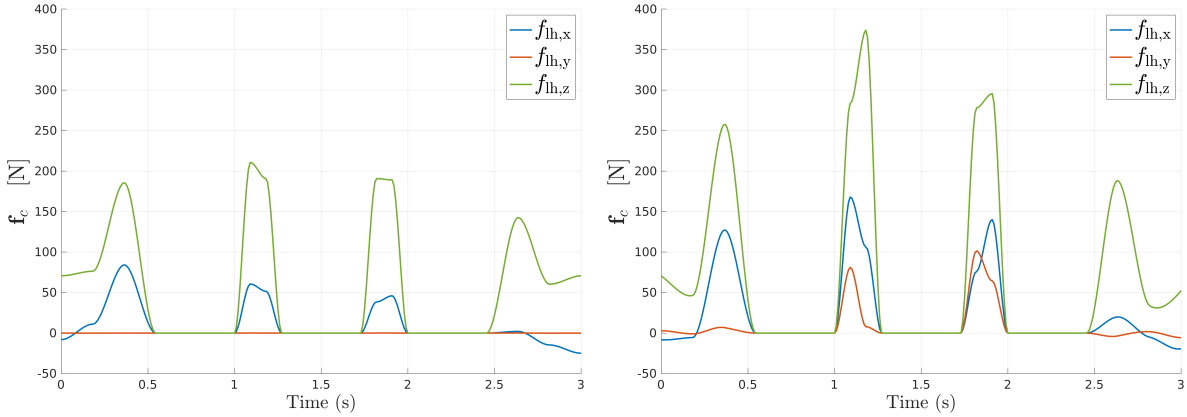


(a) **(b)**

Figure 5-6: Angular base orientation $\boldsymbol{\theta}_{\mathbf{IB}}(t)$ a) Enabled torso articulation, b) Disabled torso articulation.

5-3-2 Foot forces and position

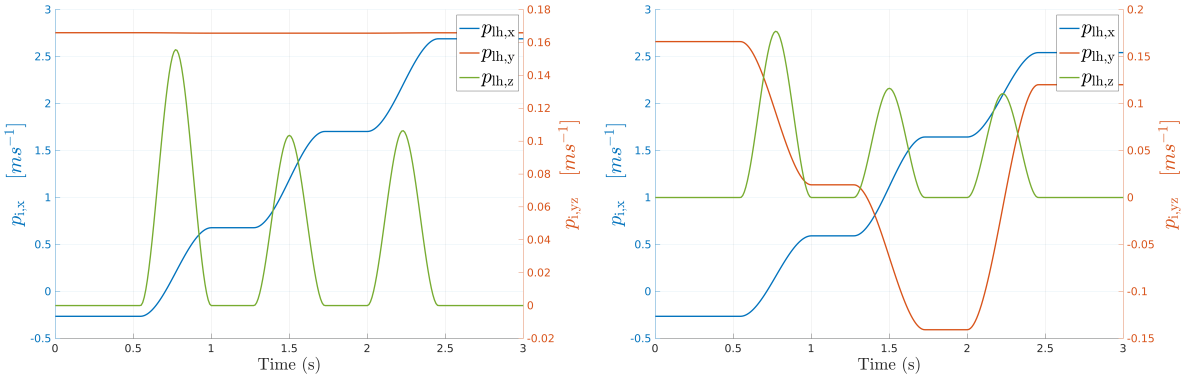
In several sources discussed in Section 2-4, it was argued that the articulation of the base allows a reduction in contact forces. This is considered beneficial for energy efficiency as the contact force is related to the torque required to exert the force on the ground. For the motion computed in this experiment, it can indeed be observed that the contact forces are drastically reduced when the trajectory is computed with an articulated base.



(a) **(b)**

Figure 5-7: Left-Hind (LH) contact forces $\mathbf{f}_{\text{c,lh}}(t)$ a) Enabled torso articulation, b) Disabled torso articulation.

Furthermore, in the literature discussed in Section 2-4, it is also discussed that the torso articulation allows a longer stride length. However, from this experiment, it is difficult to see whether the stride length is greater. Measuring the horizontal displacement in Figure 5-8 gives no clarity as the step distance varies each step.

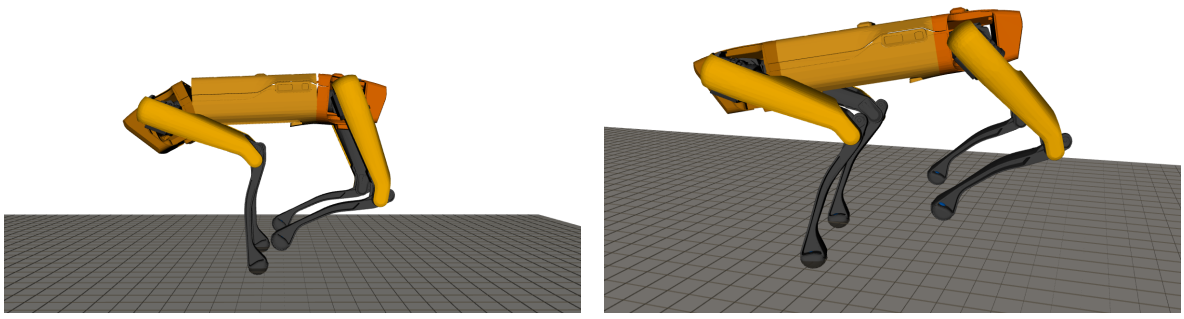


(a) **(b)**

Figure 5-8: LH foot position $\mathbf{r}_{\text{lh}}(t)$ a) Enabled torso articulation, b) Disabled torso articulation.

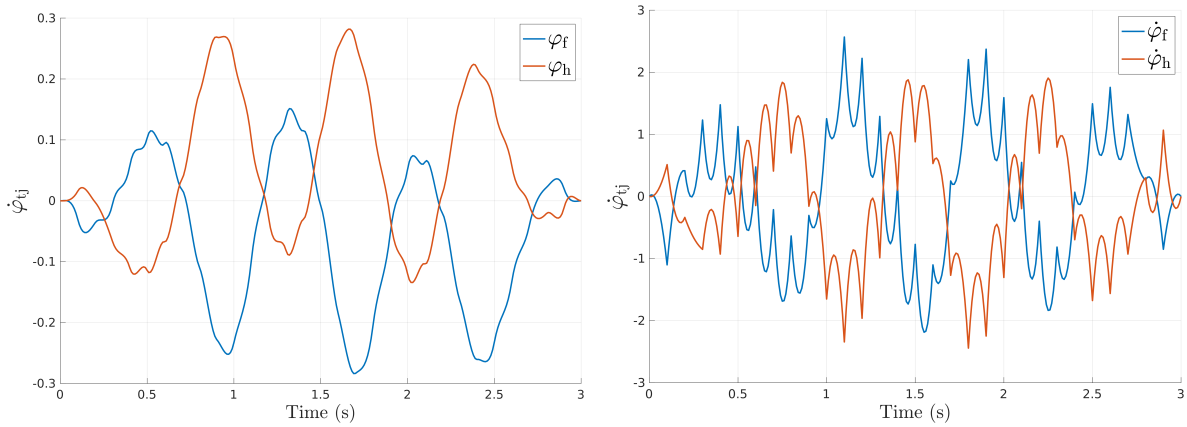
5-3-3 Torso joint trajectories

The torso joint trajectories in Figure 5-10 provide an interesting insight into the rotation of the torso joints. It can be seen that the torso joints move in opposite directions, similar to the bounding motion of the cheetah in Figure 2-4. Moreover, if we look at the robot trajectory at an arbitrary timestamp, we can see that the optimized motion is very similar to the expected motion. For example, in Figure 5-9 we can see that the torso flexes to allow the middle and hind sections of the torso to rotate. This gives the hind legs the ability to place their feet more towards the front legs. In contrast, we can see that the rigid torso cannot rotate its torso in Figure 5-9b.



(a) **(b)**

Figure 5-9: Frame of robot trajectory at timestamp $t = 0.9s$, a) Enabled torso articulation, b) Disabled torso articulation.



(a) **(b)**

Figure 5-10: Torso joint trajectories a) Angular orientation $\boldsymbol{\varphi}_{\mathrm{tj}}$, b) Angular velocity $\dot{\boldsymbol{\varphi}}_{\mathrm{tj}}$.

Chapter 6

Validation

To validate whether the generated trajectory would be feasible in real-world scenarios, there are several options to explore. The most straightforward option would be to test the trajectory on a real system or in a simulation environment. This method is very effective and provides many insights into the dynamic feasibility of the generated trajectory. However, testing in real-world scenarios requires the availability of an articulated robot or a simulation model, and an advanced motion controller that is capable of tracking the motion. The development of an advanced motion control scheme for an agile articulated robot would be a very interesting research topic in itself. While an effort was made to design such a control scheme, not enough time was available to complete it.

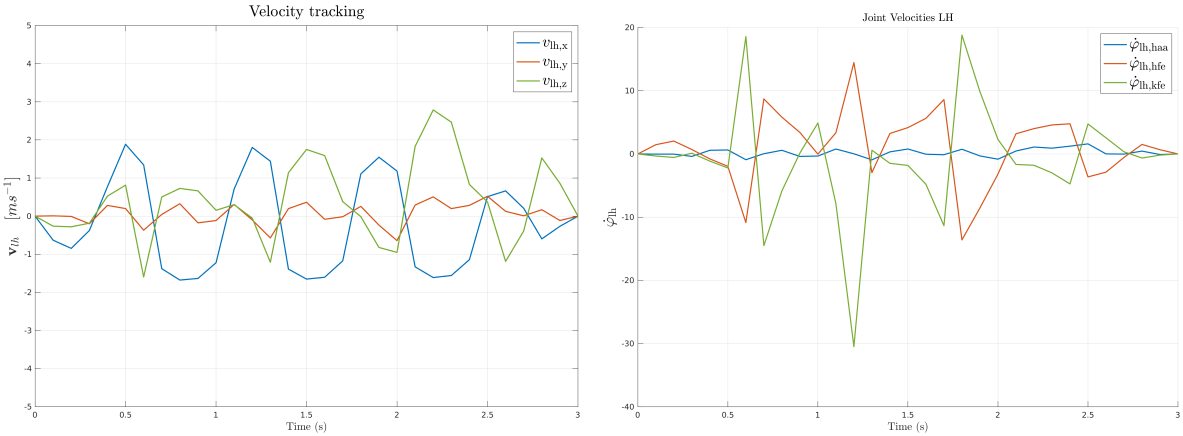
However, the availability of open-source computational tools like the Pinocchio library [30] allows a convenient comparison of the analytic derivation and the automated numeric derivation. The Pinocchio library is available in both C++ and Python and can derive various kinematic and dynamic relationships. The library computes these relations directly using the kinematic and inertial properties provided in a Unified Robotics Description Format (URDF) file. As the Triple Rigid Body Dynamics (TRBD) model is an abstraction of the Centroidal Dynamics (CD) model, the most important functionality for this study, is the capability of deriving the CD numerically. This not only allows us to validate the correctness of the analytic derivation but also allows us to simulate the trajectory for both the CD model and the TRBD model.

However, before the Pinocchio library can be used to compute the CD, the joint-space inputs for the CD model need to be derived from the Cartesian end-effector Trajectory Optimization (TO) output. This requires the use of Inverse Kinematics (IK), which must also be validated to ensure no singularities are present.

6-1 Kinematic model validation

As discussed in Section 3-3-1, using the end-effector Jacobian is a common way to transform end-effector velocities and accelerations. However, this method can suffer from kinematic

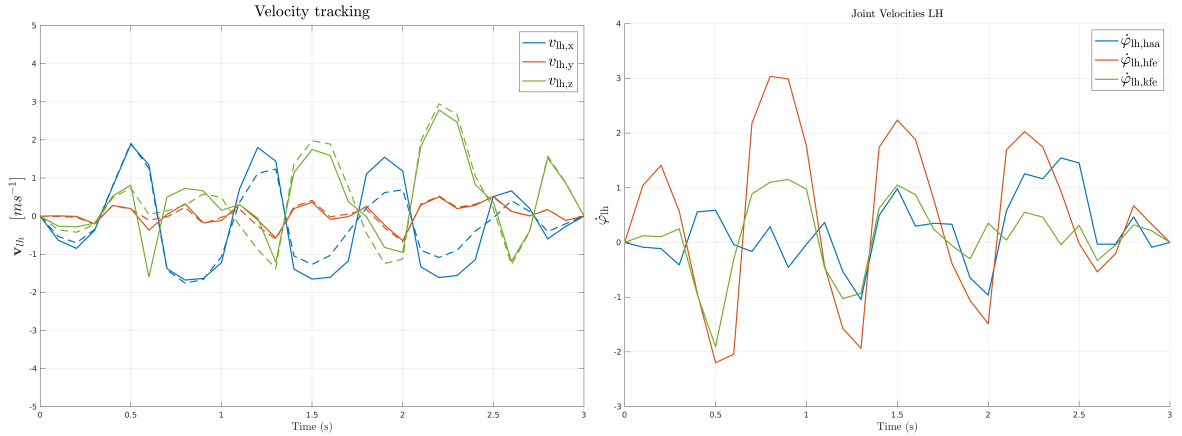
singularities near singular configurations.



(a) **(b)**

Figure 6-1: Direct inverse solution for LH-foot $\mathbf{J}^{-1}\mathbf{v}$ a) Cartesian velocity tracking, b) Joint velocities.

In Figure 6-1, the end-effector velocity of the left-hind foot is plotted as well as the joint velocities of this foot computed by the inverse of the Jacobian. As can be observed in Figure 6-1b, the joint velocities exhibit high peak velocities. However, the end-effector velocity given by these joint velocities tracks the desired velocity perfectly.



(a) **(b)**

Figure 6-2: Damped Least Squares solution for LH-foot a) Cartesian velocity tracking, b) Joint velocities.

From Figure 6-1, it can be observed that this particular trajectory suffers from near-singular configurations. This limits the ability to use inverse IK to transform the end-effector trajectories to joint-space trajectories. Therefore, the IK used in Figure 6-2, uses the damped least-squares approach in combination with a filter on the singular velocity provided in Equation (A-14). As shown in Figure 6-2, the peak joint velocities are smoother than the direct inverse solution in Figure 6-1. As expected, the elimination of the peaks caused by the singular joint configuration comes at the expense of the velocity tracking.

6-2 Pinocchio centroidal dynamics correction

When using a numeric library as a measure of correctness, it is advisable to be aware of potential computational errors in the toolbox. A miscalculation in both derivations can cause a discrepancy in output. Chapter 3 thoroughly elaborates on the derivation of the different dynamic models. This is incredibly helpful to fully understand the model and trace back any potential modeling errors.

When checking the first-order dynamics, the Centroidal Momentum Matrix (CMM) matrix computed by the Pinocchio library was identical to the matrix computed analytically. However, when comparing the second-order dynamics output, a discrepancy was observed in the time-derivative of the CMM matrix. Two observations were essential to the identification of the origin of the discrepancy. First, it was observed that the CMM output had the following structure:

$$\mathbf{A}_{\mathrm{G}}(\boldsymbol{\varphi}_{\mathrm{tj}}) = \begin{bmatrix} \mathbf{A}_{\mathrm{G11}} & \mathbf{0}_{3\times3} & \mathbf{A}_{\mathrm{G13}} \\ \mathbf{A}_{\mathrm{G21}} & \mathbf{A}_{\mathrm{G22}} & \mathbf{A}_{\mathrm{G23}} \end{bmatrix}, \tag{6-1}$$

with a block matrix reduced to zeros. Intuitively, the time-derivative of such a matrix would provide a similar structure, with $\dot{\mathbf{A}}_{\mathrm{C},12} = \mathbf{0}$. However, the matrix computed by the Pinocchio library contained only non-zero values. This observation led to the belief that the second-order centroidal dynamics computed by the Pinocchio library contained a miscalculation.

Furthermore, the second observation was that the angular half $\dot{\mathbf{A}}_{\mathrm{G},2}$, mapping the joint rates to the angular momentum, was identical in both derivations. These two observations lead to the belief that the discrepancy was caused by the transformation to the centroidal frame ${}^{\mathrm{G}}\mathbf{X}^*$ and especially its time-derivative ${}^{\mathrm{G}}\dot{\mathbf{X}}^*$. By analyzing the source code of the Pinocchio library, it was discovered that the CMM matrix was computed by deriving the momentum matrix with respect to the Inertial frame $\mathbf{A}_{\mathrm{I}}$ and using a transformation to the Centroidal frame:

$$\mathbf{A}_G = {}^{G}\mathbf{X}_{\mathrm{I}}^*\mathbf{A}_{\mathrm{I}} = \begin{bmatrix} \mathbb{I}_{3\times3} & \mathbf{S}(\mathbf{r}_{IG})^\intercal \\ \mathbf{0} & \mathbb{I}_{3\times3} \end{bmatrix} \mathbf{A}_{\mathrm{I}} \tag{6-2}$$

The time-derivative of the CMM, in this case, can easily be derived using the chain rule.

$$\dot{\mathbf{A}}_{\mathrm{C}} = {}^{\mathrm{G}}\dot{\mathbf{X}}_{\mathrm{I}}^*\mathbf{A}_{\mathrm{I}} + {}^{\mathrm{G}}\mathbf{X}_{\mathrm{I}}^*\dot{\mathbf{A}}_{\mathrm{I}} \tag{6-3}$$

However, when looking at the implementation in the Pinocchio Library, it was discovered that the chain rule was not applied properly, leading to a mismatch in results.

$$\dot{\mathbf{A}}_G \neq {}^{\mathrm{G}}\mathbf{X}_{\mathrm{I}}^*\dot{\mathbf{A}}_{\mathrm{I}} \tag{6-4}$$

This reveals the origin of the discrepancies and all observations can be traced back to this error. Fortunately, after raising this issue with the main developer of the library, the computational error was resolved in the new Pinocchio release (2.6.21).

6-3 Dynamic model validation

As the analytically derived CD is rather complex and prone to modeling errors it is advisable to find a way to verify the dynamic representation. Fortunately, the Pinocchio library [30] allows

the computation of various kinematic and dynamic properties by providing the kinematic relations of the robot in the form of a URDF file. One of its features is that it can also compute the centroidal properties. By comparing the analytically derived output to the Centroidal properties computed by Pinocchio, we can get insights into the validity of the analytic derivation of the CD model.

The CMM matrix is an excellent identifier of potential modeling errors in the first-order Centroidal dynamics, as it is only dependent on the generalized state vector $\mathbf{q}$. Furthermore, when the first-order dynamics seem to contain no modeling errors, the time-derivative of the CMM can be compared to identify modeling errors in the second-order Centroidal dynamics. However, as we are primarily interested in the correctness of the TRBD model, the CD model needs to be converted to the TRBD model. This allows us to analyze the correctness of the analytic description of the TRBD model.

Besides validating the correctness of the analytic description, we are also interested in the accuracy of the proposed dynamic representation. This provides some insight into the effect of the model abstractions on the model accuracy. One simple way to do this is to compare the TRBD output to the CD output in terms of the centroidal momentum $\mathbf{h}_{\mathrm{G}}$ and its time-derivative $\dot{\mathbf{h}}_{\mathrm{G}}$.

6-3-1 Model correctness

After the computational error in the Pinocchio library was resolved, the correctness of the analytical derivation was validated. First, the numerically computed Centroidal properties were derived, and a numeric TRBD model was obtained by fixing the leg-joint position to the nominal leg-joint positions $\boldsymbol{\varphi}_{\mathrm{lj}} = \boldsymbol{\varphi}_{\mathrm{lj,nom}}$ and setting the leg-joint velocities to zero $\dot{\boldsymbol{\varphi}}_{\mathrm{lj}} = \mathbf{0}$. A schematic visualization of the proposed validation scheme can be observed in Figure 6-3.

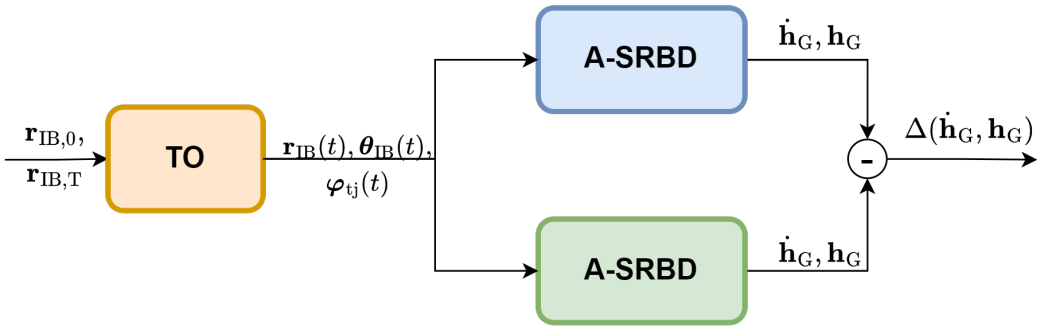


Figure 6-3: Model correctness analysis of analytic TRBD (blue) and the numeric TRBD (green).

It was then observed that the output computed by Pinocchio was identical to the analytically derived output, in terms of the CMM matrices $\mathbf{A}_{\mathrm{G}}, \dot{\mathbf{A}}_{\mathrm{G}}$ and the centroidal momentum vectors $\mathbf{h}_{\mathrm{G}}, \dot{\mathbf{h}}_{\mathrm{G}}$. Therefore, it can be concluded that the analytic derivation of the TRBD is correct.

6-3-2 Model accuracy

As discussed throughout this research, the accuracy of the dynamic representation is of vital importance for the feasibility of the generated trajectory. Therefore, especially when using

a simplified representation of the system dynamics, it can be incredibly useful to investigate the effects of the model abstractions on the model accuracy. Typically, the behavior of the derived model is compared to the behavior of the real system or the behavior of a dynamic model that is known to be more accurate. As the TRBD model is an abstraction of the CD model, it seems straightforward to compare the behavior of the TRBD model to the CD model.

The most straightforward way to investigate the accuracy of the TRBD representation compared to the CD model, is by comparing the centroidal momentum $\mathbf{h}_{\mathrm{G}}$ and its time-derivative $\dot{\mathbf{h}}_{\mathrm{G}}$ for a predefined set of generalized states and velocities. However, as the centroidal momentum and its time-derivative are typically difficult to interpret, the base acceleration $\dot{\mathbf{r}}_{\mathrm{IB}}, \dot{\boldsymbol{\omega}}_{\mathrm{IB}}$ can provide a much more convenient interpretation of the model accuracy. Given that the IK can convert the end-effector trajectories to feasible joint-space trajectories, the TRBD and the CD model can be inverted to find the base accelerations, as shown in Equation (6-5).

$$\ddot{\mathbf{q}}_{\mathrm{b}} = \mathbf{A}_b^{-1}\left[\mathbf{w}_G - \dot{\mathbf{A}}\dot{\mathbf{q}}_{\mathrm{b}} - \dot{\mathbf{A}}_{\varphi}\dot{\boldsymbol{\varphi}} - \mathbf{A}_{\varphi}\ddot{\boldsymbol{\varphi}}\right] \tag{6-5}$$

Here, $\mathbf{A}_{\mathrm{b}} \in \mathbb{R}^{6\times 6}$ are the six left columns of $\mathbf{A}_{\mathrm{G}}$ and $\mathbf{A}_{\mathrm{varphi}} \in \mathbb{R}^{6\times n_j}$ are the n_j right columns of $\mathbf{A}_{\mathrm{G}}$, $\mathbf{w}_{\mathrm{G}} \in \mathbb{R}^6$ is the wrench exerted on the centroid. By using the output of the TO, the approach allows us to compare the base accelerations for realistic behaviors. A schematic visualization of the proposed approach is provided in Figure 6-4.

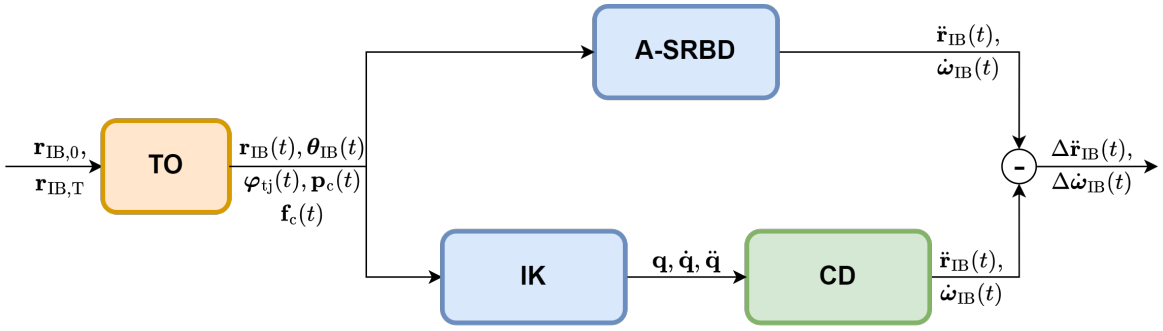
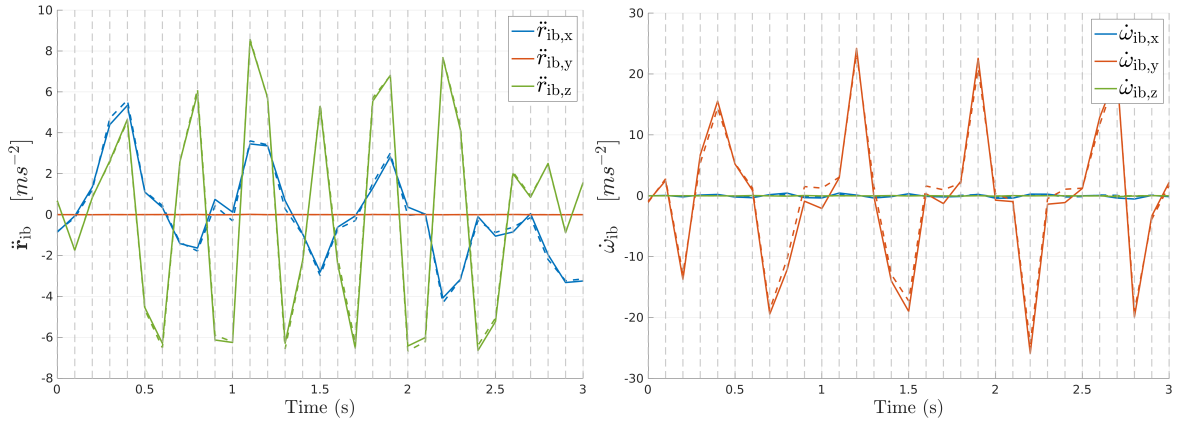


Figure 6-4: Model accuracy analysis of TRBD compared to the CD model.

The trajectory computed by the TO is first converted to joint space trajectories using the IK relations discussed in Section 3-3-2. This approach allows us to observe the model's accuracy for realistic behaviors computed by the TO. It can be observed in Figure 6-5, that the TRBD is able to represent the dynamics of the CD accurately. Only slight deviations are noticeable, especially with the angular acceleration $\omega_{\mathrm{IB,y}}$. The origin of these deviations can easily be traced back to the peak joint accelerations and velocities of the leg joints shown in Figure 6-6. These angular accelerations create an additional moment on the base, that is neglected in the TRBD model. Still, with quite significant joint velocities and accelerations, the TRBD is still able to model the dynamics in a lower-dimensional representation.

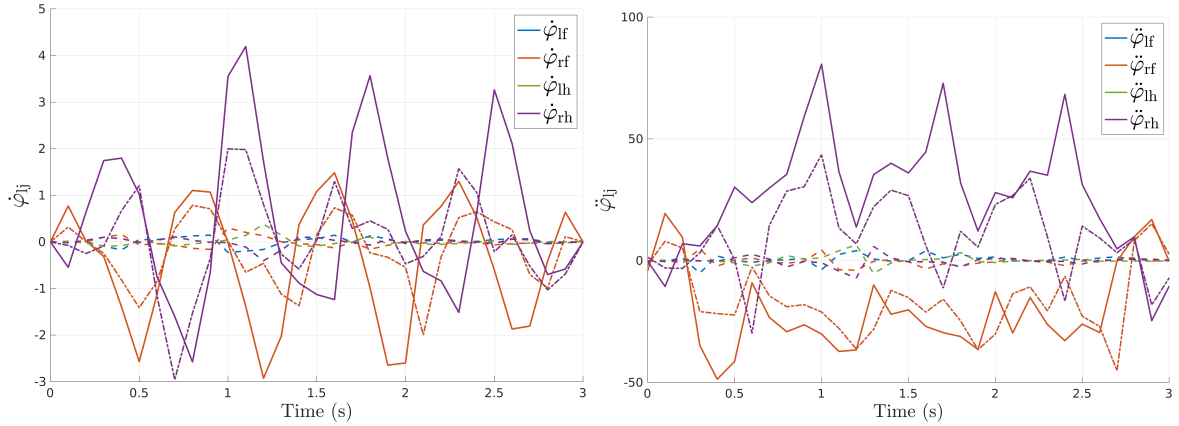
6-4 Trajectory dynamic consistency

The observation that the TRBD is able to provide an accurate representation of the system dynamics is no guarantee that the generated trajectory is actually feasible. As discussed in



(a) **(b)**

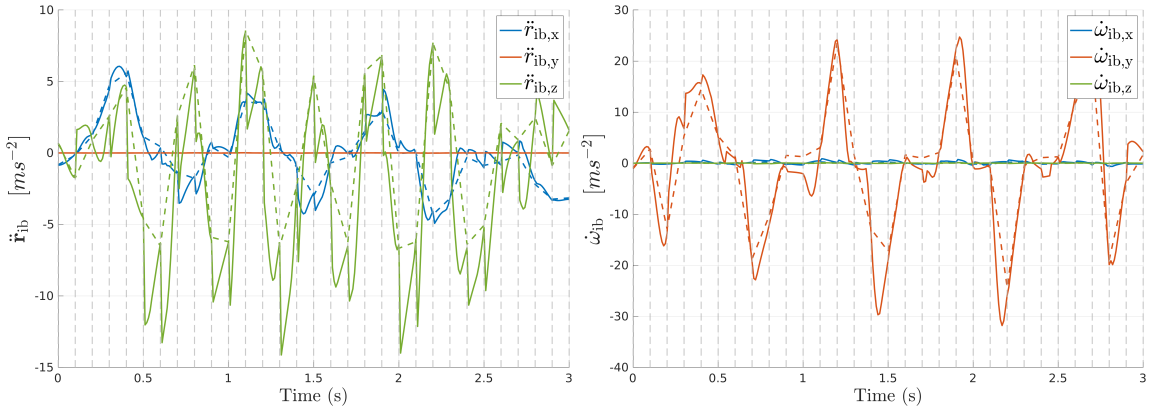
Figure 6-5: Comparison between the CD (solid) and TRBD (dashed) base acceleration $\ddot{\mathbf{q}}_b$ at the collocations points, a) Linear acceleration $\dot{\mathbf{r}}_{IB}$, b) Angular acceleration $\dot{\boldsymbol{\omega}}_{IB}$.



(a) **(b)**

Figure 6-6: Leg joint velocities and accelerations derived using IK for the Hip Abduction-Adduction (HAA) (dashed), HIP Flexion-Extension (HFE) (solid), Knee Flexion-Extension (KFE) (dashed-dotted) a) Joint velocities $\dot{\mathbf{q}}_{lj}$, b) Joint accelerations $\ddot{\mathbf{q}}_{lj}$

Section 2-5-1, the system dynamics are only enforced at the collocation points. Therefore, before the TO can be used, it is important to quantify to what extent the generated trajectory is consistent. The previous analysis in Figure 6-4, can be used for this analysis as well. Instead of only observing the acceleration at the collocation points, we forward simulate the base acceleration using the IK and CD model for the intermediate trajectories between the collocation points. The resulting base acceleration is plotted in Figure 6-7. As was also observed in the accuracy analysis, the acceleration values coincide exactly at the collocation times. The acceleration profile in Figure 6-7, show that with the exception of some peak deviations, the system dynamics are represented in an accurate way. In [13], a similar dynamic consistency analysis was performed. Here, similar deviations were observed between the collocation points. It was argued that a more accurate resemblance to the actual dynamics can be obtained by increasing the number of collocation points. However, the motion generated by the planners cannot be executed perfectly anyway due to sensor noise, inaccurate force



(a) **(b)**

Figure 6-7: Comparison between the CD (solid) and TO (dashed) base acceleration $\ddot{\mathbf{q}}_b$. a) Linear acceleration $\dot{\mathbf{r}}_{IB}$, b) Angular acceleration $\dot{\boldsymbol{\omega}}_{IB}$.

tracking, and delays. Therefore, slight misrepresentations of the system dynamics can be counteracted by a motion-tracking controller.

Chapter 7

Discussion

7-1 Observations

The main limitation observed in literature in the context of articulated-legged robotics is the lack of efficient motion planners that fully leverage the articulation of the torso. The motion optimization schemes proposed for articulated robots in recent literature generally limited their optimization to planar Rigid Body Dynamics (RBD) models. These studies only allowed a bound gait in a two-dimensional simulation. In contrast, advanced motion optimization algorithms are typically proposed for ordinary robots, and use low-dimensional models to speed up the optimization. However, it was observed that no three-dimensional model that was suitable for fast motion optimization had yet been derived.

The conventional models were either too complex in terms of nonlinearity and number of dimensions, or could not leverage the articulation of the base. Therefore, the main research objective of this study was to derive a low-dimensional dynamic model that matched models used in ordinary motion optimization algorithms in terms of model complexity and dimensionality. The proposed dynamic representation ultimately represents an excellent trade-off between model dimensionality, complexity, and model accuracy.

The derivation of the model is relatively straightforward and proves to be convenient to implement in a standard legged-robot Trajectory Optimization (TO) framework. For example, each element of the momentum matrices is either zero, constant, or consists of a short nonlinear expression. Without any prior experience in C++, the TO algorithm proposed in [13] was fully adapted to the new decision variables and constraints in a few months. Furthermore, the symbolic expressions for the centroidal momentum matrices are all automated in a Matlab algorithm. This algorithm prints these matrices in a C++ format that can be pasted directly into the TO framework. These matrices include the centroidal momentum matrices and their constraint Jacobians. This allows for a convenient integration of different articulated robots.

Furthermore, the validation proves that the representation of the base dynamics is accurate. In addition, the trajectory also seems to follow the system's dynamics closely, when simulating the trajectory on the full centroidal model. Although some deviations are observed in the

acceleration between the collocation nodes, it is argued in [13] that these deviations can typically be counteracted using a tracking controller. Unfortunately, this assumption was not validated for this TO algorithm but can be an interesting topic for future studies.

Finally, the generated trajectories computed by the adapted TO algorithm seem consistent with the observations in previous studies on quadruped robots with flexible spines. The torso articulation is most apparent, especially in challenging test cases requiring fast agile movement. The base trajectory seems stable and the contact forces are significantly reduced compared to the TO solution with fixed torso joints.

7-2 Considerations

Performance

It is important to consider that no hard conclusions on the difference in performance can be made solely based on these trajectories. This is due to the use of abstractions of the system dynamics. The rigid motion optimization might suffer from its sole dependence on the contact forces. In contrast, the articulated optimization benefits from the additional inputs the torso joints provide. A more complete comparison of potential performance can be realized with a full centroidal model for both systems and the inclusion of an objective function. However, comparing both systems was not the aim of this project and has been examined extensively in related studies. However, we can argue that the trajectory computation benefits from the Degrees of Freedom imposed by the torso joints.

Alternative joint configurations

The application of the proposed modeling framework is not limited to quadruped systems with two parallel torso joints. Hence, it can be used to create a dynamic abstraction for any articulated floating base system. For example, for systems with only a single revolute torso joint, as shown in Figure 7-1a, the middle segment can be interpreted as a virtual body without any mass or inertia. The pitch rotation of the virtual middle segment stays aligned with the inertial frame. The joint rotation of each torso segment is then provided with respect to the inertial frame. For revolute joints in other directions, the spatial transformation matrices and the spatial Jacobian matrices can be adapted conveniently. This approach can even provide a low-dimensional representation for systems with one or more prismatic torso joints as shown in Figure 7-1b.

Mass distribution

It is important to consider some design factors that influence the accuracy of the TRBD model and the achievable TO performance. For example, the mass distribution significantly impacts the model's accuracy and the robot's potential performance. Regarding performance, it is discussed in [31] that the robot's performance increases when the legs' mass is reduced. However, regarding model accuracy, a high torso/legs mass distribution is also beneficial for both the Single Rigid Body Dynamics (SRBD) and the TRBD models. The assumption that

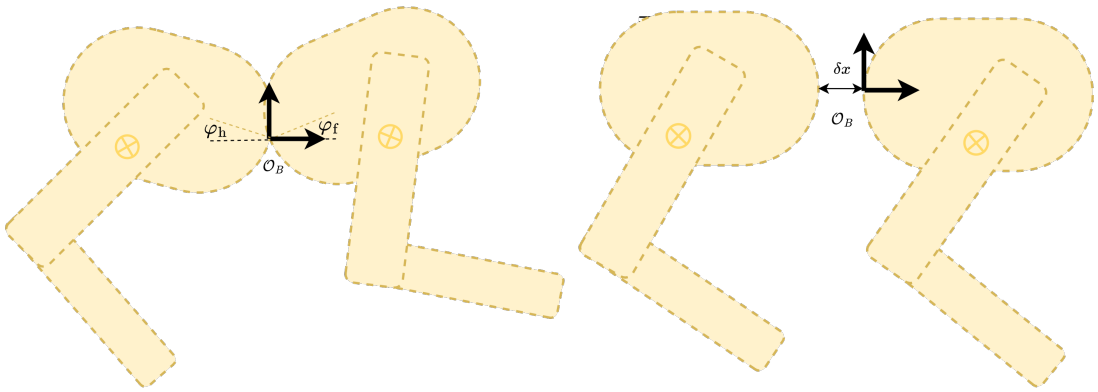


Figure 7-1: Visualized Triple Rigid Body Dynamics (TRBD) abstraction for different articulated quadruped robots, a) Single revolute joint, b) Single prismatic joint.

the momentum created by the leg joints can be neglected typically fails when a robot has legs with high inertial properties compared to the base. Therefore, the performance observed in Chapter 5 and the good tracking in Chapter 6 can be degraded when applied to other robotic designs.

Torso stability

One major factor that becomes apparent when observing the base trajectories is that the trajectory of the flexible torso seems much more stable than the trajectory of the rigid torso. One possible explanation could be that the torso joints can actuate the floating base during the flight phase. In contrast, the rigid model depends on short contact periods to actuate the base. This could explain the difference in the magnitude of the contact forces. The contact forces must be planned carefully for these short periods to maintain balance. In contrast, the articulated model is less dependent on the contact forces and can adjust the acceleration of the base by accelerating its torso joints. This could be why it generally takes longer for the TO to find a solution for the rigid model than for the articulated design.

Experiment settings

In Chapter 5, the trajectory is visualized for a flat terrain without the need for any lateral movement. Nevertheless, the TO allows experimentation on more challenging terrains like a gap, stairs, and an elevated platform. Therefore, various other trajectories are visualized in [32] and show that the algorithm can compute feasible trajectories on challenging terrain. Furthermore, although the TO allows the optimization of the step durations, the trajectories presented in Chapter 5 are optimized with a fixed bound gait schedule.

7-3 Limitations and future work

First of all, the Rviz trajectory visualization in Figure 7-2 proves that the proposed Range of Motion (ROM) indeed rotates with respect to the orientation of the torso segments. However, for some trajectories, a critical limitation of the kinematic constraint is discovered. As shown in Figure 7-2, the flexion of the torso may cause the front and hind legs to intersect. This limitation generally only occurs with the articulated design and needs to be averted by excluding some region of the ROM defined earlier.

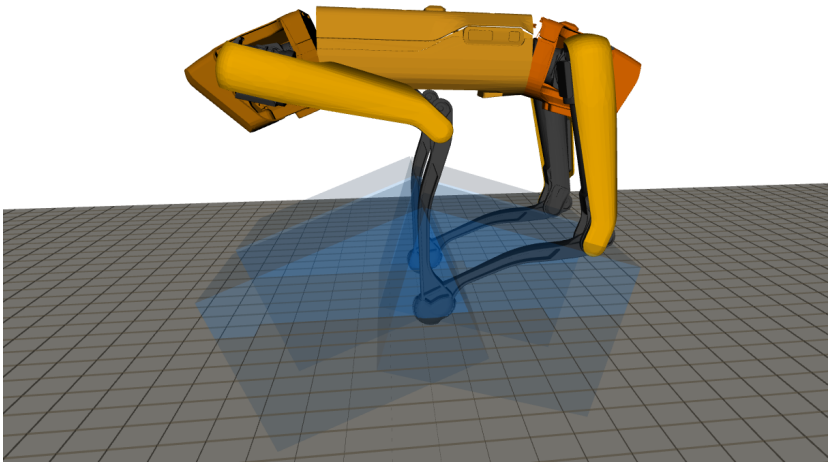


Figure 7-2: Kinematic constraint limitation.

Furthermore, the dynamic constraint is enforced only at the collocation points and deviates slightly between the intervals. To verify that the trajectory can still be tracked in real-time experiments, a feedback controller that tracks all trajectories effectively is required. The tracking ability can be investigated by tracking the motion in a real-world or simulated environment. Therefore, I set up a simulation environment in Gazebo [33], with the articulated Unified Robotics Description Format (URDF) model. Furthermore, a main control structure was implemented using Range of Motion (ROS) [34] in C++, which processes sensor information, processes desired trajectories computed by the TO using Inverse Kinematics (IK). Unfortunately, the control algorithm that computes the required joint torques to track the desired motion was not finished before the end of this research. In addition, for direct implementation of the TO, it would be desirable if the TO would use the current states of the system from the robot as initial states visualized by the red-dotted line in Figure 7-3.

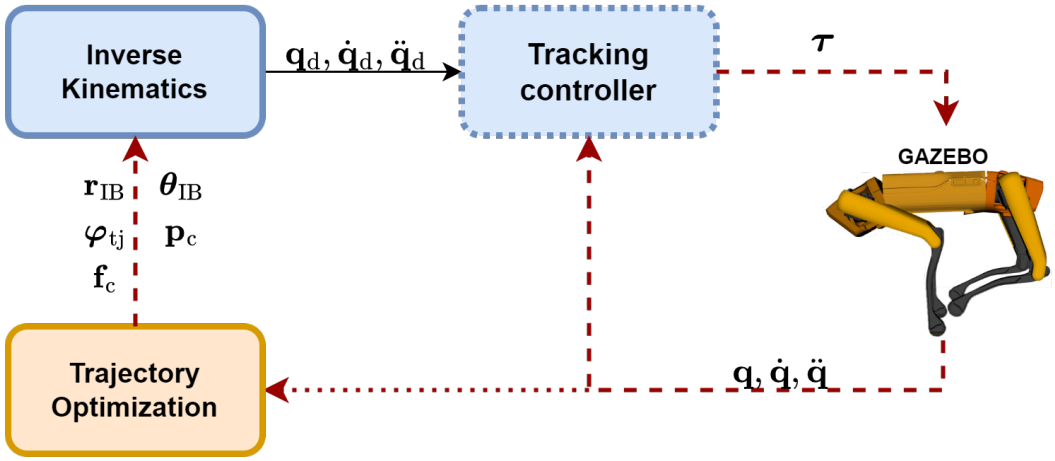


Figure 7-3: Trajectory tracking control structure with ROS information streams (red-dashed) and uncompleted elements (dotted).

While it is demonstrated that the modeling framework can be integrated into a motion optimization algorithm, it can also be extended to other model-based trajectory optimization algorithms. Similarly, this dynamic representation can also be applied to the design of model-based tracking controllers, such as Model-Predictive Control (MPC). For future studies, testing this framework on an MPC algorithm like Optimal Control for Switched Systems (OCS2) can be interesting.

Finally, the option to switch between the rigid and the flexible torso design described in Figure 5-1 bounds the limits of the torso joints to the nominal position. However, this behavior is typically suboptimal regarding computational effort as the Nonlinear Programming (NLP) still allocates a decision variable to the optimization. Therefore, disabling the construction of the torso joint decision variables would be more consistent when selecting a rigid torso configuration.

Chapter 8

Conclusion

8-1 Summary

This thesis addresses the main challenge in legged robotics of enhancing the adaptability and efficiency of legged robots in general. The conclusions in several related studies confirm that four-legged robots equipped with flexible spines typically improve the robots' adaptability and efficiency. However, motion planning algorithms specifically designed for such robots are limited. Therefore, the primary focus of this research lies in developing a computationally efficient model-based motion optimization algorithm for quadruped robots with an actuated articulated torso.

The representation of the system dynamics is an essential element in motion planning. The description of the system dynamics enforces a dynamically consistent motion trajectory. However, especially in legged systems, the dynamic representation typically has detrimental effects on the computational load of the optimization problem. These effects are mainly introduced by the complex nonlinear behavior and the many dimensions that describe the behavior. Therefore, a detailed analysis of conventional dynamic models for legged robots was provided. Further, it was observed that for articulated systems, the conventional models do not meet the requirements regarding computational efficiency, accuracy, or ability to exploit the articulation of the torso. Therefore, a new dynamic model was proposed that provides a low-dimensional representation while also considering the effects of the torso articulation.

A well-known motion optimization algorithm was adapted to the new system dynamics and constraints in the next phase. Ultimately, the motion optimization generated feasible trajectories that showed behavior similar to mammalian motion seen in nature. A careful validation process further verified that the dynamic representation was derived correctly and that the computed trajectories were dynamically feasible. Finally, the findings indicate significant improvements in the robot's adaptability and energy efficiency, showcasing the potential impact of this technology in advancing robotic mobility and versatility. Future work could explore further algorithmic enhancements and broader applications.

8-2 Conclusions

This study's main objective was the derivation of a framework to adapt existing motion planning algorithms to robots with actuated torso joints. As discussed, this objective primarily depends on selecting a dynamically consistent and computationally efficient dynamic constraint.

The main conclusion drawn from this study is that the proposed Triple Rigid Body Dynamics (TRBD) modeling framework can represent the dynamic behavior of an articulated quadruped robot well. Conventional representations used for motion planning algorithms were either too complex or dynamically deficient. Contrarily, the proposed representation lies exactly between these models regarding complexity and accuracy. It was demonstrated that the proposed low-dimensional representation is dynamically consistent with the high-dimensional model. Finally, the versatility and the application to other torso configurations were discussed.

Furthermore, it was demonstrated that a well-known Trajectory Optimization for legged systems can easily be adapted to the new framework. Within a few months, the algorithm was adapted by constructing new decision variables, integrating the new dynamic constraint, and adapting the kinematic constraint.

When testing the adapted Trajectory Optimization algorithm with a simulation model, it was observed that the integration of the torso articulation improved the ability to find a feasible solution. In the Trajectory Optimization algorithm, the articulation of the torso acts as an additional control input on the base, allowing it to actuate the base during flight phases. This control input is absent in the low-dimensional model used for rigid torso systems, making it solely dependent on the contact forces. Furthermore, the trajectories demonstrated a behavior that was expected from previous studies in the motion of running animals. With respect to the disabled torso articulation, the smoothness of the base trajectory and the magnitude of the contact forces benefit significantly.

In the future, it would be exciting to see this framework being extended to other motion optimization or motion control frameworks. Hopefully, this study has contributed to the improvement of the performance of quadruped robots in general. Ultimately, we hope to encourage more research in the planning and control of quadruped robots with an articulated torso.

Appendix A

Appendix

A-1 ZYX Euler mapping

When using ZYX Euler angles, we can describe the rotation from the inertial frame $\mathcal{I}$ to the base frame $\mathcal{B}$ using the ZYX (yaw-pitch-roll) euler angles described by $\boldsymbol{\theta} = [\ \phi \quad \theta \quad \psi\ ]^\top$. The rotation is then described as follows, with $c_x = \cos(\psi)$, $s_z = \sin(\phi)$, etc:

$$\mathbf{R}_{\mathrm{IB}} = \mathbf{R}_{\mathrm{z}}(\phi)\mathbf{R}_{\mathrm{y}}(\theta)\mathbf{R}_{\mathrm{x}}(\psi) = \begin{bmatrix} c_y c_z & c_z s_x s_y - c_x s_z & s_x s_z + c_x c_z s_y \\ c_y s_z & c_x c_z + s_x s_y s_z & c_x s_y s_z - c_z s_x \\ -s_y & c_y s_x & c_x c_y \end{bmatrix} \tag{A-1}$$

To map the time-derivative of the Euler angles to the angular velocity ${}_{\mathcal{I}}\boldsymbol{\omega}_{IB}$, a mapping matrix $\mathbf{E}$ can be derived for the selected Euler parameterization.

$$\boldsymbol{\omega}_{\mathrm{IB}} = \mathbf{E}_{\mathrm{R}}(\boldsymbol{\theta}_{\mathrm{IB}}) \cdot \dot{\boldsymbol{\theta}}_{\mathrm{IB}} \tag{A-2a}$$

$$\dot{\boldsymbol{\omega}}_{\mathrm{IB}} = \mathbf{E}_{\mathrm{R}}(\boldsymbol{\theta}_{\mathrm{IB}}) \cdot \ddot{\boldsymbol{\theta}}_{\mathrm{IB}} + \dot{\mathbf{E}}_{\mathrm{R}}(\dot{\boldsymbol{\theta}}_{\mathrm{IB}}, \boldsymbol{\theta}_{\mathrm{IB}}) \cdot \dot{\boldsymbol{\theta}}_{\mathrm{IB}} \tag{A-2b}$$

$$\mathbf{E}(\boldsymbol{\theta}) = \begin{bmatrix} 0 & -s_z & c_y c_z \\ 0 & c_z & c_y s_z \\ 1 & 0 & -s_y \end{bmatrix} \tag{A-3}$$

A-2 Derivation of the Rigid Body Dynamics (RBD) matrices

Euler-Lagrange method

For the derivation of the RBD system matrices of a multi-body legged-system, multiple different methods can be used. To provide some insight into the full system dynamics, the Euler-Lagrange method is described below, which exploits the conservation of energy. The Lagrangian $\mathcal{L} \in \mathbb{R}$ describes the net energy in the system consisting of the kinetic energy

$\mathcal{T} \in \mathbb{R}$ and potential energy $\mathcal{U} \in \mathbb{R}$, as described by [22]. The Lagrangian is used to define the Euler-Lagrange equation in Equation A-9, and provides a general form for the Equation of Motion (EoM) of a multi-body system.

$$\mathcal{L}\left(\dot{\mathbf{q}}, \mathbf{q}\right) = \mathcal{T}\left(\dot{\mathbf{q}}, \mathbf{q}\right) - \mathcal{U}\left(q\right) \tag{A-4a}$$

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}_i} - \frac{\partial \mathcal{L}}{\partial \mathbf{q}_i} = \tau, \quad i = 1, \ldots, n \tag{A-4b}$$

The total kinetic energy in the floating-base system can be described by summing the kinetic energy of each individual rigid body i. The kinetic energy of an rigid object i is dependent on its spatial velocity $\mathbf{v}_i = [\dot{\mathbf{r}}_i{}^\intercal, \boldsymbol{\omega}_i^\intercal]^\intercal$. The potential energy is dependent on the absolute Center of Mass (CoM) position of body i and its mass m_i. These relations can be observed below:

$$\mathcal{T} = \sum_{i=1}^{n_b} \left(\frac{1}{2} m_i \dot{\mathbf{r}}_i^\intercal \dot{\mathbf{r}}_i + \frac{1}{2} \boldsymbol{\omega}_i^\intercal \mathbf{I}_i(\mathbf{q}) \boldsymbol{\omega}_i \right), \tag{A-5a}$$

$$\mathcal{U} = \sum_{i=1}^{n_b} \mathbf{r}_{c,i}^\intercal(\mathbf{q}) \mathbf{F}_{g,i}, \tag{A-5b}$$

where the Moment of Inertia (MoI) tensor of link i is denoted by $\mathbf{I}_i$. It is important to note that the CoM position and velocity vectors are represented in a common frame and thus, the inertia tensor needs to be provided with respect to this frame as well. To transform the reference frame of the MoI matrix, a similarity transformation is used as shown in Appendix **??**.

As described in Section 3-3-1, the spatial velocity of the body i can be described by Equation (2-12) and Equation (2-13) respectively. The spatial Jacobian ${}_{\mathcal{I}}\mathbf{J}_i$ maps the generalized velocity vector to the spatial velocity of CoM-frame of body i. This spatial Jacobian allows a more convenient derivation of the kinetic energy, as shown in Equation (A-6a). Furthermore, the potential energy of each link $\mathcal{U}_i$ can be described by the product of the CoM-position $\mathbf{r}_{c,i}$ and the gravitational force $\mathbf{F}_{g,i} = -m_i \mathbf{g}$.

$$\mathcal{T} = \frac{1}{2} \dot{\mathbf{q}}^\intercal \underbrace{\left(\sum_{i=1}^{n_b} \left(m_i \mathbf{J}_{P,i}^\intercal \mathbf{J}_{P,i} + \mathbf{J}_{R,i}^\intercal {}_I\mathbf{I}_i(\mathbf{q}) \mathbf{J}_{R,i} \right) \right)}_{\mathbf{M}(\mathbf{q})} \dot{\mathbf{q}}, \tag{A-6a}$$

$$\mathcal{U} = \sum_{i=1}^{n_b} \mathbf{r}_{c,i}^\intercal(\mathbf{q}) \mathbf{F}_{g,i} = \sum_{i=1}^{n_b} m_i g h_i = \mathbf{P}(\mathbf{q}) \tag{A-6b}$$

Ultimately, the Lagrangian $\mathcal{L}$ can be described as follows, using the generalized Mass matrix $\mathbf{M}(\mathbf{q})$.

$$\mathcal{L}(\dot{\mathbf{q}}, \mathbf{q}) = \frac{1}{2} \dot{\mathbf{q}}^\intercal \mathbf{M}(\mathbf{q}) \dot{\mathbf{q}} + \mathbf{P}(\mathbf{q}) \tag{A-7a}$$

Using the Euler-Lagrange equation in (A-4b), we can derive the dynamic equation provided in Equation (3-11).

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}} - \frac{\partial \mathcal{L}}{\partial \mathbf{q}} = \underbrace{\mathbf{M}(\mathbf{q})}_{\mathbf{M}(\mathbf{q})} \ddot{\mathbf{q}} + \underbrace{\dot{\mathbf{M}}(\mathbf{q})\dot{\mathbf{q}} - \frac{1}{2}\dot{\mathbf{q}}^\intercal \frac{\partial}{\partial \mathbf{q}} \mathbf{M}(\mathbf{q}) \dot{\mathbf{q}}}_{\mathbf{b}(\dot{\mathbf{q}}, \mathbf{q})} + \underbrace{\frac{\partial}{\partial \mathbf{q}} \mathbf{P}(\mathbf{q})}_{\mathbf{g}(\mathbf{q})} \tag{A-8}$$

Projected Newton-Euler method

Similarly, we can use a combination of the Lagrange method and the Newton-Euler method as described in [22], to extract the matrices directly. This method combines the Newton-Euler formulation with the constraint consistent Lagrange formulation. The general definitions of the dynamic model matrices for floating-base systems are provided below.

$$\mathbf{M}(\mathbf{q}) = \sum_{i=1}^{n_b} \left(\mathbf{J}_{p,c_i}^{\intercal} m_i \mathbf{J}_{r,c_i} + \mathbf{J}_{r,c_i}^{\intercal} {}_I\mathbf{I}_i \mathbf{J}_{r,c_i} \right) \tag{A-9a}$$

$$\mathbf{b}(\dot{\mathbf{q}}, \mathbf{q}) = \sum_{i=1}^{n_b} \left(\mathbf{J}_{p,c_i}^{\intercal} m_i \dot{\mathbf{J}}_{p,c_i} \dot{\mathbf{q}} + \mathbf{J}_{r,c_i}^{\intercal} \left(\mathbf{I}_i \dot{\mathbf{J}}_{p,c_i} \dot{\mathbf{q}} + \boldsymbol{\omega}_i \times {}_I\mathbf{I}_i \boldsymbol{\omega}_i \right) \right) \tag{A-9b}$$

$$\mathbf{g}(\mathbf{q}) = \sum_{i=1}^{n_b} \left({}_I\mathbf{J}_{p,c_i}^{\intercal} \mathbf{g}_i \right) \tag{A-9c}$$

A-3 System Jacobian

Instead of deriving each spatial Jacobian individually, a convenient derivation for system Jacobian is described in [25]. This stacked Jacobian matrix maps the generalized velocity vector $\dot{\mathbf{q}}$ to the system spatial velocity vector $\mathbf{v}$. It is based on the observation that the spatial velocity of body i $\mathbf{v}_i$ is directly related the the spatial velocity of its predecessor $p(i)$ with the following relation:

$$\mathbf{v}_i = {}^i\mathbf{X}_{p(i)} \mathbf{v}_{p(i)} + \boldsymbol{\Phi}_i \dot{\mathbf{q}}_i, \tag{A-10}$$

where ${}^i\mathbf{X}_{p(i)} \in \mathbb{R}^{6\times6}$ is the spatial transform, which transforms the spatial motion vector from $p(i)$ to i coordinates. Furthermore, $\boldsymbol{\Phi}_i \in \mathbb{R}^6$ is the joint vector of joint i and is dependent on the type of joint. An expression for the system velocity vector $\mathbf{v} \in \mathbb{R}^{6n_b}$ can be derived by using the expression in Equation (A-10) for all links:

$$\boldsymbol{\Phi}_i \dot{\mathbf{q}}_i = \mathbf{v}_i - {}^i\mathbf{X}_{p(i)} \mathbf{v}_{p(i)} \tag{A-11a}$$

$$\boldsymbol{\Phi} \dot{\mathbf{q}} = \mathbf{P}\mathbf{v} \tag{A-11b}$$

$$\mathbf{v} = \mathbf{P}^{-1} \boldsymbol{\Phi} \dot{\mathbf{q}} = \mathbf{J} \dot{\mathbf{q}} \tag{A-11c}$$

Here, the sparse and lower-triangular incidence matrix is denoted by $\mathbf{P} \in \mathbb{R}^{6n_b \times 6n_b}$ and the system Jacobian is denoted by $\mathbf{J} \in \mathbb{R}^{6n_b \times n}$.

A-4 Kinematic singularities

It is important to note that the Jacobian matrix can become rank-deficient when a robot configuration $\mathbf{q}$ is singular. This singularity will result in infeasible joint velocities in the singular direction. However, the effect of the kinematic singularity is also experienced near singular configurations. Common methods, to detect the singularity are listed in [24], these are all related to a measure of the Jacobian matrix, like the *manipulability measure* $\mu = \sqrt{\det(\mathbf{J})}$,

the *condition number* $\kappa = \frac{\sigma_{\max}}{\sigma_{\min}}$ or the *smallest singular value* $\sigma_{\min}$. All these singularity measures can be computed by retrieving the Singular Value Decomposition (SVD):

$$\mathbf{J} = \sum \frac{1}{\bar{\sigma}_{\mathrm{i}}} \bar{\mathbf{v}}_{\mathrm{i}} \bar{\mathbf{u}}_{\mathrm{i}}^{\mathsf{T}} \tag{A-12}$$

where $\bar{\sigma}_i$ is the singular value, $\bar{\mathbf{v}}_i$ is the output singular vector and $\bar{\mathbf{u}}_i$ is the input singular value. To increase the robustness against singularity, a modification of the planned trajectory, elimination of the unfeasible task velocities, or the damped least-squares method can be useful [24]. The modification of the planned trajectory could include the avoidance of singular configurations or re-planning the joint-space trajectory in the region around the singular configuration. The elimination of the unfeasible task velocities makes use of the input singular value, by avoiding task-space velocities along the unfeasible direction $\bar{\mathbf{u}}_{\min}^{\mathsf{T}} \mathbf{v} \approx 0$.

The damped least-squares method is a commonly used method to make the differential kinematics more robust to singularity [24][35]. The damped least-squares method provides a trade-off between the least-square $||\mathbf{J}\dot{\mathbf{q}} - \mathbf{v}||_2$ and the minimum norm $||\dot{\mathbf{q}}||_2$. In essence, it is a trade-off between the accuracy and feasibility of the trajectory. The solution can be described as:

$$\dot{\mathbf{q}} = \mathbf{J}^{\mathsf{T}} \left(\mathbf{J}\mathbf{J}^{\mathsf{T}} + \lambda^2 \mathbb{I} \right)^{-1} \mathbf{v} \tag{A-13}$$

where $\lambda \in \mathbb{R}$ is a damping factor, which is selected based on the closeness to singularity. Various derivations of the damping factors are discussed in [35]. While this method can be very effective for increasing the robustness of the solution, it typically affects the accuracy of the solution in all directions. A more appropriate approach would be to filter the singular end-effector velocities using the output singular vectors $\bar{\mathbf{u}}$, corresponding to the singular values close to the singular region. This filtering can be applied using a damping factor $\beta \in \mathbb{R}$, that provides damping only along the unfeasible velocity.

$$\dot{\mathbf{q}} = \mathbf{J}^{\mathsf{T}} \left(\mathbf{J}\mathbf{J}^{\mathsf{T}} + \lambda^2 \mathbb{I} + \beta^2 \sum \bar{\mathbf{u}}_{\mathrm{i}} \bar{\mathbf{u}}_{\mathrm{i}}^{\mathsf{T}} \right)^{-1} \mathbf{v}, \tag{A-14}$$

A-5 Continuous parameterization

One important aspect of the Nonlinear Programming (NLP) problem defined by the Trajectory Optimizer for Walking Robots (TOWR) is how each optimization variable is constructed and optimized. First, the total time interval from t_0 to T is divided into smaller time intervals. Then, for each optimization variable, a set of fourth-order polynomials is constructed for each time interval, as shown in Equation A-15. By constraining the intersections to align, a continuous spline is constructed for each parameter from t_0 to T.

$$\begin{aligned} x(t) &= a_0 t^3 + a_1 t^2 + a_2 t + a_3 t, \\ \dot{x}(t) &= a_0 t^2 + a_1 t + a_2, \\ \ddot{x}(t) &= a_0 t + a_1 \end{aligned} \tag{A-15}$$

The NLP problem then optimizes over the coefficients of each variable polynomial while enforcing a set of constraints at predefined time intervals.

References

[1] J. George, *5 Real-World Applications of Quadruped Robot*, May 2022. [Online]. Available: https://www.robotics247.com/article/5_real_world_applications_of_quadruped_robots.

[2] A. Shah, *Four-legged robot designed to deliver packages*, Jan. 2022. [Online]. Available: https://www.controleng.com/articles/four-legged-robot-designed-to-deliver-packages/.

[3] H. Ferrolho, W. Merkt, V. Ivan, W. Wolfslag, and S. Vijayakumar, "Optimizing Dynamic Trajectories for Robustness to Disturbances Using Polytopic Projections," *IEEE International Conference on Intelligent Robots and Systems*, pp. 7477–7484, Mar. 2020.

[4] P. Grinter, *Hazard Detection & Rescue Quadruped Robot Released*, Jul. 2022. [Online]. Available: https://www.unmannedsystemstechnology.com/2022/07/hazard-detection-rescue-quadruped-robot-released/.

[5] L. T. Phan, Y. H. Lee, Y. H. Lee, H. Lee, H. Kang, and H. R. Choi, "Study on effects of spinal joint for running quadruped robots," *Intelligent Service Robotics*, vol. 13, no. 1, pp. 29–46, Jan. 2020, ISSN: 1861-2776. DOI: 10.1007/s11370-019-00297-4. [Online]. Available: http://link.springer.com/10.1007/s11370-019-00297-4.

[6] C. Fisher and A. Patel, "On the optimal spine morphology of rapidly accelerating quadrupeds," *SAIEE Africa Research Journal*, vol. 112, no. 3, pp. 126–133, Sep. 2021.

[7] M. Khoramshahi, A. Parsa, A. Ijspeert, and M. N. Ahmadabadi, "Natural dynamics modification for energy efficiency: A data-driven parallel compliance design method," in *Proceedings - IEEE International Conference on Robotics and Automation*, Institute of Electrical and Electronics Engineers Inc., Sep. 2014, pp. 2412–2417.

[8] C. Wang, Y. Li, H. Sun, X. Hou, C. Fan, and Y. Yang, "The Dynamics Research of Quadruped Robotics with Flexible Spine: A Review," in *2021 6th International Conference on Control, Robotics and Cybernetics, CRC 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 13–18.

[9] E. Koco and Z. Kovacic, "Multiobjective locomotion optimization of quadruped robot with different 2DOF configurations of actuated spine," in *2016 24th Mediterranean Conference on Control and Automation (MED)*, IEEE, Jun. 2016, pp. 504–511.

[10] G. A. Folkertsma, S. Kim, and S. Stramigioli, "Parallel stiffness in a bounding quadruped with flexible spine," in *IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 2210–2215, ISBN: 9781467317375. DOI: 10.1109/IROS.2012.6385870.

[11] S. Pouya, · . M. Khodabakhsh, · . A. Spröwitz, and · . A. Ijspeert, "Spinal joint compliance and actuation in a simulated bounding quadruped robot," vol. 41, pp. 437–452, 2017. DOI: 10.1007/s10514-015-9540-2.

[12] J. Chen, Z. Liang, Y. Zhu, *et al.*, "Towards the exploitation of physical compliance in segmented and electrically actuated robotic legs: A review focused on elastic mechanisms," *Sensors (Switzerland)*, vol. 19, no. 24, Dec. 2019.

[13] A. W. Winkler, "Optimization-based motion planning for legged robots," Ph.D. dissertation, ETH Zurich, Zürich, 2018. DOI: 10.3929/ETHZ-B-000272432. [Online]. Available: https://doi.org/10.3929/ethz-b-000272432.

[14] S. Bhattacharya, A. Singla, Abhimanyu, *et al.*, "Learning Active Spine Behaviors for Dynamic and Efficient Locomotion in Quadruped Robots," *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, May 2019.

[15] O. K. Adak, B. Bahceci, and K. Erbatur, "Modeling of a Quadruped Robot with Spine Joints and Full-Dynamics Simulation Environment Construction," Mar. 2022. [Online]. Available: http://arxiv.org/abs/2203.09622.

[16] W. Du, M. Fnadi, E. Moullet, and F. Benamar, "Leg Centroidal Dynamics Based New Locomotion Principle of a Quadruped Robot with On-line Legged Motion Generation," *Journal of Intelligent & Robotic Systems*, vol. 103, no. 4, p. 70, Dec. 2021, ISSN: 0921-0296. DOI: 10.1007/s10846-021-01503-1. [Online]. Available: https://link.springer.com/10.1007/s10846-021-01503-1.

[17] H. Chai, Y. Li, R. Song, *et al.*, "A survey of the development of quadruped robots: Joint configuration, dynamic locomotion control method and mobile manipulation approach," *Biomimetic Intelligence and Robotics*, vol. 2, no. 1, p. 100 029, Mar. 2022.

[18] M. Hildebrand, "Motions of the Running Cheetah and Horse," *Journal of Mammalogy*, vol. 40, no. 4, p. 481, Nov. 1959, ISSN: 00222372. DOI: 10.2307/1376265.

[19] N. Schilling and R. Hackert, "Sagittal spine movements of small therian mammals during asymmetrical gaits," *Journal of Experimental Biology*, vol. 209, no. 19, pp. 3925–3939, Oct. 2006, ISSN: 1477-9145. DOI: 10.1242/jeb.02400.

[20] T. Kamimura, S. Aoi, Y. Higurashi, N. Wada, K. Tsuchiya, and F. Matsuno, "Dynamical determinants enabling two different types of flight in cheetah gallop to enhance speed through spine movement," *Scientific Reports*, vol. 11, no. 1, Dec. 2021, ISSN: 20452322. DOI: 10.1038/s41598-021-88879-0.

[21] R. Tedrake, *Underactuated Robotics, Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. 2023. [Online]. Available: https://underactuated.csail.mit.edu.

[22] "Robot Dynamics Lecture Notes," Robotic Systems Lab, ETH Zurich, Zurich, Tech. Rep., 2017.

[23] D. Dholakiya, S. Bhattacharya, A. Gunalan, *et al.*, "Design, Development and Experimental Realization of a Quadrupedal Research Platform: Stoch," Jan. 2019. [Online]. Available: http://arxiv.org/abs/1901.00697.

[24] S. Chiaverini, G. Oriolo, and I. D. Walker, "Kinematically Redundant Manipulators," in *Springer Handbook of Robotics*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 245–268. DOI: 10.1007/978-3-540-30301-5{_}12. [Online]. Available: http://link.springer.com/10.1007/978-3-540-30301-5_12.

[25] D. Orin and A. Goswami, "Centroidal Momentum Matrix of a humanoid robot: Structure and properties," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Sep. 2008, pp. 653–659, ISBN: 978-1-4244-2057-5. DOI: 10.1109/IROS.2008.4650772.

[26] P. M. Wensing and D. E. Orin, "Improved Computation of the Humanoid Centroidal Dynamics and Application for Whole-Body Control," *International Journal of Humanoid Robotics*, vol. 13, no. 1, Mar. 2016, ISSN: 02198436. DOI: 10.1142/S0219843615500395.

[27] F. Farshidian *et al.*, *OCS2: An open source library for optimal control of switched systems*, [Online]. Available: https://github.com/leggedrobotics/ocs2.

[28] P. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent, "A fully asynchronous multifrontal solver using distributed dynamic scheduling," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 1, pp. 15–41, 2001.

[29] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "Rviz: A toolkit for real domain data visualization," *Telecommun. Syst.*, vol. 60, no. 2, pp. 337–345, Oct. 2015, ISSN: 1018-4864. DOI: 10.1007/s11235-015-0034-5. [Online]. Available: https://doi.org/10.1007/s11235-015-0034-5.

[30] J. Carpentier, G. Saurel, G. Buondonno, *et al.*, "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *SII 2019 - International Symposium on System Integrations*, Paris, France, Jan. 2019. [Online]. Available: https://hal.laas.fr/hal-01866228.

[31] S. Seok, A. Wang, M. Y. Chuah, *et al.*, "Design Principles for Energy-Efficient Legged Locomotion and Implementation on the MIT Cheetah Robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 3, pp. 1117–1129, Jun. 2015, ISSN: 1083-4435. DOI: 10.1109/TMECH.2014.2339013. [Online]. Available: https://ieeexplore.ieee.org/document/6880316/.

[32] M. Kockelkoren, *Playlist: Model-based Trajectory Optimization for Articulated Quadruped Robots*, Sep. 2023. [Online]. Available: https://www.youtube.com/playlist?list=PLmZ4ovuqQhMDMwtWewSW37cCfZYxINFxL.

[33] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[34] M. Quigley, K. Conley, B. P. Gerkey, *et al.*, "Ros: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[35] A. S. Deo and I. D. Walker, "Overview of Damped Least-Squares Methods for Inverse Kinematics of Robot Manipulators," Tech. Rep., 1995, p. 43.

Glossary

List of Acronyms

TRBD	Triple Rigid Body Dynamics
CD	Centroidal Dynamics
CMM	Centroidal Momentum Matrix
CoM	Center of Mass
CoP	Center of Pressure
CRBI	Composite Rigid Body Inertia
DOF	Degree of Freedom
EoM	Equation of Motion
FK	Forward Kinematics
HAA	Hip Abduction-Adduction
HFE	HIP Flexion-Extension
LF	Left-Front
LH	Left-Hind
IK	Inverse Kinematics
KFE	Knee Flexion-Extension
LIP	Linear Inverted Pendulum
MoI	Moment of Inertia
NLP	Nonlinear Programming
OCS2	Optimal Control for Switched Systems
RBD	Rigid Body Dynamics
RF	Right-Front
RH	Right-Hind
ROM	Range of Motion

ROS	Range of Motion
SRBD	Single Rigid Body Dynamics
SVD	Singular Value Decomposition
TO	Trajectory Optimization
TOWR	Trajectory Optimizer for Walking Robots
URDF	Unified Robotics Description Format

List of Symbols

Coordinate frames

$\mathcal{B}$	Base frame
$\mathcal{C}$	CoM frame
$\mathcal{E}$	End-effector frame
$\mathcal{F}$	Front torso frame
$\mathcal{G}$	Centroidal frame
$\mathcal{H}$	Hind torso frame
$\mathcal{I}$	Inertial frame

Matrices

$\bar{\mathbf{I}}$	Spatial inertia matrix
$\bar{\mathbf{R}}$	Spatial rotation matrix
$\mathbf{A}$	Momentum matrix
$\mathbf{E}$	Euler mapping matrix
$\mathbf{I}$	Moment of Inertia matrix
$\mathbf{J}$	Spatial Jacobian matrix
$\mathbf{R}$	Rotation matrix
$\mathbf{S}(\cdot)$	Skew-symmetric matrix function
$\mathbf{T}$	Coordinate transformation matrix

Scalars

β	Filtering damping factor
$\ell(\cdot)$	Objective function
λ	Damping factor
$\det(\cdot)$	Determinant function
ϕ	roll angle
ψ	yaw angle
θ	pitch angle
φ	Joint angle
h	Terrain height
m	Mass
t	Time

Vectors

$\bar{\boldsymbol{\sigma}}$	Singular values
$\bar{\mathbf{u}}$	Input singular vector
$\bar{\mathbf{v}}$	Output singular vector
$\boldsymbol{\omega}$	Angular velocity vector
$\boldsymbol{\sigma}$	Singular values
$\boldsymbol{\tau}$	RBD Generalized torque vector
$\boldsymbol{\theta}$	Euler angle vector
$\boldsymbol{\varphi}$	Joint angle vector
$\boldsymbol{a}$	Spatial linear acceleration vector
$\boldsymbol{f}$	Force vector
$\boldsymbol{v}$	Spatial linear velocity vector
$\Delta\mathbf{T}$	Phase-durations vector
$\mathbf{b}$	RBD Coriolis vector
$\mathbf{b}_{\text{ROM}}$	Range of Motion (ROM) boundaries
$\mathbf{f}_{\text{IK}}$	Inverse Kinematics function
$\mathbf{F}_c$	RBD Force vector
$\mathbf{g}$	RBD Gravitational vector
$\mathbf{g}_{\text{d}}$	Trajectory Optimization (TO) Dynamic constraint
$\mathbf{g}_{\text{k}}$	TO Kinematic constraint
$\mathbf{h}$	Spatial momentum vector
$\mathbf{k}$	Angular momentum vector
$\mathbf{l}$	Linear momentum vector
$\mathbf{n}(\cdot)$	Vector normal
$\mathbf{n}$	Moment vector
$\mathbf{p}$	Foot position vector
$\mathbf{q}$	Generalized state vector
$\mathbf{r}$	Position vector
$\mathbf{u}$	Input trajectory vector
$\mathbf{v}$	Spatial velocity vector
$\mathbf{w}$	Spatial wrench vector
$\mathbf{x}$	State trajectory vector
$\mathcal{F}(\cdot)$	Friction cone