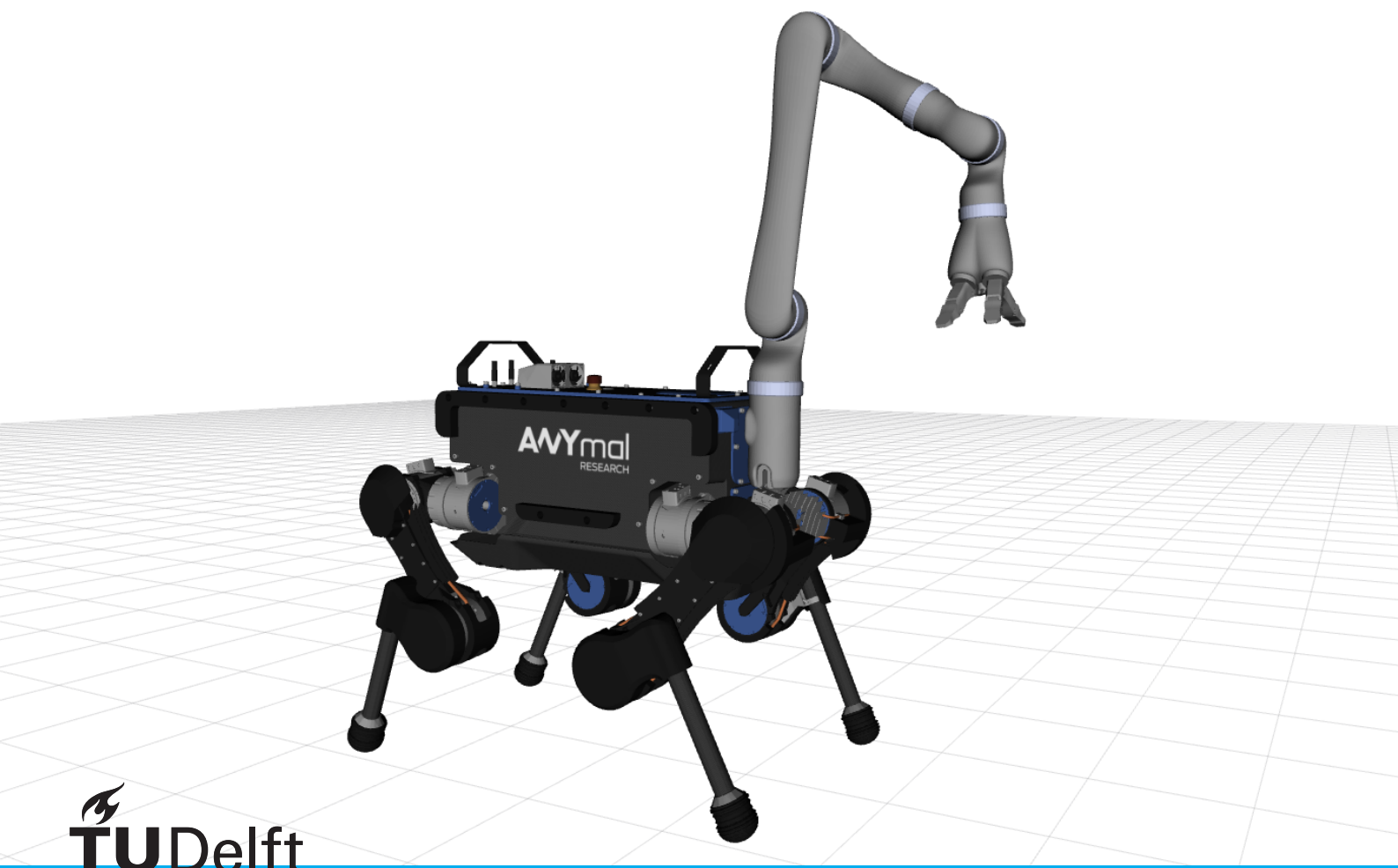
Control of a quadrupedal manipulator using hierarchical inverse dynamics

K. Krämer

Supervised by:
Dario Bellicoso (ETH)
Dr. Christian Gehring (ETH)
Mukunda Bharatheesha (TU Delft)
Prof. Dr. Heike Vallery (TU Delft)



TU Delft

Control of a quadrupedal manipulator using hierarchical inverse dynamics

by

K. Krämer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday June 20, 2018 at 2:00 PM.

Student number:	4419642	
Project duration:	July 1, 2017 – December 31, 2017	
Thesis committee:	Prof. Dr. H. Vallery,	TU Delft, supervisor
	M. Bharatheesha,	TU Delft, supervisor
	Dr. J. Alonso-Mora,	TU Delft

This thesis is confidential and cannot be made public until December 10, 2018.

TU Delft

Abstract

Many robots exist which are fixed to their environment and excel in a specific task. In contrast, the prevalence of general purpose mobile robots with manipulation capabilities is still low, despite various applications of such systems such as: disaster response, payload delivery, and assistive/service tasks. A suitable design for such a robot would be that of a torque-controllable quadrupedal manipulator. Its capability for legged locomotion enables high mobility, specifically on rough terrain, while its quadrupedal morphology provides a relatively large and stable base of support compared to bipedal robots. Torque-controlled joints allow for safer and more controllable interaction with the environment.

For such a system a challenge lies in the design of a controller that actually achieves the promising capabilities for locomotion and manipulation that the mechanical system offers. One of the currently most promising control frameworks for this purpose is that of hierarchical inverse dynamics. This real-time whole-body control framework allows for dynamic whole-body motions and compliant interaction with the environment while enforcing a strict priority between the desired control tasks. Although promising results have previously been attained with such controllers, it has not yet been applied for the control of a quadrupedal manipulator. The focus of this thesis is on the implementation of a hierarchical inverse dynamics controller for the control of a simulated quadrupedal manipulator, with a particular focus on the design of prioritized sets of control tasks which generate forms of stationary whole-body manipulation.

First a basic set of prioritized control tasks is presented, which is shown to satisfy the robot's most crucial control requirements. Subsequently three extended sets of control tasks are presented, which result in additional desirable emergent behavior of the robot. The first one of these depends on the inclusion of kinematic joint limit tasks to prevent kinematic singularities and self-collision, and to trigger whole-body reaching motion. Secondly a set of tasks is presented which is focused on utilizing motion of some of the torso's degrees of freedom to optimize the arm's posture according to a posture-related cost function. Thirdly a set of tasks is presented which enables contact force control at the end-effector for force-based manipulation. In addition to this final set of tasks, a higher-level controller is presented which detects external forces acting on the robot and computes a desired shift of the position of the robot's center of mass in order to mitigate the balance-disturbing effects of external forces. All of the presented sets of tasks do not exclude each other and allow to be implemented simultaneously in order to combine the individual benefits that they offer.

The results of this research project show that hierarchical inverse dynamics control can be successfully applied for the control of a simulated torque-controllable quadrupedal manipulator. Moreover, it is shown that well-designed sets of prioritized control tasks allow for emergent whole-body behaviors which exploit the advantages that both the robotic system and the control framework offer. Future work will need to investigate the transferability of these results to a physical robot.

Contents

1 Introduction

1.1. Torque-controlled quadrupedal manipulators

Robotics is aimed at providing machines which, instead of humans, can interact with our physical environment. This field has progressed far in industrial settings, where robots typically are fixed to their environment and are developed for very specific tasks only. In contrast, mobile robots that can perform a wide variety of manipulation tasks across an unstructured environment are still rare. This is the case even despite obvious useful applications that this type of robot would have in fields such as disaster response, payload delivery, construction and assistive/service robotics. This lack of generally applicable mobile systems became obvious during the 2011 Fukushima disaster in Japan, where an earthquake and subsequent tsunami hit the Fukushima-Daiichi nuclear power plant. In order to inspect and attempt to mitigate negative effects of the disaster, various teleoperated mobile robots were sent into the power plant, which was too dangerous for humans because of high radiation levels [29]. However, these robots turned out to have great difficulties dealing with: the rough terrain inside the plant, manipulation tasks such as opening doors, and communication issues.

For this reason the US Defense Advanced Research Projects Agency (DARPA) organized the DARPA Robotics Challenge from 2012 to 2015 to stimulate the development of robots that could interface with environments designed for humans, use human tools, and be commanded by humans without specialized training [39]. The competition consisted of a variety of tasks that mimicked a disaster response scenario. Tasks included: traversing rough terrain and stairs, removing rubble, and opening doors and valves. These tasks had to be performed by robots which were remotely controlled through a low-quality network connection. This competition gave an excellent overview of the current state-of-the-art performance of general purpose mobile manipulators.

One of the observations that could be made was that many of the robots used some form of legged locomotion. Legged robots, in contrast to wheeled robots, have the advantage of being able to negotiate unstructured, non-smooth terrain. This capability is essential for tasks that lack a structured and predictable environment, such as disaster areas, typical household settings or natural environments like forests.

Results from the DARPA Robotics Challenge show however that legged locomotion and manipulation are not easily combined. During the competition one of the main problems turned out to be robots losing balance and falling as a result of attempting to manipulate their environment [9]. This is especially a problem for humanoid robots, which many of the participating robots were. These robots are designed according to human morphology in order to function in a human-centered environment, but therefore also have an inherent low stability because of the relatively small base of support offered by the two feet. This illustrates that there is a clear need for robots that can combine manipulation with a reliable and stable means of legged locomotion.

This is a requirement that can be met by quadrupedal robots. These have shown to be able to display a wide

range of highly mobile gaits [36][51][11], while also having the advantage of being inherently more stable than bipedal robots because of the large base of support that four legs offer. Quadrupedal robots would therefore make an ideal base for a general purpose mobile manipulator. A quadrupedal manipulator would be able to combine manipulation capabilities with a high mobility and a large robustness against external forces.

The concept of a quadrupedal mobile manipulator is not new. [41] provides an overview of previously built quadrupedal manipulators and contains several notable recent projects. One of these is IIT's hydraulic torque-controllable HyQ robot [40]. This robot depends on two discoupled controllers for the arm and the legged base, where the controller for the legged base takes into account the expected disturbances generated by dynamic motions of the arm. A different project is Boston Dynamics' BigDog platform which is extended with a 7 DOF arm to create a fully hydraulic torque-controllable robot that can use coordinated whole-body motion to throw a cinder block [8]. The controller combines an off-line computed throwing motion for the arm with their on-line trotting controller to achieve impressive whole-body manipulation. However, they only report on the specific task of cinder block throwing and not on a control framework for general purpose mobile manipulation. Boston Dynamics' newest quadrupedal manipulator, SpotMini, does show on video to be able to combine versatile and robust legged locomotion with general purpose whole-body manipulation [6]. Unfortunately, no published results on this robot are currently available.

It is interesting to note that several of the more recent quadrupedal manipulators in this list[41] have been built with torque-controlled joints instead of traditional position-controlled joints. Compared to position-control, torque-control allows for more advanced control frameworks and safer interaction between the robot and its environment. Indeed in recent years the need for inherently safe robots has been answered by an increasing development of lightweight torque-controllable robots [14][7][5]. As a key feature, these robots allow for the application of inverse dynamics (ID) control: for desired robot motions, a dynamic model of the robot is inverted to compute the required joint torques. Because this approach explicitly compensates for expected gravitational and dynamic disturbances, it only requires relatively low gains for good motion tracking. This is in sharp contrast with more traditional position-controlled robots, which are characterized by the need for high gains to ensure proper tracking performance under the exposure of external disturbances such as gravity. Therefore torque-controlled robots are more adaptable to their environment and safer in interaction, making them more desirable than position-controlled robots in human-centered environments.

A similar shift towards torque-controlled robots is being seen in the field of mobile manipulation with research being performed on newly developed robotic platforms such as on DLR's humanoid TORO robot [16], IIT's quadrupedal HyQ [44] and humanoid COMAN [49], and ETH's quadrupedal ANYmal [26]. Although fully torque-controlled legged manipulators are currently still rare, the aforementioned properties and the increased availability of torque-controllable actuators, make a compelling argument for future quadrupedal manipulators to rely on torque-controllable joints. For this reason the focus in this thesis will be on this type of platform.

1.2. Hierarchical inverse dynamics

This next generation of quadrupedal manipulators will have to be able to deal with rough terrain and be able to safely and robustly interact with a human-centered environment, while also taking full advantage of all of the robot's degrees of freedom (DOFs). This means that the legged base is not only used for locomotion and that any manipulator arms are not only used for manipulation. Instead these two systems would ideally complement each other where possible, in order to fully exploit all of the robot's DOFs for an optimal task performance. The legged base could for example be used to enlarge the manipulator's workspace reachability, or the manipulator could be used to aid in locomotion by engaging in additional contacts with the environment. Besides these behavioral requirements, the possible application of quadrupedal manipulators in for example disaster response would also make it very desirable to have the possibility to teleoperate the robot in real-time.

A big challenge therefore lies not only in the mechanical design, but also in the control of such a quadrupedal manipulator such that these requirements are met. As mentioned in [38], there currently exists a large gap between what legged manipulators can theoretically do, and what they can actually do. A suitable control framework will enable the robot to perform multiple desirable tasks simultaneously, such as: center of mass

motion tracking, manipulator motion tracking, and contact force distribution over the limbs. A control approach that allows for this is that of (multi-task) operational-space control (OSC) [30]. This approach is aimed at intuitively specifying motion of specific parts of the robot (e.g. end-effector) and/or contact force tasks in the Cartesian operational space, after which the control algorithm uses inverse dynamics to compute the required joint torques at the joint level. This approach provides instantaneous solutions based on a linearized model of the robot. This makes it necessary to run the controller at a high ms-level control rate, but also allows for real-time control and direct teleoperation.

One of the key features of multi-task OSC is that it allows for specification of a task hierarchy in which lower priority tasks are only executed as well as is possible without disturbing the performance of higher priority tasks. Although these frameworks traditionally depended on analytical computations [30][25], recent advances have shown a shift towards a numerical optimization-based approach [17][22][10]. The dependence on an optimization algorithm allows for the explicit incorporation of tasks which are formulated as inequality constraints. This is very useful for application on a real robot because it allows for incorporation into the controller of the robot's physical limits, such as kinematic joint limits, actuator torque limits, and the unilateral and friction-dependent nature of contact forces.

Multi-task OSC can currently be considered the 'de facto' standard for the control of legged manipulators, as it is well understood and able to produce practically useable results [38]. This can be illustrated by looking at the control frameworks used by the teams that performed best in the DARPA Robotics Challenge. Many of these teams used a form of multi-task OSC, like for example the top three performing teams [32][47][31]. In the last decades, because of a lack of accurately torque-controllable robots, this control framework has most often been used on an inverse kinematics level for the control of position-controlled robots. However, in recent years several research groups have also started publishing on the application of hierarchical inverse dynamics controllers on torque-controllable robots, resulting in some impressive behaviors [15][21][23][10]. These controller implementations have mostly not yet focused on generally usable mobile manipulation, but usually only focused on subtopics such as multi-contact balance or locomotion. Because most of the few existing fully torque-controllable legged robots have only recently been developed, the application of hierarchical inverse dynamics control is still a strongly developing field of research. Intentions for implementation of such a multi-task inverse dynamics controller on a torque-controlled quadrupedal manipulator platform have been expressed in literature [10][40], but to date no published results on actual implementation exist yet.

1.3. Contribution and outline

Because of their high mobility and inherent stability, torque-controllable quadrupedal manipulators appear to be an excellent platform to meet the demands of a general purpose mobile manipulator. Despite the fact that applying hierarchical inverse dynamics for the control of this specific type of robot appears to be a promising approach, currently no research has yet been published on such an implementation. For these reasons the objective of the research presented in this thesis is to explore specific forms of implementation of an optimization-based hierarchical inverse dynamics control framework for the control of a quadrupedal manipulator, which exploits the characteristic advantages offered by both the control framework and the robotic platform.

In the next chapter the theoretical background for this research will be provided, followed by a more detailed problem statement and research focus description. In Chapters 3 till 7 specific controller implementations will be presented, each of which is focused on meeting a specific set of control requirements for the robot. These chapters each contain their individual Results and Discussion sections. Chapter 8 provide a general discussion on the research presented in this thesis, followed by a conclusion in Chapter 9.

2

Theoretical basis

The research presented in this report was performed at the Robotic Systems Lab (RSL) at ETH Zurich, Switzerland. This lab has considerable experience with the control of quadrupedal robots, such as StarlETH [24] and the more recent ANYmal robot [26]. Plans for the near future include extending the ANYmal robot with manipulation capabilities in order to create a quadrupedal mobile manipulator. It is the simulated version of this robot that the research of this thesis has been conducted on. A state-of-the-art inverse dynamics control framework based on hierarchical optimization has already been successfully used for the control of the ANYmal robot itself [10]. The same control framework is planned to be used for the control of the quadrupedal manipulator and is therefore the control framework which is considered in this thesis. This chapter contains a description of the robot and simulation environment and of the employed control framework.

2.1. Robotic platform

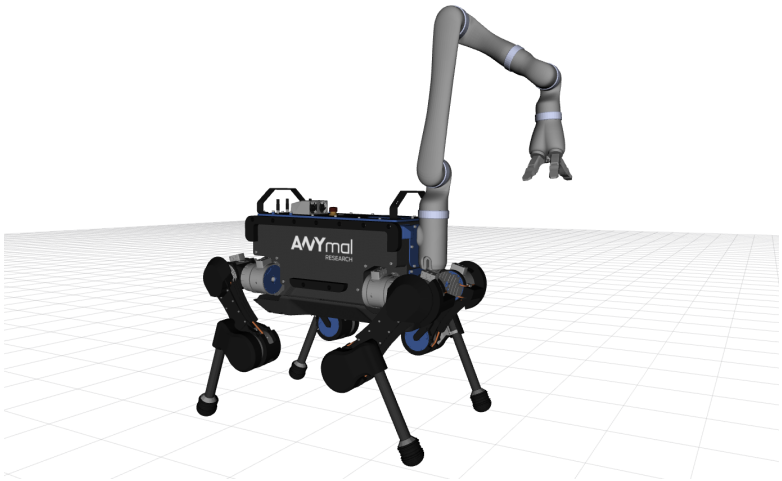


Figure 2.1: Simulated version of the quadrupedal ANYmal robot with 6-DOF Kinova manipulator arm.

The quadrupedal manipulator that is planned to be built by the RSL consists of the existing ANYmal robot, with the addition of a 6 DOF Kinova manipulator arm [3] attached to ANYmal's torso, as shown in Figure 2.1.

ANYmal is a quadrupedal robot developed for operation in harsh environments. It has a mass of 30 kg and a height of 0.5 m, and is built in a modular way for simple maintenance and user-friendly handling. It consists of a torso with four 3-DOF torque-controllable legs with point-feet. All joints are actuated by the RSL's in-house developed ANYdrives [1]. These series-elastic actuators form compliant joint modules with integrated

electronics. The ANYdrives allow for precise torque-control and robustness against impulsive loads during running and jumping. Previous research has demonstrated that ANYmal is capable of executing various gaits such as dynamic walking and crossing rough terrain [10][11][18].

The manipulator that will be mounted on ANYmal to form a quadrupedal manipulator is the Kinova JACO^2 6 DOF-S robotic arm [3]. This robotic arm is equipped with 360 degrees rotatable joints which include torque sensors that allow for torque-control. Its lightweight carbon-fiber design gives it a mass of 4.4 kg and a reach of almost 1 m. When combined with the ANYmal robot it will form a fully torque-controllable mobile manipulator.

2.2. Simulation

Because the actual robot has not been built yet at the time of this research, all experiments have been conducted in the open-source physics simulation software Gazebo [2]. Accurate dynamic models for these simulations were available for both ANYmal and the Kinova arm. Past experiments on the ANYmal robot in the same simulation environment have shown good transferability to implementation of the same control software on the actual robot.

2.3. Control framework

Previous research at the RSL has focused on whole-body control of quadrupedal robots. This was initially done in the form of a hierarchical inverse dynamics control approach for the robot StarlETH [25]. This controller depended purely on analytical methods. In contrast, recent research has been focused on the implementation of a numerically-based inverse dynamics controller which utilizes hierarchical optimization. The main advantage of this approach is that it allows for an easy integration of inequality constraints, for example to take into account the robot's physical limits. This recent control framework has shown very satisfactory results [10][11] and has also been used for the experiments on the quadrupedal manipulator in this thesis. The following subsections will give an overview of this control framework.

2.3.1. Model formulation

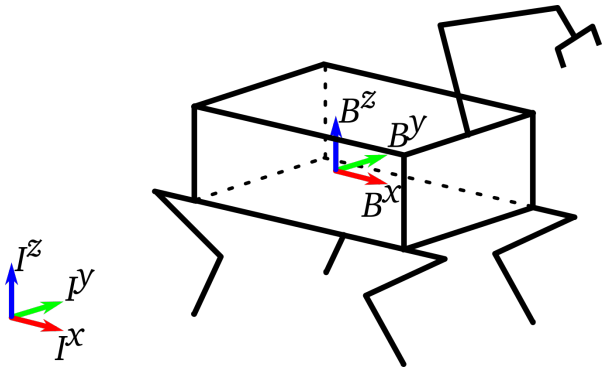


Figure 2.2: Reference frames used to describe the state of the robot. I is an inertial frame that is fixed to the world. The free-floating base frame B is fixed to the robot's torso.

Legged robots can dynamically be modelled as a free-floating base B with limbs attached [46][45]. In this report the same notations for the model are used as in [11]. The state of the robot can be described with respect to an inertial frame I. The position of the floating base, in this case the robot's torso, with respect to the inertial frame and expressed in the inertial frame is written as ${}_I\mathbf{r}_{IB} \in \mathbb{R}^3$. The orientation of the base frame is parametrized as a unit quaternion $\mathbf{q}_{IB} \in SO(3)$. This is the quaternion that projects the components of a vector expressed in the base frame B to the same vector expressed in the inertial frame I. The robot's n_j joint positions are stacked into the vector $\mathbf{q}_j \in \mathbb{R}^{n_j}$. The generalized positions vector $\mathbf{q}$ and the generalized velocity

vector $\mathbf{u}$, which contain all $n_u = 6 + n_j$ degrees of freedom, are then written as

$$\mathbf{q} = \begin{bmatrix} {}_I\mathbf{r}_{IB} \\ \mathbf{q}_{IB} \\ \mathbf{q}_j \end{bmatrix} \in SE(3) \times \mathbb{R}^{n_j} \tag{2.1}$$

$$\mathbf{u} = \begin{bmatrix} {}_I\mathbf{v}_{IB} \\ {}_B\boldsymbol{\omega}_{IB} \\ \dot{\mathbf{q}}_j \end{bmatrix} \in \mathbb{R}^{n_u} \tag{2.2}$$

where ${}_I\mathbf{v}_{IB}$ is the linear velocity of the base B with respect to I, expressed in I . ${}_B\boldsymbol{\omega}_{IB}$ is the angular velocity of B with respect to I, expressed in B. The reason why an extra variable $\mathbf{u}$ is introduced for the generalized velocities instead of using $\dot{\mathbf{q}}$ is because $\int {}_B\boldsymbol{\omega}_{IB}\, dt \neq \mathbf{q}_{IB}$. The equations of motion (EOM) of the entire system can now be written as

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{h}(\mathbf{q},\mathbf{u}) = \mathbf{S}^T\boldsymbol{\tau} + \mathbf{J}_s^T\boldsymbol{\lambda} \tag{2.3}$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n_u \times n_u}$ is the generalized inertia matrix, $\mathbf{h}(\mathbf{q},\mathbf{u}) \in \mathbb{R}^{n_u}$ is the vector of Coriolis, centrifugal and gravity terms, and the selection matrix $\mathbf{S} = \begin{bmatrix} \mathbf{0}_{n_r \times (n_u - n_r)} & \mathbb{I}_{n_\tau \times n_\tau} \end{bmatrix}$ selects the n_τ degrees of freedom that are actuated by the joint torques $\boldsymbol{\tau}$. If all limb joints are actuated then $n_\tau = n_j$. The constraint forces at the end of the robot's limbs that are in contact with the environment are stacked in the vector $\boldsymbol{\lambda}$. These are related to the joint space torques through the support Jacobian $\mathbf{J}_s$. This matrix is obtained by stacking the Jacobians that relate the generalized joint velocities to each of the limbs' end-effector velocity for the limbs that are in contact with the environment. These are typically some or all of the feet and possibly the manipulator's end-effector.

2.3.2. QP formulation

The whole-body control framework used in this research is a form of hierarchical inverse dynamics where a set of operational-space and joint-space tasks is formulated after which the required joint torques are computed. This is done by using numerical optimization to find a dynamically consistent combination of generalized joint accelerations $\dot{\mathbf{u}}$, joint torques $\boldsymbol{\tau}$ and constraints forces $\boldsymbol{\lambda}$ which fulfill the specified tasks as well as possible in an instantaneous fashion.

These unknown variables are stacked into the optimization vector $\boldsymbol{\xi}$. All control tasks are formulated as linear functions of the optimization vector. This task-function formulation [43] describes the instantaneous relationship between the robot's control action space and the execution of the operational-space tasks. This can be done for both equality constraints and inequality constraints in the general form:

$$\mathbf{A}\boldsymbol{\xi} = \mathbf{b} \tag{2.4}$$

$$\mathbf{D}\boldsymbol{\xi} \leq \mathbf{f} \tag{2.5}$$

Examples of these tasks will be given in Section 2.4 and 2.5. Multiple tasks can be stacked vertically into the $\mathbf{A}$ and $\mathbf{D}$ matrices and $\mathbf{b}$ and $\mathbf{f}$ vectors, and can be weighted against each other through the weight matrices $\mathbf{Q}$ and $\mathbf{R}$. By writing the fulfillment of the equality and inequality tasks as vectors, respectively $\mathbf{w}$ and $\mathbf{v}$, a QP problem can be formulated as

$$\begin{aligned} \underset{\mathbf{w},\mathbf{v},\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}^T\,\mathbf{Q}\,\mathbf{w} + \frac{1}{2}\mathbf{v}^T\,\mathbf{R}\,\mathbf{v} \\ \text{subject to} \quad & \mathbf{w} = \mathbf{A}\boldsymbol{\xi} - \mathbf{b} \\ & \mathbf{v} \geq \mathbf{D}\boldsymbol{\xi} - \mathbf{f} \end{aligned} \tag{2.6}$$

Solving this QP problem through numerical optimization yields a solution vector $\boldsymbol{\xi}$ that fulfills the specified tasks as well as possible and contains the required joint torques. Tasks can be weighted differently with respect to each other when proper execution of some tasks are of higher importance than that of others. This is particularly relevant when control tasks conflict. Consider a situation where a robot's reaching task conflicts with the robot's center of mass (COM) position tracking task which regulates the robot's balance. To guarantee the robot's balance, it would in this situation be desirable that the execution of the COM position tracking task stays intact, even if it means that this goes at the expense of the execution of the reaching task.

2.3.3. Prioritized set of tasks

Weight matrices can be used to give more importance to good execution of some specific tasks with respect to others. In order to enforce a strict hierarchy in task priorities a different approach can be taken. Because the equality tasks are formulated as linear equations it is possible to determine the null-space of these tasks. By forcing the solution to lower priority tasks to lie within the null-space of the higher priority tasks it can be guaranteed that the lower priority tasks will get executed as well as possible without corrupting the execution of any higher priority tasks.

Assume a primary task of the form $\mathbf{A}_1\boldsymbol{\xi}_1^* = \mathbf{b}_1$. Here $\boldsymbol{\xi}_1^*$ specifies the specific solution vector that fulfills this primary task. We now specify an additional solution vector $\boldsymbol{\xi}_2^*$ which will satisfy a secondary task while leaving the execution of the first task undisturbed:

$$\begin{aligned} \mathbf{A}_2(\boldsymbol{\xi}_1^* + \boldsymbol{\xi}_2^*) &= \mathbf{b}_2 && (2.7)\\ \mathbf{A}_1\boldsymbol{\xi}_2^* &= \mathbf{0} && (2.8) \end{aligned}$$

We can combine these two requirements by requiring the additional solution $\boldsymbol{\xi}_2^*$ to live within the vector-space spanned by the null-space basis of the primary task matrix: $\mathbf{Z}_2 = \mathcal{N}(\mathbf{A}_1)$. This is done by requiring $\boldsymbol{\xi}_2^*$ to be a linear combination of the vectors that make up the null-space basis through multiplication of $\mathbf{Z}_2$ with a solution vector $\mathbf{z}_2^*$, resulting in the following expression:

$$\mathbf{A}_2\left(\boldsymbol{\xi}_1^* + \mathbf{Z}_2\mathbf{z}_2^*\right) = \mathbf{b}_2 \tag{2.9}$$

Here $\boldsymbol{\xi}_1^*$ is already known from solving for the primary task. When a vector $\mathbf{z}_2^*$ is found that satisfies this equation, the additional solution vector $\boldsymbol{\xi}_2^*$ for the secondary task is computed as:

$$\boldsymbol{\xi}_2^* = \mathbf{Z}_2\mathbf{z}_2^* \tag{2.10}$$

This additional solution will result in an execution of the secondary task that is as good as possible without disturbing the execution of the primary task. The final solution vector for both tasks can then be computed as:

$$\boldsymbol{\xi}_2 = \boldsymbol{\xi}_1^* + \boldsymbol{\xi}_2^* \tag{2.11}$$

This approach can be generalized to a set of tasks with i hierarchy levels such that we can compute the additional solution $\boldsymbol{\xi}_i^*$ for the task at priority level i:

$$\begin{aligned} \mathbf{A}_i(\boldsymbol{\xi}_1^* + \cdots + \boldsymbol{\xi}_i^*) &= \mathbf{b}_i && (2.12)\\ \mathbf{A}_k\boldsymbol{\xi}_i^* &= \mathbf{0}\ (\forall\ k < i) && (2.13)\\ \mathbf{A}_i(\boldsymbol{\xi}_1^* + \cdots + \boldsymbol{\xi}_{i-1}^* + \mathbf{Z}_i\mathbf{z}_i^*) &= \mathbf{b}_i && (2.14)\\ \mathbf{A}_i(\boldsymbol{\xi}_{i-1} + \mathbf{Z}_i\mathbf{z}_i^*) &= \mathbf{b}_i && (2.15)\\ \boldsymbol{\xi}_i^* &= \mathbf{Z}_i\mathbf{z}_i^* && (2.16)\\ & where: && \\ \mathbf{Z}_i &= \mathcal{N}\left(\underline{\mathbf{A}}_i\right) && (2.17)\\ \mathbf{Z}_1 &= \mathbf{I} && (2.18) \end{aligned}$$

Here $\underline{\mathbf{A}}_i$ is the augmented task matrix which is stacked with all previous priority task matrices: $\begin{bmatrix}\mathbf{A}_1^T & \cdots & \mathbf{A}_{i-1}^T\end{bmatrix}^T$. By recursively computing the solutions $\boldsymbol{\xi}^*$ out of (2.15) and (2.16) for hierarchy level 1 till i, the complete solution $\boldsymbol{\xi}_i$ for an entire hierarchy of i tasks is obtained by adding all individual solutions: $\boldsymbol{\xi}_i = \boldsymbol{\xi}_1^* + \cdots + \boldsymbol{\xi}_i^*$. In order to solve (2.15) for one level of the hierarchy it can be formulated as a QP problem as:

$$\begin{aligned} \underset{\mathbf{w}_i, \mathbf{z}_i^*}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}_i^T\,\mathbf{Q}\,\mathbf{w}_i \\ \text{subject to} \quad & \mathbf{w} = \mathbf{A}_i\left(\boldsymbol{\xi}_{i-1} + \mathbf{Z}_i\mathbf{z}_i^*\right) - \mathbf{b}_i \end{aligned} \tag{2.19}$$

The null-space basis $\mathbf{Z}_i$ can be computed using a QR or SVD decomposition of the augmented task matrix $\underline{\mathbf{A}}_i$. Here the QR decomposition is used because it is less computationally intensive. In [10] it is shown that

because the complexity of a QR decomposition grows with the cube of the rows of its argument, it is computationally lighter to compute the null-space basis for each hierarchy level recursively. This is done based on the previous priority level's null-space basis $\mathbf{Z}_{i-1}$ and the previous priority level task matrix $\mathbf{A}_{i-1}$

$$\mathbf{Z}_i = \mathbf{Z}_{i-1} \mathcal{N}\left(\mathbf{A}_{i-1}\mathbf{Z}_{i-1}\right) \tag{2.20}$$

The number of columns of $\mathbf{Z}_i$ and the length of the solution vector $\mathbf{z}_i^*$ is equal to the nullity of the augmented task matrix $\underline{\mathbf{A}}_i$. Because the size of the null-space of $\underline{\mathbf{A}}_i$ decreases with every added priority level, the size of optimization problem, which is required to live in this null-space, also decreases.

So far the optimization problem formulation only takes into account equality constraints. Because of the projection into the null-space of the higher priority tasks, this approach results in a cascade of increasingly smaller QP problems. Unfortunately the inequality constraints cannot benefit from a similar approach and have to be explicitly added to every QP problem. This means that for every priority level the inequality constraints of that level and of all higher priority levels have to be taken into account. This results in the following general formulation of the QP problem for priority level i:

$$\begin{aligned}
\underset{\mathbf{w}_i,\mathbf{v}_i,\mathbf{z}_i^*}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}_i^T\,\mathbf{Q}\,\mathbf{w}_i + \frac{1}{2}\mathbf{v}_i^T\,\mathbf{R}\,\mathbf{v}_i \\
\text{subject to} \quad & \mathbf{w}_i = \mathbf{A}_i\left(\boldsymbol{\xi}_{i-1} + \mathbf{Z}_i\mathbf{z}_i^*\right) - \mathbf{b}_i \\
& \mathbf{v}_i \geq \mathbf{D}_i\left(\boldsymbol{\xi}_{i-1} + \mathbf{Z}_i\mathbf{z}_i^*\right) - \mathbf{f}_i \\
& \mathbf{v}_{i-1} \geq \mathbf{D}_{i-1}\left(\boldsymbol{\xi}_{i-1} + \mathbf{Z}_i\mathbf{z}_i^*\right) - \mathbf{f}_{i-1} \\
& \vdots \\
& \mathbf{v}_1 \geq \mathbf{D}_1\left(\boldsymbol{\xi}_{i-1} + \mathbf{Z}_i\mathbf{z}_i^*\right) - \mathbf{f}_1
\end{aligned} \tag{2.21}$$

The solution for i priorities is computed by recursively solving (2.21) for each priority level and using $\boldsymbol{\xi}_i^* = \mathbf{Z}_i\mathbf{z}_i^*$ to compute the final solution as

$$\boldsymbol{\xi}_i = \sum_{k=1}^{i} \boldsymbol{\xi}_k^* \tag{2.22}$$

2.3.4. Dynamic consistency

In order to guarantee that the final solution for the set of control tasks is dynamically feasible, the EOM can be incorporated as a top-priority task. When assuming that all dynamic variables are in the optimization vector such that $\boldsymbol{\xi} = \begin{bmatrix}\dot{\mathbf{u}} & \boldsymbol{\tau} & \boldsymbol{\lambda}\end{bmatrix}^T$, the EOM task can be written in the general task form $\mathbf{A}\boldsymbol{\xi} = \mathbf{b}$ as:

$$\begin{bmatrix}\mathbf{M} & -\mathbf{S}^T & \mathbf{J}_s^T\end{bmatrix}\boldsymbol{\xi} = -\mathbf{h} \tag{2.23}$$

This task by itself has an infinite number of solutions. However its strength lies in the fact that the solutions of all lower priority tasks then have to be consistent with this EOM task and therefore be dynamically feasible. Motion tasks can now simply be kinematically formulated as a function of the generalized accelerations $\dot{\mathbf{u}}$, because compliance with the EOM task means that the corresponding joint torques will also be found during the optimization for the motion task. Furthermore, the compliance with the highest priority EOM task also guarantees that different motion tasks do not dynamically interfere with each other, even though they are only specified on a kinematic level.

In order to reduce the computational burden of the optimization problem several reductions of the problem have been proposed. A very considerable reduction can be obtained by decomposing the EOM into a part related to the robot's floating base (fb) and a part related to the robot's actuated joints (j) [22]:

$$\mathbf{M}_{fb}\dot{\mathbf{u}} + \mathbf{h}_{fb} = \mathbf{J}_{s,fb}^T\boldsymbol{\lambda} \tag{2.24}$$

$$\mathbf{M}_j\dot{\mathbf{u}} + \mathbf{h}_j = \boldsymbol{\tau} + \mathbf{J}_{s,j}^T\boldsymbol{\lambda} \tag{2.25}$$

Equation (2.24) is obtained by just taking the first 6 equations of the EOM, which correspond to the floating base, and can be interpreted as the Newton-Euler equations of the whole system. It relates the change of

momentum of the system to the external forces. The interesting feature is that the joint torques $\boldsymbol{\tau}$ do not appear in equation (2.24). By rewriting equation (2.25) it can be seen that the joint torques can be expressed as a linear function of the generalized accelerations and contact forces:

$$\boldsymbol{\tau} = \mathbf{M}_j \dot{\mathbf{u}} + \mathbf{h}_j - \mathbf{J}_{s,j}^T \boldsymbol{\lambda} \tag{2.26}$$

This decomposition can be exploited by removing the joint torques from the optimization vector, thereby reducing it to $\boldsymbol{\xi} = \begin{bmatrix} \dot{\mathbf{u}} & \boldsymbol{\lambda} \end{bmatrix}^T$. The EOM control task can then be reduced to equation (2.24). This means that any other control tasks also need to be specified as a linear function of only the generalized accelerations $\dot{\mathbf{u}}$ and contact forces $\boldsymbol{\lambda}$. Control tasks as a direct function of the joint torques $\boldsymbol{\tau}$ can be written as a function of the new reduced optimization vector $\boldsymbol{\xi} = \begin{bmatrix} \dot{\mathbf{u}} & \boldsymbol{\lambda} \end{bmatrix}^T$, by using the relationship from (2.26).

2.4. Equality tasks

Using the described control framework, control tasks can be expressed as a function of the generalized accelerations $\dot{\mathbf{u}}$, joint torques $\boldsymbol{\tau}$, and contact forces $\boldsymbol{\lambda}$. In the previous section the EOM-based equality task which is based on all three of these variables was presented. In this section the most common usages of equality tasks for these three variables individually will be presented.

2.4.1. Motion tasks

The incorporation of the EOM task allows for an easy specification of inverse dynamics motion tasks. Consider a function $\mathbf{f}(\mathbf{q})$ that relates the robot's generalized coordinates to for example the operational-space pose $\mathbf{r}$ of the manipulator's end-effector in $SE(3)$. By computing the partial derivatives of $\mathbf{f}(\mathbf{q})$ with respect to the generalized coordinates $\mathbf{q}$ the Jacobian matrix $\mathbf{J}(\mathbf{q})$ is constructed. This matrix represents a configuration-dependent, linearized relationship between the robot's generalized velocities $\mathbf{u}$ and the end-effector's linear and angular velocities in operational-space $\dot{\mathbf{r}}$. By taking the time derivative of this relationship, an expression is acquired that describes the relationship between the robot's generalized accelerations $\dot{\mathbf{u}}$ and end-effector accelerations $\ddot{\mathbf{r}}$.

$$\mathbf{r} = \mathbf{f}(\mathbf{q}) \tag{2.27}$$

$$\dot{\mathbf{r}} = \mathbf{J}(\mathbf{q})\mathbf{u} \tag{2.28}$$

$$\ddot{\mathbf{r}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{u}} + \dot{\mathbf{J}}(\mathbf{q})\mathbf{u} \tag{2.29}$$

$$\tag{2.30}$$

Assuming the reduced optimization vector $\boldsymbol{\xi} = \begin{bmatrix} \dot{\mathbf{u}} & \boldsymbol{\lambda} \end{bmatrix}^T$, equation (2.29) can be rewritten as a function of the generalized accelerations to arrive at the following general formulation for a motion control task:

$$\begin{bmatrix} \mathbf{J}(\mathbf{q}) & \mathbf{0} \end{bmatrix} \boldsymbol{\xi} = \ddot{\mathbf{r}}_{des} - \dot{\mathbf{J}}(\mathbf{q})\mathbf{u} \tag{2.31}$$

The desired operational-space accelerations $\ddot{\mathbf{r}}$ are typically computed according to a PD-law that can be interpreted as a virtual spring-damper between the actual and the desired pose:

$$\ddot{\mathbf{r}}_{des} = \ddot{\mathbf{r}}^{FF} + \mathbf{k}_d \left(\dot{\mathbf{r}}_{des} - \dot{\mathbf{r}} \right) + \mathbf{k}_p \left(\mathbf{r}_{des} \boxminus \mathbf{r} \right) \tag{2.32}$$

where the desired end-effector accelerations are computed based on an optional feed-forward term $\ddot{\mathbf{r}}^{FF}$, and the errors between the current end-effector velocity and position, and corresponding desired values. Note the box minus operator ($\boxminus$) used to calculate the pose error. This operator is required when it is desired to control orientation and not just position. Because orientations form a Lie group, namely $SO(3)$, there is no vector-space in which to perform addition or subtraction. Instead, a special subtraction operator $\boxminus$ has to be defined, which will make it possible to specify a difference magnitude between two orientations [13].

The method described here can be used for the control of any point or frame on the robot for which the position and/or orientation can be expressed as a function of the generalized coordinates of the robot. It can therefore also be used for the control of the motion of the robot's center of mass.

2.4.2. Contact force tasks

For the described torque-controllable legged manipulator it is desirable to control the contact forces when the limbs are in contact with the environment. This can be the linear contact forces at the robot's point-feet or the linear and torsional contact forces at the manipulator's end-effector. By actively controlling these forces it is for example possible to distribute the robot's contact forces evenly over its feet, or to achieve force-based manipulation of the environment.

Control tasks for the contact forces can be easily designed because these forces ($\boldsymbol{\lambda}$) appear directly in the optimization vector $\boldsymbol{\xi}$. An equality task for desired values of certain contact forces can then be simply written as

$$\begin{bmatrix}\mathbf{0} & \mathbf{S}\end{bmatrix}\boldsymbol{\xi} = \boldsymbol{\lambda}_{des} \tag{2.33}$$

where $\mathbf{S}$ is a selection matrix that selects the contact forces that are desired to be controlled, and the vector $\boldsymbol{\lambda}_{des}$ contains the desired values for those forces.

This can for example be used to specify desired contact forces at the manipulator's end-effector for a force-based manipulation task. However, because the contact forces are directly related to the robot's generalized accelerations $\dot{\mathbf{u}}$ through the reduced equations of motion (2.24), it is common to not prescribe equality tasks for the robot's limbs unless it is strictly necessary such as for a force-based manipulation tasks. Instead it is common to mostly specify motion-based equality tasks while the required contact forces follow automatically from the compliance with the EOM task. To keep these automatically computed contact forces within physically feasible bounds, inequality tasks can be designed that describe the limited nature of the contact. This will be described in Section 2.5.2.

2.4.3. Joint torque tasks

The robot's joint torques $\boldsymbol{\tau}$ are usually left to be computed as a result of the motion and contact force tasks through compliance with the EOM task. However, it could sometimes still be desirable to directly specify a desired value $\boldsymbol{\tau}_{des}$ for some of the robot's joint torques. Because the joint torques do no longer appear in the reduced version of the optimization vector $\boldsymbol{\xi}$, the equality task has to be expressed as a function of the generalized accelerations $\dot{\mathbf{u}}$ and contact forces $\boldsymbol{\lambda}$ according to (2.26). The equality task can then be formulated similar to (2.33) by including a selection matrix $\mathbf{S}$ that selects the desired joints:

$$\mathbf{S}\begin{bmatrix}\mathbf{M}_j & \mathbf{J}_{s,j}^T\end{bmatrix}\boldsymbol{\xi} = \boldsymbol{\tau}_{des} - \mathbf{S}\mathbf{h}_j \tag{2.34}$$

2.5. Inequality tasks

One of the main reasons for choosing optimization-based inverse dynamics controllers over analytical approaches is that numerical optimization allows for the explicit inclusion of inequality tasks. These are particularly useful in order to let the controller take into account the robot's physical limits. Examples of these limits are kinematic joint limits, inequality constraints on the contact forces, and actuator torque limits. The following subsections will present how inequality tasks based on the robot's generalized accelerations $\dot{\mathbf{u}}$, contact forces $\boldsymbol{\lambda}$, and joint torques $\boldsymbol{\tau}$ can be formulated.

2.5.1. Inequality tasks - Motion

The motion tracking equality tasks described in Section 2.4.1 implicitly assume limitless joints and unconstrained operational-space motion. In reality kinematic joint limits need to be considered and operational-space collision-avoidance can be desirable. When kinematic limits are desirable to be taken into account by the controller, they can be written as inequality tasks of the form $\mathbf{D}\boldsymbol{\xi} \leq \mathbf{f}$.

Kinematic limits are typically present at the position-level. However, to incorporate these limits into the control framework they need to be expressed at acceleration-level. This can be done using a Taylor expansion. Consider the simple example of specifying kinematic limits, ϕ_{min} and ϕ_{max}, on the angle ϕ of one single joint. In this case the related control action is the joint acceleration $\ddot{\phi}$, which is part of the generalized accelerations

$\dot{\mathbf{u}}$. It is required that the joint acceleration $\ddot{\phi}(t)$ at the current time step (t) is chosen such that it will not result in violation of the joint limits by the next time step $(t+\delta t)$. The joint angle $\phi(t+\delta t)$ at the next time step as a function of the joint acceleration can be estimated as follows by a Taylor expansion:

$$\phi(t+\delta t) = \phi(t) + \dot{\phi}(t)\delta t + \ddot{\phi}(t)\frac{\delta t^2}{2} \tag{2.35}$$

Here $\phi(t)$ and $\dot{\phi}(t)$ are known from the current sensor measurements. $\ddot{\phi}(t+\delta t)$ is the control variable that has to be chosen such that $\phi_{min} \leq \phi(t+\delta t) \leq \phi_{max}$. This results in the following inequality task formulations as a function of the robot's generalized accelerations:

$$\frac{\delta t^2}{2}\begin{bmatrix}\mathbf{S} & \mathbf{0}\end{bmatrix}\boldsymbol{\xi} \quad \leq \quad \phi_{max} - \phi - \delta t\dot{\phi} \tag{2.36}$$

$$-\frac{\delta t^2}{2}\begin{bmatrix}\mathbf{S} & \mathbf{0}\end{bmatrix}\boldsymbol{\xi} \quad \leq \quad -\phi_{min} + \phi + \delta t\dot{\phi} \tag{2.37}$$

Here $\mathbf{S}$ is a selection vector that selects the generalized acceleration vector entry which corresponds to the appropriate joint(s). Theoretically δt should be equal to the control period, but in practice the value of δt can be increased to obtain smoother behavior near the joint limits. When approaching a joint limit, this approach will result in an earlier activation of the constraint, which will in turn lead to a lower required limit-avoiding control action. This can help prevent issues where the controller suddenly generates large accelerations in order to make sure that the inequality constraint is not violated by the next control step.

We can generalize the avoidance of kinematic limits to enforce operational-space motion limits for any physical or abstract point $\mathbf{r}$ on the robot by incorporating an associated Jacobian $\mathbf{J}$ such as used for motion tracking equality tasks (Section 2.4.1). The inequality tasks are then formulated as:

$$\frac{\delta t^2}{2}\begin{bmatrix}\mathbf{J} & \mathbf{0}\end{bmatrix}\boldsymbol{\xi} \quad \leq \quad (\mathbf{r}_{max} - \mathbf{r}(t)) - \left(\mathbf{J} + \dot{\mathbf{J}}\frac{\delta t}{2}\right)\delta t\mathbf{u} \tag{2.38}$$

$$-\frac{\delta t^2}{2}\begin{bmatrix}\mathbf{J} & \mathbf{0}\end{bmatrix}\boldsymbol{\xi} \quad \leq \quad (\mathbf{r}(t) - \mathbf{r}_{min}) + \left(\mathbf{J} + \dot{\mathbf{J}}\frac{\delta t}{2}\right)\delta t\mathbf{u} \tag{2.39}$$

These inequality tasks can be used for example to prevent a point on the robot from entering a certain part of the operational-space; to set limits on the robot's COM position; or to prevent an end-effector to move too far or too close with respect to the robot's torso.

2.5.2. Inequality tasks - Contact forces

Inequality tasks can also be formulated for the robot's contact forces $\boldsymbol{\lambda}$. The most common usage for this is to describe the physically feasible bounds of the contact forces at the feet. As mentioned in Section 2.4.2, it is common not to specify any contact force equality tasks for most or all of the robot's limbs - especially the feet - that are in contact with the environment. Instead these contact forces then follow from other control tasks, such as motion tasks, that are related to the contact forces through compliance with the higher priority EOM task. To guarantee that the computed contact forces at the point-feet are physically feasible, corresponding inequality tasks can be designed which describe the unilateral and friction-dependent nature of the contact.

Typically a Coulomb friction model is assumed, which asserts that in order to prevent slipping the maximum allowed tangential component of a contact force is proportional to the normal component. The theoretical contact force vectors that have this maximum allowable ratio between their tangential and normal components form a cone shape, the so-called friction cone [33]. This means that in order to prevent slippage it is necessary to control the direction of the contact force vectors to lie within their corresponding friction cone. The unidirectional nature of these friction cones also takes into account the constraint that the feet can only push, and not pull, on their contact point. The requirement for the contact forces to lie within their friction cones can be expressed as inequality constraints on the contact forces.

For this purpose the friction cone is typically approximated with a friction pyramid which allows to formulate linear constraint tasks. Consider a friction pyramid that is oriented such that two of the four sloped sides

are parallel to the frontal direction of the robot, with the angle of the sides corresponding to the friction coefficient μ. The contact force vector $\boldsymbol{\lambda}$ can then be decomposed along the unit vector $\hat{\mathbf{n}}$ in the normal direction and the tangential unit vectors along the frontal $\hat{\mathbf{f}}$ and lateral $\hat{\mathbf{l}}$ directions. In order to keep the contact force vectors inside the friction pyramid, the following inequality constraints can be formulated as a function of each of the individual the contact force vectors $\boldsymbol{\lambda}_i$ in the optimization vector $\boldsymbol{\xi}$ as:

$$(\hat{\mathbf{f}} - \hat{\mathbf{n}}\mu)^T \boldsymbol{\lambda}_i \leq 0 \tag{2.40}$$
$$-(\hat{\mathbf{f}} + \hat{\mathbf{n}}\mu)^T \boldsymbol{\lambda}_i \leq 0 \tag{2.41}$$
$$(\hat{\mathbf{l}} - \hat{\mathbf{n}}\mu)^T \boldsymbol{\lambda}_i \leq 0 \tag{2.42}$$
$$-(\hat{\mathbf{l}} + \hat{\mathbf{n}}\mu)^T \boldsymbol{\lambda}_i \leq 0 \tag{2.43}$$

2.5.3. Inequality tasks - Joint torques

The most common application of inequality tasks for the joint torques is to design tasks that guarantee that the computed torques can actually be achieved by the robot's actuators. Similar to the discussed approach for the torque equality tasks (Section 2.4.3), the control task will need to be specified as a function of the generalized accelerations $\dot{\mathbf{u}}$ and contact forces $\boldsymbol{\lambda}$. Inequality tasks that enforce compliance with the actuator's torque limits can be formulated as:

$$\begin{bmatrix} \mathbf{M}_j & \mathbf{J}_{s,j}^T \end{bmatrix} \boldsymbol{\xi} \leq \boldsymbol{\tau}_{max} - \mathbf{h}_j \tag{2.44}$$

where $\boldsymbol{\tau}_{max}$ is a vector containing the maximum torque values for each actuated joint.

2.6. Research focus and problem statement

This work is focused on the implementation of the described hierarchical inverse dynamics control framework for the control of a quadrupedal manipulator, and presents several specific implementation approaches that exploit the benefits of both the control framework and the robotic platform.

The focus in this work is on manipulation. A quadrupedal manipulator can be expected to perform two types of manipulation. First, typical manipulation tasks consist of controlling the end-effector's motion, for reaching, picking and placing tasks. The robot should for example be able to bring its end-effector to the floor or high shelf for object grasping, and be able to move its end-effector or a tool such as a camera or drill to a desired position and orientation. The second category of manipulation is that of force-based manipulation tasks, where instead of motion it is required to control the interaction force at the end-effector. An example is the opening of a heavy door where it is important to control the force that the robot exerts on the door, while the position of the end-effector is allowed to passively follow the constrained motion of the door. Grasping is not explicitly addressed in this work, although proper control of end-effector position and orientation already forms an important aspect of grasping. Finally, to restrict the complexity, the scope of this work is limited to stationary manipulation, thereby not taking into account the robot's ability for locomotion.

The control of the robot has to meet several requirements. For the application of hierarchical inverse dynamics on a realistic legged platform standard requirements will have to be met, such as compliance with the robot's equations of motion, and respecting of the robot's physical limitations. Besides these requirements, several additional requirements where identified to be of special interest for the control of a quadrupedal manipulator:

1. End-effector pose control
2. End-effector contact force control
3. Maintaining balance (under external disturbances)
4. Exploiting whole-body potential
5. Avoiding/handling singularities
6. Collision avoidance

7. Simple real-time teleoperation

1. End-effector pose control is the control of the position and orientation of the end-effector for effective manipulation of the environment. **2.** End-effector contact force control is the direct control of the contact forces between the end-effector and the environment such that force-based manipulation tasks can be executed. **3.** Maintaining balance (under external disturbances) entails that the robot's COM motion needs to be controlled in such a way that the robot does not fall over, even when external forces are exerted on the robot. **4.** Exploiting whole-body potential refers to behavior in which motion of all joints that could theoretically aid in the execution of a task are used. **5.** Avoiding/handling singularities refers to the kinematically singular configuration in which the robot can end up and around which unstable or uncontrollable behavior can emerge. **6.** Collision avoidance refers to preventing parts of the robot to collide with each other as a result of the controller output. **7.** Simple real-time controllability refers to the requirement for a user to be able to intuitively remotely control the robot in real-time.

Furthermore, the morphology of a quadrupedal manipulator offers specific advantages. When all four feet are in contact with the ground its large base of support allows for a large region of allowable horizontal motion of the robot's COM while still maintaining balance. Besides that, it is also possible to exert significant external forces on the robot without directly jeopardizing balance. Manipulators that are fixed on a wide wheeled base have similar advantages. However, additionally the legged morphology of a quadrupedal manipulator also allows the robot to move the manipulator's base (the robot's torso) while keeping its contact points with the ground fixed. These properties create opportunities for robust whole-body manipulation. Finally, the height and orientation of the robot's torso can easily be adapted to aid a manipulation task without changing the position of the COM of the robot, thereby not influencing the robot's balance.

The described control framework also offers several benefits. It allows for the strict prioritization of control tasks, ensuring that lower priority tasks will be fulfilled as well as possible without disturbing the execution of higher priority tasks. Besides that, the dependency on a numerical optimization algorithm allows for the explicit integration of various types of inequality tasks.

The objective of the research presented in this thesis is

The implementation of an optimization-based hierarchical inverse dynamics control framework for the control of a simulated quadrupedal manipulator during stationary manipulation, with a specific focus on the design of prioritized sets of control tasks which allow to exploit the characteristic benefits offered by both the control framework and robotic platform.

Effectiveness of these controller implementations are measured by their ability to tackle the control challenges listed above. As a result four promising sets of prioritized tasks were identified:

- A basic task set, for end-effector motion control while maintaining balance.
- Addition of kinematic limits to trigger whole-body reaching
- Optimizing robot posture according to a desired posture measure
- Actively controlling end-effector contact forces while maintaining multi-contact balance

These implementations will be presented in the following chapters. First, Chapter 3 will present the basic implementation which allows for pose control of the end-effector while keeping the robot balanced. Chapter 4 will show how an extension of the basic implementation with kinematic limit tasks can result in an increase of workspace reachability by triggering whole-body reaching. In Chapter 5 a method is proposed that takes advantage of the robot's base motion in order to optimize the manipulator arm's posture according to a desired measure. Chapter 6 presents an approach for directly controlling contact forces at the end-effector for force-based manipulation, while the remaining required external forces are distributed over the feet. Finally, in Chapter 7, a sensor-based balancing controller is proposed which adapts the robot's posture when large external forces are exerted on the robot through for example manipulation. These chapters all contain an individual Results and Discussion section.

3

Basic Implementation

3.1. Introduction

In this chapter a basic set of control tasks for the hierarchical inverse dynamics control framework is presented. This task setup will ensure that the most important control requirements are met, such as end-effector pose control and maintaining balance. The control task hierarchy presented and discussed here will function as the basis for the more advanced sets of control tasks which are discussed in the following chapters.

3.2. Controller design

The basic control task setup presented here is based on task setups that have previously been successfully implemented on the ANYmal robot without a manipulator arm, and depends on the same numerical hierarchical optimization algorithm that was used for this previous research [10][11]. Figure 3.1 shows this basic task hierarchy setup which is ordered from high priority (red) tasks to low priority (green) tasks. The highest control task priority level contains the control tasks which are most crucial to be correctly fulfilled, such as the EOM task and the inequality tasks that represent the robot's physical limits. The second priority level consists of motion tasks which should be executed as well as possible as long as they comply with the execution of the high priority tasks. Finally a optimization task is added which removes the robot's leftover torque-redundancy after all other tasks have been fulfilled. Although it is possible to give each control task an individual priority level, it is chosen to group the control tasks in only three priority levels. This reduction of the number of priority levels in the QP cascade leads to lower computational costs. The highest priority tasks are grouped together because they are all of very high importance, but are also expected to be simultaneously feasible at all times.

This task setup shown in Figure 3.1 is similar to task setups previously successfully used for the control of the ANYmal robot without arm, but includes the addition of a pose tracking task for the arm's end-effector for motion control and of a contact force minimization task. This final task is added to remove any torque-redundancy left in the system after solving the higher priority tasks such that there is only one optimal solution to the set of control tasks. A brief description of each of these tasks is provided here:

Equations of motion
This is the task described in Section 2.3.4 and will guarantee dynamic feasibility of the other tasks.

Friction cones
This inequality task ensures that the computed contact forces at the feet lie within the friction cones which describe the unilateral and friction-dependent nature of the contacts (Section 2.5.2).

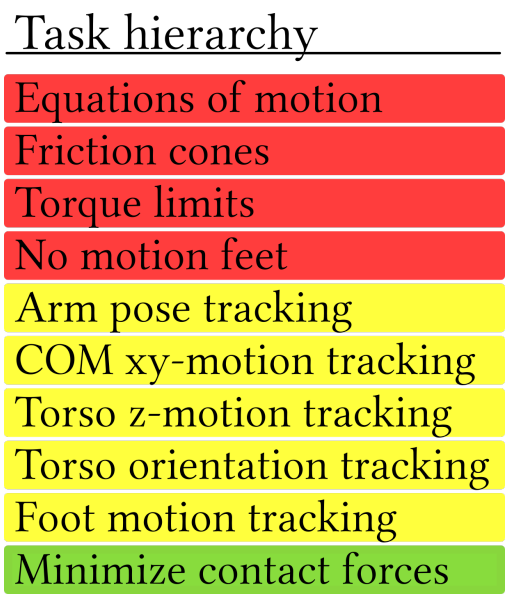


Figure 3.1: The basic task hierarchy for the quadrupedal manipulator. Colors indicate the priority levels, ranging from top priority (red) to lowest priority (green). This task hierarchy allows for inverse dynamics motion control of the robot's torso and limbs while being compliant with the robot's EOM and while respecting the robot's physical limits.

Torque limits
In order to guarantee that all computed torques actually fall within the actuator's physical torque limits this inequality task is added (see Section 2.5.3).

No motion feet
This equality motion task prescribes zero motion for the feet that are in contact with the ground. This is done similar to (2.31) with $\ddot{\mathbf{r}}_{des} = 0$ such that the task is formulated as $\begin{bmatrix}\mathbf{J}(\mathbf{q}) & \mathbf{0}\end{bmatrix}\boldsymbol{\xi} = -\dot{\mathbf{J}}(\mathbf{q})\mathbf{u}$.

Arm pose tracking
The motion tracking equality task for the position and orientation tracking of the end-effector at the end of the 6 DOF manipulator arm.

COM xy-motion tracking
To control the balance of the robot this motion tracking task controls the horizontal position of the robot's COM. Because the COM position is dependent on the entire robot's configuration, this task will use all (available) kinematic degrees of freedom.

Torso z-motion tracking
The torso's horizontal motion is deliberately left uncontrolled so it is available to the COM motion task. Because the COM height is not relevant for balance an explicit height tracking task for the robot's torso is added to give direct control over the torso's height. This task is easily designed because the torso pose is chosen as the generalized floating base coordinates, which means that the torso's vertical acceleration is part of the optimization vector $\boldsymbol{\xi}$.

Torso orientation tracking
This equality task controls the torso's angular motion. This task can also be simply formulated as a function of the generalized coordinates which contain the torso's orientation as floating base coordinates.

Foot motion tracking
The motion tracking task which is responsible for the motion tracking of the swing foot/feet along motion trajectories which are supplied by a higher-level controller. Because locomotion is not considered in this thesis, this task is not used in any of the provided examples. It is however still included in the presented task setups for completeness.

Minimize contact forces
This final equality task exploits the leftover torque-redundancy in the system to minimize the robot's contact forces. This is done by adding a task that prescribes the contact forces to be equal to zero. Because of the QP formulation of the control tasks, this will result in the computation of a set of contact forces with a minimum norm value, which are also compliant with the correct execution of the higher priority tasks. Therefore, unnecessary tangential components of the contact forces will be absent as much as possible, while the normal components will be equally distributed over the feet. Without this final task an infinite amount of contact force distributions are possible while all higher priority tasks are still fulfilled. This can result in the optimization algorithm computing significantly different solutions for the contact forces between subsequent control periods, generating undesirable jumps in the robot's commanded joint torques.

The motion tasks are chosen such that they cover exactly the 24 kinematic DOFs of the robot, which are related to the joints (18 DOF) and the floating base (6 DOF). The fact that the robot has exactly as many DOFs available as the motion tasks require means that the motion tasks can be independently fulfilled. It is therefore possible to for example control the position and orientation of the torso without interfering with the position and orientation of the end-effector and vice versa. The presence of the high-priority EOM task guarantees that even disturbances which result from dynamic effects due to fast motions will be compensated for such that the executions of the tasks do not disturb each other.

3.3. Results

A simple demonstration of the effectiveness of this task hierarchy is given by inputting a motion reference trajectory that commands the end-effector to move laterally from side to side while keeping the orientation constant. The reference position for the robot's horizontal COM position is also kept constant at the middle of the support polygon formed by the feet in order to retain balance. This results in a motion of the end-effector in the desired motion direction, while the robot's torso will move in the opposite direction in order to also fulfill the COM position task (Figure 3.2).

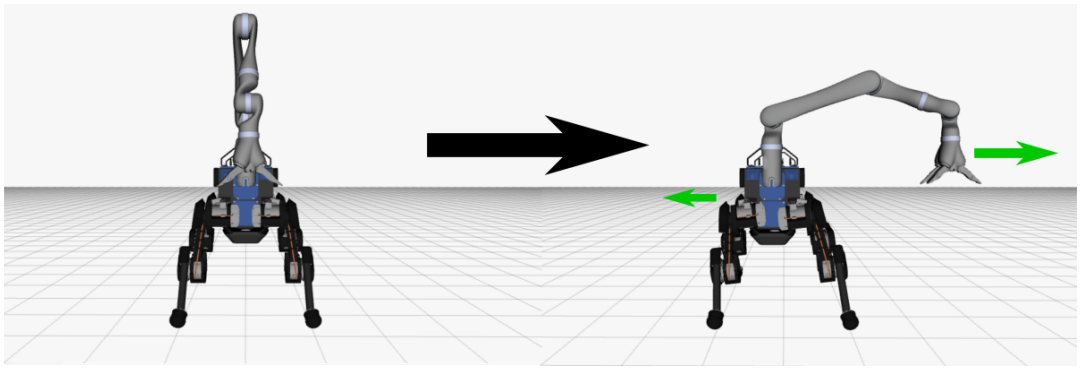


Figure 3.2: When the end-effector is commanded to move laterally away from a neutral position in front of the robot, the robot's torso will move in the opposite direction so that the COM remains at the same position.

Figure 3.3 shows the results for this experiment. The end-effector is gradually commanded to move from a neutral position in front of the robot to a position 30 cm to one side, and from there to a new position 30 cm on the other side. It can be clearly seen that this results in lateral motion of the torso opposite and proportional to the end-effector motion. Because of this the COM remains at its desired constant position which ensures the robot's balance.

3.4. Discussion

The basic task hierarchy presented in this Chapter already meets three of the requirements listed in Section 2.6. It allows pose tracking of the end-effector. It also actively generates motion such that the robot's COM remains at a desired position, thereby keeping the robot balanced. Finally, it allows simple real-time teleoperation. The only required control input is the desired direction for translation or rotation of the end-effector.

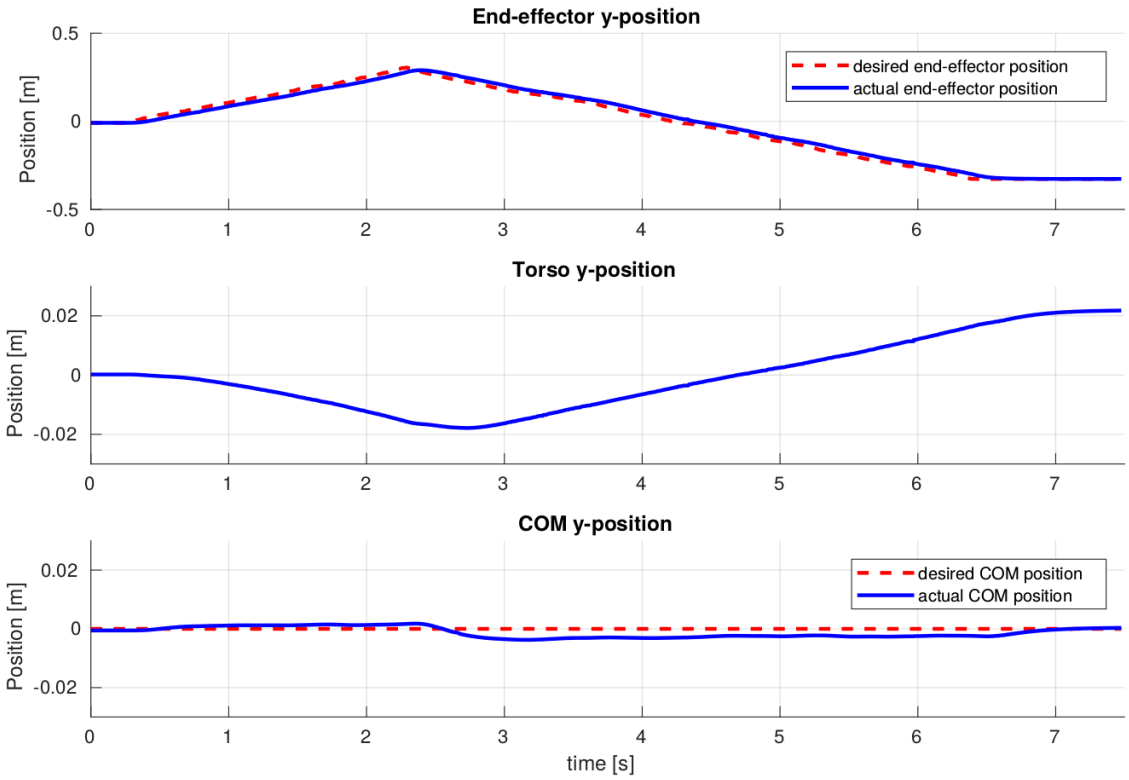


Figure 3.3: While lateral motion is commanded for the end-effector (top plot), the COM position remains constant (bottom plot) through motion of the robot's torso (middle plot).

Based on this reference input the end-effector will track the desired motion while the robot also uses all joints to actively keep the COM at its desired position.

The requirement of exploiting the robot's whole-body potential is not yet considered to be met. When the end-effector's reference position is commanded too far away from the base this will result in a full extension of the elbow joint. This is a well-known problematic kinematically singular configuration of the robot. Around the extended elbow configuration the shoulder flexion/extension joint and the elbow joint have a similar effect on the end-effector motion, which means that the arm (nearly) loses an independent DOF. This can be seen when inspecting the singular values of the motion task Jacobian for the end-effector. Near the extended elbow configuration one of the singular values will approach zero. This results in the controller computing very high joint accelerations in order to still utilize the near-uncontrollable DOF. Therefore the robot will display undesirable jerky and unstable motion around this configuration.

An intuitive theoretical solution would be to use the DOFs of the base to accommodate the end-effector motion task. This could be done by non-balance-disturbing rotation of the base, or even by sacrificing some balance margin through translation of the base. However, even when placing the end-effector motion tracking task at a higher priority than the base motion tracking tasks, this behavior will not occur. This is the case because to the optimization algorithm it appears that all 6 DOFs of the arm are still available for the end-effector pose tracking task, meaning that the DOFs of the floating base can be used for other control tasks. The fact that one of the DOFs of the arm is not desirable to be used because of its (near-)singular configuration is not taken into account.

This means that for this task hierarchy the robot's torso will move in order to fulfill the base height and orientation tasks, and the COM position task, but will never contribute directly to the end-effector motion task because the controller will always attempt to perform that task with the 6 DOFs of the arm regardless of the arm's configuration. Because of this, commanded motion of the end-effector will result in coordinated whole-body motion, but the full potential contribution of all DOFs for the end-effector motion task is not be-

ing achieved. The workspace reachability of the end-effector could be significantly enlarged if the base DOFs would support the end-effector motion when required.

4

Whole-body reaching

4.1. Introduction

The previous chapter showed a basic implementation of a task hierarchy that allowed independent motion control of the robot's base and the end-points of all five limbs. However, important limitations were found to lie in undesirable controller solutions at an extended elbow configuration, and in absence of base motion to increase end-effector workspace reachability.

This chapter shows how the basic controller implementation can be extended by the addition of kinematic limits in the form of inequality tasks. When added to the task hierarchy in the right way several of the mentioned limitations of the basic controller implementation (Chapter 3) can be overcome by this one adaptation. As a result, elegant whole-body motion can emerge which allows the robot's end-effector to reach otherwise hard-to-reach places. This allows for controlling only the end-effector motion without having to take into account the elbow singularity or the whole-body motion that is required to reach difficult positions such as a high shelf or an object on the floor.

4.2. Controller design

One of the main limitations of the basic task hierarchy discussed in the previous chapter is that motion control of the torso is unrelated to the motion task for the end-effector. One of the consequences of this is (over)extension of the elbow when the end-effector is commanded to move too far away from the robot's torso. This results in undesirable behavior around this configuration such as unstable motion and awkward arm configurations. Another consequence is that the workspace reachability of the end-effector is limited by the prescribed pose of the robot's torso. The absence of contributing torso motion for the end-effector motion not only limited end-effector reachability in the horizontal plane, but also made it very difficult for the end-effector to reach the floor without explicitly ordering the torso height to be lowered first. These problems can be addressed through the implementation of kinematic constraints in the form of inequality tasks.

In Section 2.5.1 the implementation was presented of inequality tasks which represent kinematic limits such as joint limits. Although the joints of the Kinova arm used on the robot are limitless, it can still be desirable to specify joint limits such that undesirable configurations are avoided and whole-body motion can be triggered.

In order to address the mentioned issues, two joint limit tasks were implemented according to the method described in Section 2.5.1. First a joint limit for the elbow was specified such that (over)extension is prevented. The elbow joint typically moves between a fully flexed position (0 rad) and fully extended position (π rad). A maximum joint limit value is set at an angle of 3 rad (Figure 4.1). This allows the elbow to access almost its full range of motion, while avoiding the unstable behavior that was witnessed at full extension of the joint.

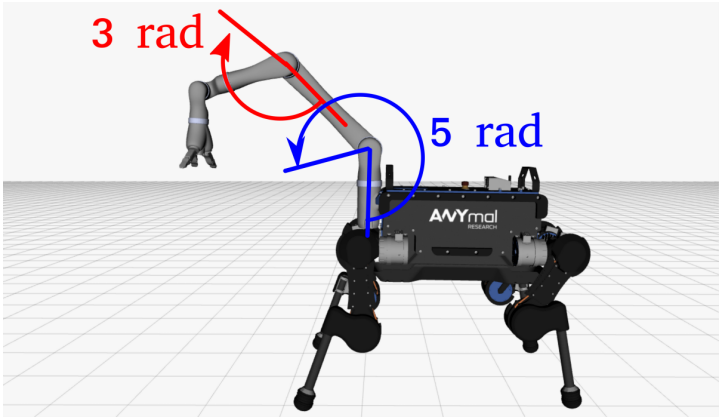


Figure 4.1: Maximal arm joint angles for the joint limit tasks. The maximum elbow angle is set to 3 rad. The maximum shoulder flexion/extension angle is set to 5 rad.

The second joint limit task was designed for the shoulder flexion/extension joint. When the arm points vertically up this shoulder joint is defined to be at an angle of π rad. Positive rotation results in the upper arm moving to a forward-pointing position. When commanding the end-effector to move towards the floor, two issues arise. First, without additional motion of the torso, the end-effector can only barely reach the floor. Second, when attempting to reach the floor it is possible for the upper arm to collide with the torso as a result. To solve both of these problems a joint limit task was implemented for the shoulder joint which specifies a maximum joint angle of 5 rad, allowing the upper arm to rotate down only slightly further beyond a horizontal position (Figure 4.1).

Figure 4.2 shows the new task hierarchy which includes the new kinematic limit tasks. Compared to the basic implementation (Figure 3.1) from the previous chapter an extra hierarchy level is added. This is done such that desirable whole-body motion is initiated when one of the kinematic limits of the arm is reached. When one of the kinematic limit tasks is activated, the arm loses an available DOF for the control of the end-effector pose. Because the end-effector pose also depends on the 6 DOFs of the floating-base, one or more of these can be used to contribute to the end-effector motion task. Although accurate tracking of the horizontal COM position is crucial for balance, accurate tracking for torso height and orientation is less critical. Therefore the current task hierarchy is designed such that it will result in vertical and angular motion of the torso when the higher priority end-effector tracking task cannot be fulfilled without that. The torso height and orientation task will be fulfilled as well as possible in a least squares sense while guaranteeing correct execution of the end-effector motion task. Individual weights can be set for the torso height task and the rotational directions of the torso orientation task such that a desirable amount of contribution to the end-effector task can be attained for each of these torso motions.

The joint limit tasks are designed according to the design discussed in Section 2.5.1. They are combined into a single task of the following form:

$$\frac{\delta t^2}{2}\begin{bmatrix}\mathbf{S} & \mathbf{0}\end{bmatrix}\boldsymbol{\xi} \leq \begin{bmatrix}\phi_{max,sh}\\ \phi_{max,el}\end{bmatrix} - \begin{bmatrix}\phi_{sh}\\ \phi_{el}\end{bmatrix} - \delta t\begin{bmatrix}\dot{\phi}_{sh}\\ \dot{\phi}_{el}\end{bmatrix} \tag{4.1}$$

where the selection matrix $\mathbf{S}$ selects the generalized accelerations in the optimization vector that correspond to the shoulder (sh) and the elbow (el) joints respectively. As mentioned, $\phi_{max,sh} = 5$ rad, and $\phi_{max,el} = 3$ rad. The value for δt was chosen to be 0.1 s. This value showed to provide a good trade-off between preventing sudden control jumps when the inequality task is activated (low δt) and a too soft constraint (high δt) which can cause considerable overshoot and oscillations around the joint limits.

4.3. Results

First, in order to illustrate the effect of the elbow joint limit in the new task hierarchy a situation is considered where the robot is required to pick up an object from a high shelf (Figure 4.3). When the end-effector is

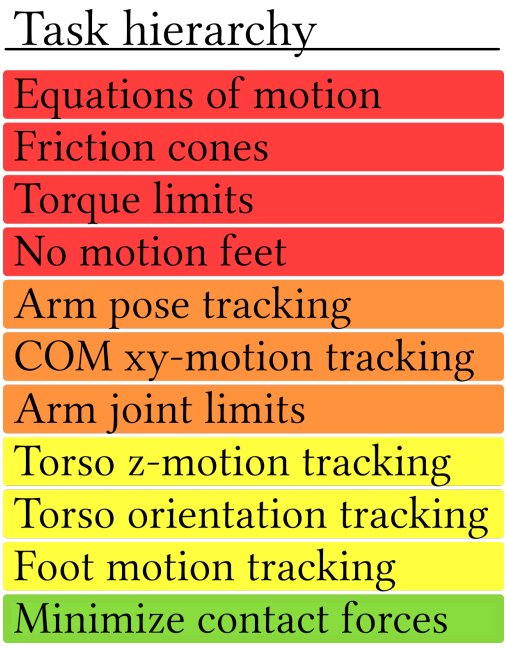


Figure 4.2: Task hierarchy including arm joint limits. Tasks are grouped and ordered from high (red) to low (green) priority. The extra priority level (orange) will result in torso motion to accommodate the end-effector motion tracking task when one of the arm joint limits is activated.

commanded to move vertically up, the maximum elbow angle ϕ_{max} is reached, which triggers motion of the torso to accommodate the reaching.

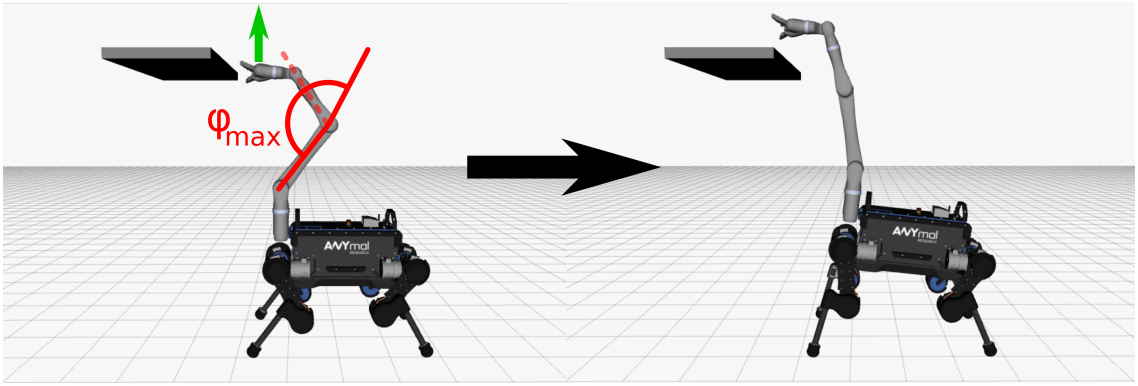


Figure 4.3: Improving workspace reachability as a result of the task hierarchy. When the end-effector is commanded far away from the torso, e.g. to a high shelf, the maximum elbow angle ϕ_{max} will be reached which will trigger a whole-body reaching motion due to torso motion.

The results of this task are presented in Figure 4.4. The top panel shows how the end-effector is commanded to move up by 20 centimetre along the z-axis. The desired reference trajectory is closely tracked throughout the entire motion, with a small delay due to damping on the end-effector velocity. However, the second panel shows that at a moment halfway through the motion (indicated by the black dotted line) the maximum elbow angle is reached. Around this same moment the torso height and pitch start increasing as a result of the DOF in the arm that is lost to the elbow joint limit task, even though the torso's desired position actually remains constant. This torso motion results in a continuous tracking of the end-effector reference motion.

It is interesting to note that the torso motion starts slightly earlier than the moment at which the joint limit is exceeded. In fact, it starts about 0.1 seconds earlier, which corresponds to the value of δt in the task formulation (eq. 4.1). This means that the controller computes a control action which can be expected to have the inequality task fulfilled after 0.1 seconds. However, this reasoning does also result in a small overshoot of the prescribed joint limit because this 0.1 second horizon moves along as time progresses and therefore

immediate satisfaction of the constraint is never demanded. Also interesting to note is that the torso pitch angle shows a small overshoot before settling at a stable value, while the torso height shows critically damped behavior towards a stable value. This can be attributed to the fact that the vertical torso motion is relatively strongly damped compared to the torso's angular motion. The controller sacrifices the performance of the torso motion tracking tasks in order to fulfill the higher priority tasks, but it tries to keep this performance sacrifice minimal. In this case a slightly higher position error for the torso pitch was valued of lower cost by the controller than a higher velocity error of the vertical torso motion.

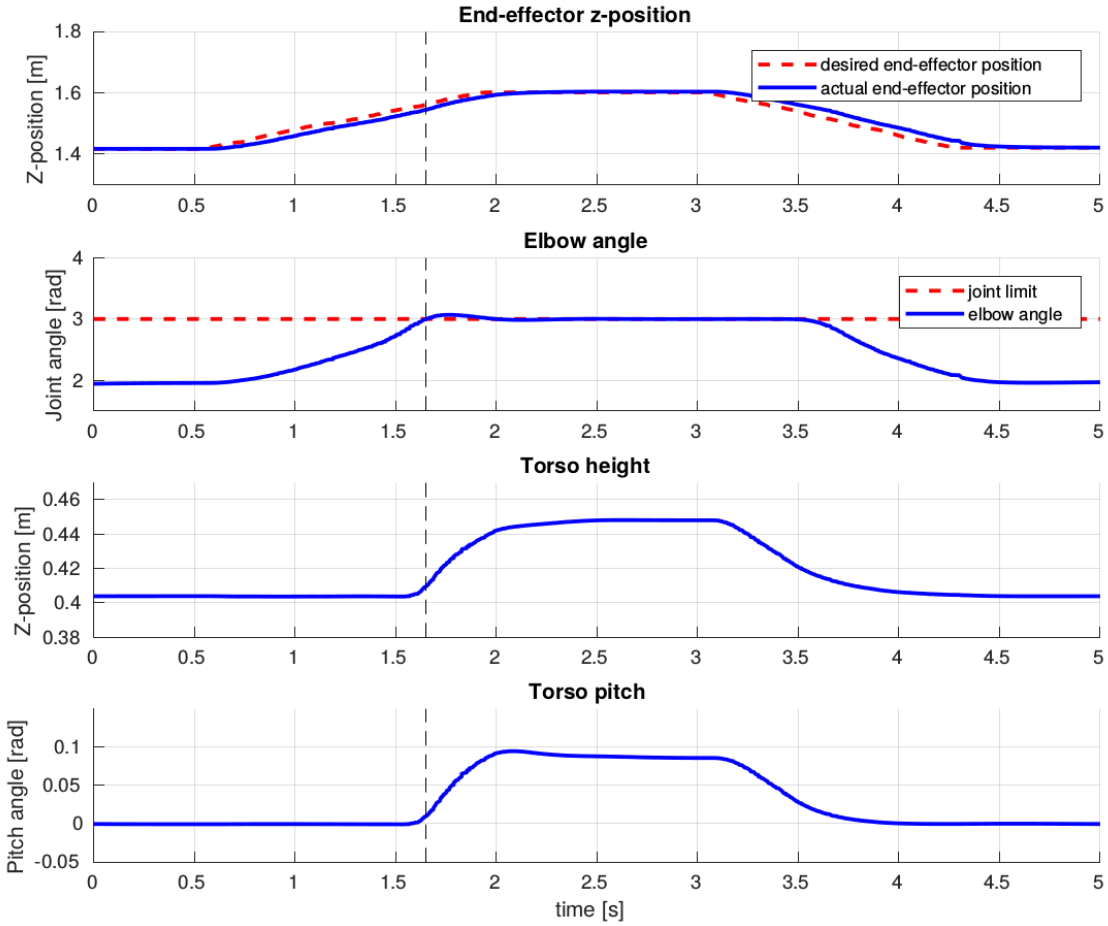


Figure 4.4: Reaching task with elbow joint limit task. As the end-effector is commanded to move up, the elbow joint limit is reached when the arm extends. This moment is indicated by the black dotted line. The activation of the joint limit task triggers motion of the torso such that the end-effector motion task remains unimpaired.

When reaching for a high position, the controller can sacrifice performance of the torso height and pitch tracking tasks to maintain performance of the end-effector pose tracking task. Similar behavior can also be seen for other configurations in which the elbow extension limit is reached. Figure 4.5 shows a situation where the end-effector is commanded to reach laterally. In this case it is mostly the torso roll and yaw motion that can contribute to the end-effector motion task and therefore the controller chooses to sacrifice performance of these tasks in this situation. Hereby again preventing overextension of the elbow, while also increasing workspace reachability, without impairing balance.

When commanding the end-effector to move down towards the ground, similar base motion can be seen to assist end-effector motion when the maximum shoulder joint angle is reached. This situation is illustrated in Figure 4.6. When the maximum shoulder angle ϕ_{max} is reached and the end-effector is still commanded to move down, the controller will sacrifice the performance of the torso's height and orientation tasks which results in a whole-body reaching motion such that the end-effector can reach the ground. The rightmost situation shows what would happen using the basic controller implementation from Chapter 3. First it is difficult for the end-effector to reach the ground around the robot because the base pose remains constant. Second,

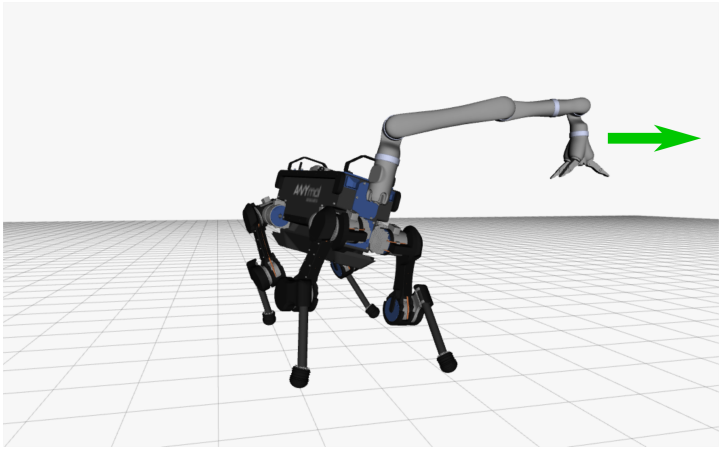


Figure 4.5: When commanding lateral motion for the end-effector, the presented task hierarchy will result in a roll and yaw motion of the robot's torso when the elbow's joint limit is reached. This additional motion increases the end-effector's workspace reachability.

when performing this task on the side of the robot the upper arm of the robot risks colliding with the robot's torso as a result of the commanded end-effector motion. Both of these problems are solved simultaneously with the shoulder joint limit task.

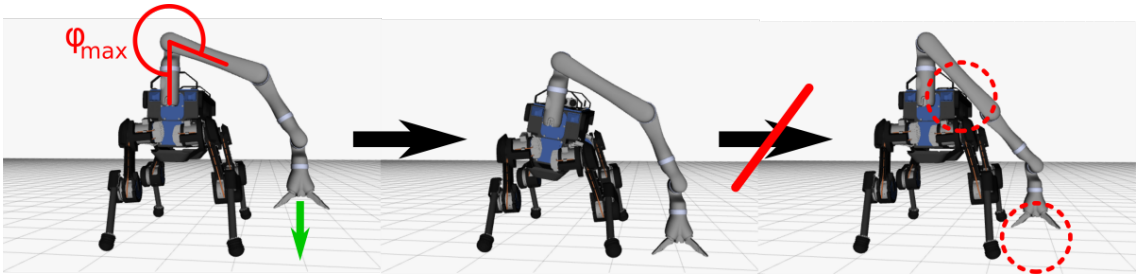


Figure 4.6: A joint limit task for the shoulder flexion joint can result in whole-body reaching and preventing self-collision. ϕ_{max} indicates the maximum shoulder angle. The middle picture shows whole-body motion to compensate for the loss of the shoulder DOF. Because the torso is lowered and tilted, the end-effector can move to the ground. The right picture shows the problematic self-collision and limited end-effector reach when the joint limit task is not included.

4.4. Discussion

The task hierarchy presented in this chapter results in behavior that addresses six of the seven control requirements listed in Section 2.6. The results in this chapter show that the implementation of simple joint limit tasks for the elbow and shoulder in the task hierarchy result in avoidance of a kinematically singular configuration and self-collision, while also resulting in the emergence of whole-body motion in which all of the robot's joints are utilized to improve workspace reachability of the end-effector. Furthermore, all this behavior is attained while the robot's user only needs to input the desired operational-space motion of the end-effector.

For this specific implementation the choice was made to only designate certain base DOFs for potential contribution to end-effector motion such that the robot's COM could remain at the desired location at all times. However, one of the main advantages of a quadrupedal robot is its large base of support which allows for considerable COM displacement before balance is compromised. It can therefore be interesting to consider also allowing some horizontal COM motion when one of the arm's inequality constraints is activated. This would allow for an even larger increase in workspace reachability at the expense of the robot's stability margin. However, allowing the robot to place all of its mass closely over just one or two of its legs can quickly cause the actuators to operate near their torque limits. Besides, in many situations it can be expected to be possible to simply trigger legged locomotion when the end-effector is commanded too far horizontally from

the robot's torso. If the robot's environment allows for this, it would be the more sensible solution because it avoids operating near actuator torque limits and risking loss of balance. However, because locomotion falls outside the scope of this work, this option has not been further investigated.

As discussed in Section 4.2, for the Taylor expansion used to estimate the robot's joint positions at the next time step a value (0.1 s) for the time step δt is used which is significantly larger than the actual control period of the robot (0.0025 s). This is done because otherwise as soon as the joint limit inequality task is activated it would result in the controller computing joint accelerations that would fulfill the task by the next time step. This results in undesirable jumps in the control signal, which is reported in [19] as a reason not to include such a joint limit task. Setting the value of δt to a value higher than the actual control period has been mentioned as a way to soften the constraint [10][38]. The results in this chapter show that by doing this the inequality task becomes activated a little while before the kinematic limit is actually reached, leading to an early and smooth initiation of compensatory torso motion. However, this same "soft" implementation also results in a considerable overshoot of the set joint limit because the controller only computes joint accelerations which will enforce the inequality constraint after the chosen value of δt, while this horizon keeps moving ahead while time proceeds. An interesting approach to investigate is to modulate the value of δt depending on the distance from the kinematic limit. A high value could be chosen on high distances, resulting in early and smooth anticipatory action, while the value can be lowered when the kinematic limit is closely approached so that a significant overshoot is prevented.

Another interesting topic for future work is variable limit values. For example the shoulder limit task presented here does not coincide with a physical limit on the joint. Instead it is chosen partly to prevent the manipulator arm to collide with the robot's torso when the arm reaches for the ground next to the robot. However, when reaching in front of the robot a larger flexion of the shoulder joint would be allowable. In this case the shoulder joint limit could be determined as a smooth function of the arm's position.

Originally many of the implementations of multi-task operational-space control frameworks avoided dealing with kinematic limits because these were difficult to integrate into the analytical control frameworks. The recent development of numerical optimization-based approaches has allowed for the easy integration of inequality tasks such as kinematic limits. Although these tasks are typically integrated purely to prevent the robot from hitting physical limits, some authors do mention integration into a task hierarchy such that compensatory motion of other joints it triggered once a kinematic limit is reached. In [17] a hierarchical task implementation is shown that results in a humanoid robot relaxing its COM position control when a reaching motion of the hand leads to hitting a joint limit in the arm, thereby adding motion of additional DOFs for the execution of the hand motion task. However, this controller is inverse kinematics-based, and therefore only takes into account positions and velocities. An example of triggering whole-body motion through implementation of kinematic limit tasks in a task hierarchy for an inverse dynamics-based controller is shown in [10]. There a kinematic limit is specified for the distance between the foot and hip of the torque-controllable quadrupedal robot ANYmal in order to achieve perception-less terrain adaptation during locomotion. When the robot reaches an unexpected drop in terrain during locomotion and the swing foot cannot be moved to the ground without exceeding the specified limit, the robot's torso pitch control task will be relaxed such that a whole-body motion is initiated by tilting the torso such that swing foot can be lowered further.

5

Posture optimization

5.1. Introduction

This chapter presents the results of a brief exploration that was done on the topic of posture optimization tasks in the considered control framework. Posture optimization is a commonly used approach to optimize the performance of a kinematically redundant robot.

As illustrated in the previous chapter, some DOFs of the robot's base, such as its orientation, do not require strict control and therefore offer a kinematic redundancy that can be used for posture optimization of the robot. This posture optimization can be be focused at improving a desired aspect of performance such as avoiding kinematic singularities or optimizing actuator efficiency. This chapter presents how equality tasks can be designed and implemented into the task hierarchy such that a whole-body behavior emerges in order to optimize the robot's posture according to a desired measure. This is illustrated by results of a posture optimization task which is aimed at optimizing the translational manipulability of the arm.

5.2. Controller setup

Posture tasks can be formulated as equality tasks based on configuration dependent cost functions. Consider a function $f(\mathbf{q})$ as a function of the robot's generalized coordinates that rates the current configuration of the robot with a single value according to a desired criterion. A higher posture rating can then be achieved by motion of the robot's generalized coordinates along the gradient of this function: $\nabla f(\mathbf{q})$. This was first done for a torque-based inverse dynamics controller by [45]. Similar to that implementation, the control framework used in this work allows specifying the posture optimization task as a function solely of the generalized accelerations $\ddot{\mathbf{q}}$, while the corresponding required joint torques will be computed due to forced compliance with the high priority EOM control task. Similar to the control law presented in [45], a control task that results in a damped motion towards the desired posture is formulated as:

$$\begin{bmatrix}\mathbf{S} & \mathbf{0}\end{bmatrix}\boldsymbol{\xi} = -k_d\left(\dot{\mathbf{q}}_S - \frac{k_p}{k_d}\nabla f(\mathbf{q}_S)\right) \tag{5.1}$$

Here $\mathbf{S}$ is a selection matrix that selects the entries in $\boldsymbol{\xi}$ that correspond to the generalized coordinates $\mathbf{q}_S$ on which the posture function depends. k_p and k_d are respectively a proportional and derivative gain. This control law directly commands joint accelerations such that the desired joint velocities according to the function gradient are achieved.

The controller implementation in this chapter will focus on arm posture optimization through whole-body motion. The choice for focusing only on posture measures for the arm is because limitations in the robot's performance are more likely to occur at the arm than at the legged base. The 6 DOF arm is expected to have to perform a wide and varying range of motions which could potentially result in awkward arm configurations

or (self-)collision. This is in contrast with the simple 3 DOF legs which typically perform the same type of translational motion to support the torso. Also in terms of joint torques the arm is more limited. While the ANYdrives[1] in the legged base have torque limits of 40 N, the Kinova arm joints have torque limits of only 30.5 N for the shoulder and elbow joints and 6.8 N for the wrist joints [4].

For implementation of the posture task into the task hierarchy the focus is again on leaving the end-effector pose tracking task unimpaired while exploiting motion of some of the base DOFs for the posture optimization of the arm. Following the reasoning presented in the previous chapter, the torso's height and orientation DOFs are chosen to be used for the fulfillment of the posture task, because these DOFs do usually not require strict motion control as long as the COM motion task is still executed properly. Figure 5.1 shows the task hierarchy designed for arm posture optimization.

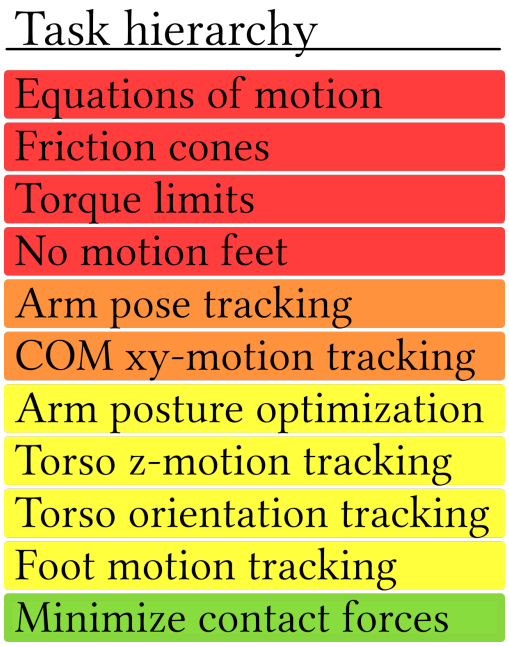


Figure 5.1: Task hierarchy for arm posture optimization. The additional posture optimization task results in a total task set that requires more DOF than the robot has available. In the current hierarchy this results in conflicting tasks for the base height and orientation DOFs. This will result in a base height and orientation that partly fulfills their corresponding motion tasks and partly fulfills the arm posture optimization task.

The arm posture optimization task is placed at the same hierarchy level as the torso height and orientation tracking task, but at a lower priority than the arm pose and COM position tracking tasks. Because the arm posture task adds additional DOFs to the total DOF requirement for all tasks, the robot no longer has enough DOFs available to independently fulfill all tasks. The chosen hierarchy therefore results in proper execution of the tasks in the first and second priority level for which enough independent DOFs are available, but in conflicting tasks in the third priority level because of a lack of remaining independent DOFs. The reason that the torso height and orientation tracking tasks are left in is because this will result in a regulating effect. Without these tasks the torso might move to extreme and unfeasible positions to satisfy the arm posture optimization task. However, in the current case the 4 DOFs of the torso height and orientation will be used such that a least squares optimal performance will be attained for all three tasks in the priority level. Individual task weights can be tuned to increase or decrease the regulating effect of the motion tasks.

As a demonstration of this control task hierarchy an arm posture optimization task is designed aimed at increasing the translational manipulability of the arm. The manipulability measure was first introduced in [52] and expresses to what extent the joint velocities can result in end-effector velocities in all directions of the operational-space. Near kinematically singular configurations, which are characterized by the loss of a controllable operational-space DOF, the manipulability of the robot will be low, whereas it will be high when all operational-space DOFs of the end-effector are well controllable. The manipulability measure $m(\mathbf{q})$ for a robotic system and corresponding end-effector can be calculated as a function of the configuration

dependent Jacobian:

$$m(\mathbf{q}) = \sqrt{\det\left(\mathbf{J}(\mathbf{q})\mathbf{J}(\mathbf{q})^T\right)} \tag{5.2}$$

In order to formulate the posture optimization control task, the gradient $\nabla m(\mathbf{q})$ of the manipulability measure with respect to the generalized coordinates is required. The individual components of this gradient vector can be computed[37] as:

$$\frac{\partial m(\mathbf{q})}{\partial \mathbf{q}_k} = m(\mathbf{q})\, Tr\left(\frac{\partial \mathbf{J}(\mathbf{q})}{\partial \mathbf{q}_k}\mathbf{J}^+\right) \tag{5.3}$$

$\mathbf{J}^+$ denotes the Moore-Penrose pseudo-inverse of the Jacobian matrix $\mathbf{J}$. The total gradient vector can then be inserted in eq. (5.1) to form the posture optimization task.

Finally the choice of the Jacobian determines which motions of which point on the robot are considered. In this demonstration the Jacobian is chosen that relates the rotational motion of the two shoulder joints and the elbow joint to the translational motions of the end-effector. This is done because of the six arm joints, these three have the strongest relationship to the end-effectors translational motion. The choice for this manipulability measure means that motion of these joints along the corresponding manipulability gradient $\nabla m(\mathbf{q})$ results in an improved ability of the arm to produce translational motion of the end-effector in the operational-space Cartesian directions.

5.3. Results

To illustrate the behavior as a result of the posture optimization task hierarchy a comparison is made with the basic implementation from Chapter 3. The task performed with both control setups consists of the tracking of an upward motion for the end-effector. Figure 5.2 shows the initial and final positions of the robot for both control setups. The positions of the end-effector and the feet are exactly the same for both setups. The difference lies in the height and pitch of the torso. For the posture optimization task hierarchy this torso motion is partly controlled such that it increases the manipulability of the arm. When comparing the arm configurations for the upward reaching posture, it can be seen that the posture optimization task decreases the amount in which the arm links are aligned, thereby increasing manipulability and staying away from the extended elbow singularity.

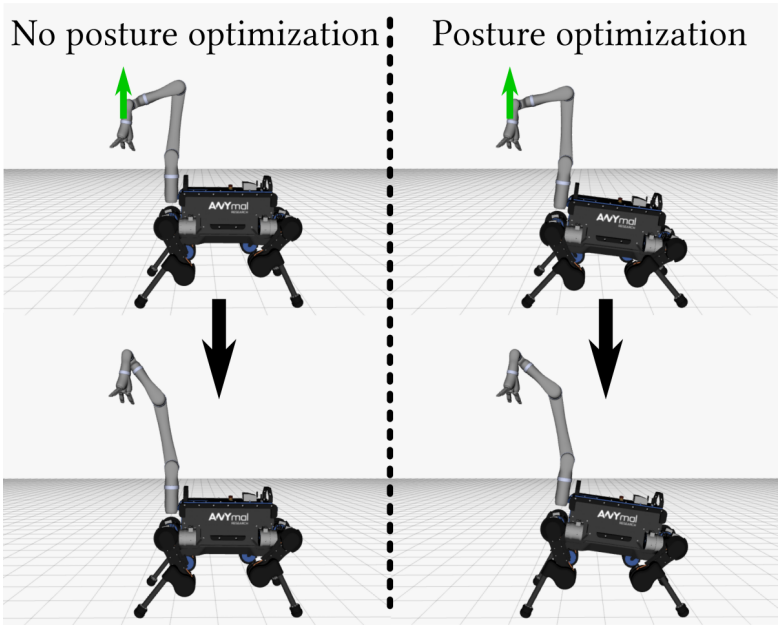


Figure 5.2: Robot posture without (left) and with (right) arm posture optimization task. When the posture task is present, the robot's torso height and orientation is partly controlled such that the end-effector's translational manipulability as a function of the lower 3 arm joints is improved. When the end-effector is commanded away from the body the inclusion of the posture task results in torso motion that prevents the elbow from overextending.

More detailed information about this comparison is given by the plots in Figure 5.3. The top plots show that both controller implementations result in identical satisfactory motion tracking of the end-effector. The second row of plots show the manipulability measure of the arm as it moves. It can be clearly seen that the manipulability of the arm remains significantly higher throughout the task when the posture optimization task is present. The bottom two rows of plots show the motion of the torso height and pitch during the reaching motion. In the case where there is no posture optimization task, the torso height and pitch can be freely controlled to remain at their desired values of respectively 0.41 m and 0 rad. In contrast, when there is a posture task, the torso height and pitch deviate considerably from their desired values because these DOFs are also used to increase the arm's manipulability.

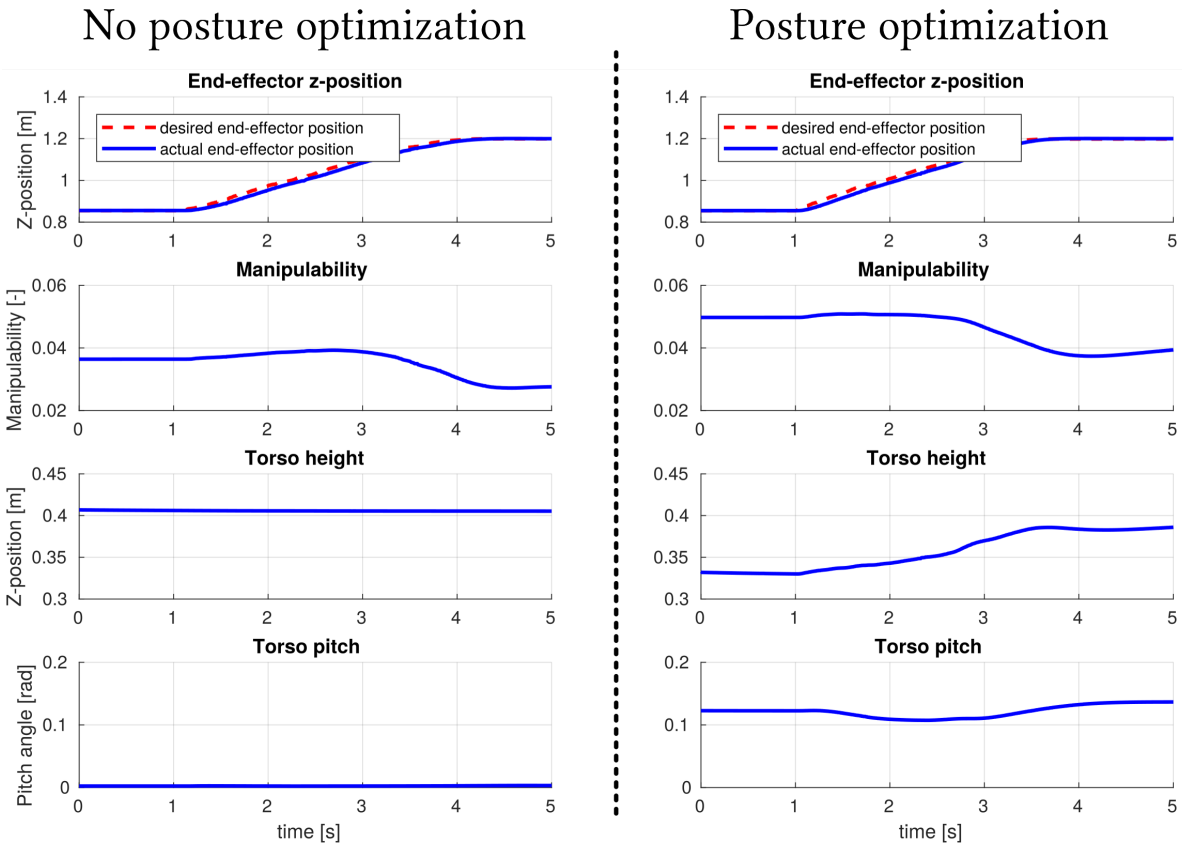


Figure 5.3: Comparison in robot behavior without (left) and with (right) arm posture optimization task when an upwards reaching motion is commanded for the end-effector. Without the posture task the torso height and pitch remain constant at their desired values while the end-effector follows it desired motion. Before the end-effector motion is initiated, the pose of the robot's torso is already different for the case where the posture optimization task is included, resulting in a higher manipulability measure of the arm. When the motion of the end-effector starts, the height and pitch of the torso change as well to increase the arm manipulability for the new end-effector positions. This results in a higher arm manipulability throughout the entire end-effector motion when the posture task in included compared to when it is not.

Figure 5.4 shows how this controller setup performs on other challenging reaching tasks. The left picture shows that when reaching for the ground the posture task results in a lowering and tilting of the torso to increase the manipulability of the arm. It hereby prevents the previously mentioned limitations of the basic controller implementation where the end-effector could not reach the ground, could end up near a singularity, and risked collision with the torso. The right picture shows how a commanded lateral reaching motion of the end-effector results in a yaw and roll of the torso, which increases the arm's workspace reachability while preventing overextension of the elbow.

5.4. Discussion

In this chapter it is shown how posture optimization can be achieved for a quadrupedal manipulator using an hierarchical inverse dynamics control framework. First a posture optimization task is designed which

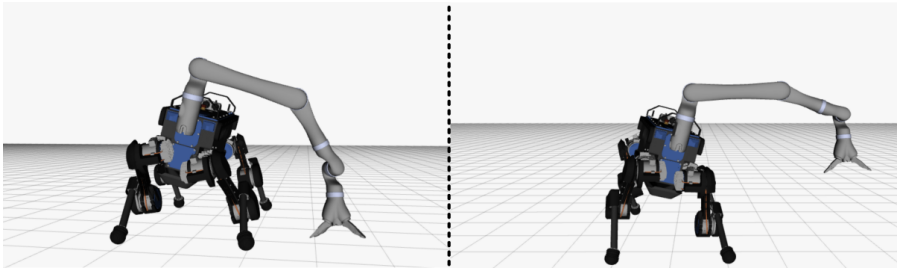


Figure 5.4: Commanding end-effector motion while optimizing for arm posture results in whole-body reaching motion. When commanding the end-effector to move to the ground (left panel), the torso's orientation (primarily roll) and height are significantly deviating from their desired values in order to keep the arm's manipulability high. This results in a whole-body reaching motion, which allows the end-effector to reach the ground without overextension of the elbow or collisions between the arm and torso. A similar whole-body reaching behavior emerges when commanding to end-effector to reach laterally (right panel). In this case it is primarily the torso's yaw and roll angles that deviate from their desired values in order to keep the arm's manipulability higher.

prescribes motion of the robot's joints along the gradient of a desired posture measure function. By placing this task at the same priority as the motion tasks for the torso's orientation and height, these DOFs will be controlled such that they partly fulfill their motion tasks and partly allow for posture optimization of the robot. By placing the end-effector pose tracking task at a higher priority, the execution of this task will not be affected by the posture optimization.

The illustration in this chapter, based on a manipulability optimization task for the arm, shows that this set of control tasks results in smooth reactive torso motion which helps avoiding arm singularities and self-collision with the torso, and results in an increased workspace reachability of the end-effector. Besides, balance is maintained at all time, while real-time control input for the motion control of the end-effector is still possible. Therefore the implementation presented in this chapter successfully addresses six of the seven control requirements listed in Section 2.6.

Here the implementation of an arm posture optimization task was demonstrated with the use of a posture measure based on the translational manipulability of the arm. However, any measure which rates the robot's posture as a function of some or all of its generalized coordinates can be used in the presented task hierarchy. For example a similar manipulability measure could be designed which results in optimization of the wrist joints to maximize rotational manipulability of the wrist. This could then be used to avoid the kinematic wrist singularity which occurs when the axes of the 1st and 3rd wrist joints align, which results in the loss of an axis of rotation of the end-effector. Other examples include optimizing posture to distribute gravitational forces according the individual torque limits of the joints [45], or to attain increased manipulability in a desired motion direction [12].

A particularly interesting measure is that of directional force manipulability. The velocity-based manipulability that is discussed so far indicates how well end-effector velocities can be obtained as a result of joint velocities. Here the manipulability is low when only low end-effector velocities can be obtained. However, the velocity-force duality based on conservation of power prescribes that a direction in which only low end-effector velocity can be attained, a high force can be generated. Such a measure can be extremely useful to optimize the robot's posture in order to increase the magnitude of the force the robot can exert on its environment, for example when pulling open a heavy spring-loaded door. Such a posture optimization has been shown for the control of an excavator boom with one redundant DOF [27]. It has however not yet been shown on a legged platform where motion of the base would be required, although it has been mentioned as future work for a quadrupedal manipulator [20].

Similar to the previous chapter the base height and orientation were designated to deviate from their desired positions in order to accommodate the performance of an additional task. This choice was made because these DOFs do not require to strictly remain at a specific position and do not directly influence COM position and balance. However, interesting about the quadrupedal platform is that considerable COM motion would be allowed without immediately impairing balance. This would particularly be an interesting implementation to pursue in combination with optimization of the directional force manipulability of the arm and with corresponding locomotion where footstep locations are planned while taking into account external forces at

the arm. Such an approach could result in efficient and robust force-based interaction with the environment, for which a basis is laid down in Chapter 6 and Chapter 7

When comparing the results from the task hierarchy presented here with the one from Chapter 4 it can be seen that they solve similar problems. Both controller implementations increase workspace reachability, prevent overextension of the elbow joint and result in whole-body collision-free motion when reaching for the ground. The advantage of using a posture optimization task is that it results in smooth continuous whole-body motions, instead of only using torso motion to improve manipulation at the moment when a kinematic limit is reached. On the downside, this additional torso motion is not always really necessary, and could perhaps be executed in a way which is not unintuitive or expected. A solution to that would be a variable weighting factor. As mentioned in Section 5.2 weighting factors can be set for the posture optimization task and the torso motion tasks. These weights can also be dynamically set as a function of a specific measure. For example the weight for a manipulability optimization task can be increased as a function of a decreasing manipulability measure or as a function of the lowest singular value in the arm's Jacobian. This way torso motion to optimize arm posture would only be initiated when truly required. Setting this weight as a function of such a measure which is smooth will also result in a smooth development of the weight factor and therefore of the computed control signal over time.

In planning-based control approaches for mobile manipulators it is not uncommon to consider the effect that base motion has on the posture of the manipulator [34]. For control approaches with an instantaneous nature such as the one considered in this thesis posture optimization is most commonly used as a form of redundancy resolution for otherwise uncontrolled DOFs [45][12][27]. In contrast, the implementation presented here exploits the fact that some DOFs of the robot require only a soft degree of control and can therefore also simultaneously be used to optimize a posture.

6

Force-based manipulation

6.1. Introduction

So far the focus of this thesis has been on motion-based manipulation. However, a mobile manipulator can be expected to have to perform tasks where force-based interaction with the environment is required. This could be carrying objects around, dragging/pulling/pushing such as opening a spring-loaded door, or handling tools which require contact force control such as a window-wiper or a drill. One of the major advantages of a fully torque-controllable robot such as considered in this work is that it allows for the direct control of these contact forces.

In this chapter a controller implementation will be shown which allows for real-time control of force-based manipulation. As discussed in Chapter 2 the hierarchical optimization control framework is able to implicitly distribute the required contact forces over the robot's point-feet while taking into account torque limits and the unilateral and friction-dependent character of the contacts. Here an extension will be presented to include potential contact forces and torques at the arm when the end-effector is in contact with the environment. Furthermore, a method is presented that allows for a smooth transition between motion-based and force-based control of the end-effector.

6.2. Controller setup

As explained in Section 2.3.4 the exact implementation of the EOM control task dictates the composition of the optimization vector $\boldsymbol{\xi}$. When using the reduced version of the EOM, as done in this research project, then the optimization vector only needs to contain the robot's generalized accelerations $\dot{\mathbf{u}}$ and contact forces $\boldsymbol{\tau}$. Which contact forces are taken into account is a design choice, but should correspond with the actual state of the robot for optimal performance. This means that the typical choice is to only include the contact forces of all the limbs that have their endpoints in contact with the world. As shown in Chapter 2, the EOM of the entire system are written as:

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{h}(\mathbf{q},\mathbf{u}) = \mathbf{S}^T\boldsymbol{\tau} + \mathbf{J}_s^T\boldsymbol{\lambda} \tag{6.1}$$

This shows that changes in the choice of incorporated contact forces in $\boldsymbol{\lambda}$ need to be reflected in the constraint Jacobian $\mathbf{J}_s$, which is the Jacobian that relates joint velocities to limb endpoint velocities. When taking into account forces at the arm's end-effector then these can be added by simply adding these to the contact forces vector $\boldsymbol{\lambda}$ (which is also part of the optimization vector $\boldsymbol{\xi}$) and updating the support Jacobian $\mathbf{J}_s$ accordingly. In the implementation presented here only the linear forces at the end-effector are taken into account. However, in contrary to the point-feet, a gripper at the end of the 6 DOF arm would also be able to exert pure contact torques on the environment. When desirable, these contact torques can also be added to the EOM and controller in the same way as described above. In this report only linear contact forces at the end-effector are considered.

Force-controlled contact between the arm's end-effector and the environment can be used to aid in balancing, but it can also be used to perform force-based manipulation tasks. Because the latter is expected to be more common, the focus on this chapter is on active control of the end-effector contact forces for manipulation tasks. For this a simple equality task can be designed which prescribes the desired values for the arm's contact forces. This can be done in the form $\begin{bmatrix}\mathbf{0} & \mathbf{S}\end{bmatrix}\boldsymbol{\xi} = \boldsymbol{\lambda}_{des}$, as discussed in Section 2.4.2, where $\mathbf{S}$ is a selection matrix selecting the entries of $\boldsymbol{\xi}$ that correspond to the contact forces at the hand. The control of the contact forces at the feet is left unchanged, which means that for the feet only the inequality task representing the friction cones is specified, leaving the exact distribution of the required contact forces over the feet to follow from compliance with the EOM task and the minimal contact forces norm task. This means that this control setup will solve the difficult task of contact force distribution while taking into account desired motions, contact forces at the hand, and the robot's dynamics.

In Figure 6.1 the task hierarchy including the arm contact force task is shown. The contact force task is placed at a separate priority level between the motion tasks and the final contact force minimization task. This is done such that the desired contact force will only be tracked as well as possible without interfering with the performance of any of the other tasks such as the motion tasks. This prevents undesirably high accelerations to be computed when the desired contact force cannot be tracked correctly otherwise.

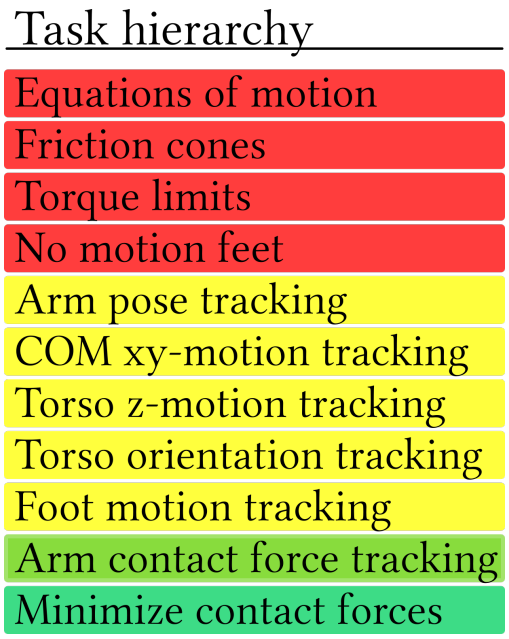


Figure 6.1: Task hierarchy including end-effector contact force control task. The additional priority level at which this task is placed guarantees that when this task will only be executed as well as is possible without disturbing the execution of any of the higher priority tasks.

Adding and removing control tasks such as the arm contact force task can result in undesirable jumps in the control signal from one control period to the next. In order to prevent this, special attention has to be payed to enforce a smooth transition when (de)activating force control at the end-effector. When examining the equations of motion (6.1) the term $\mathbf{J}_s^T\boldsymbol{\lambda}$ can be interpreted as the torque vector containing the joint torques that are required to attain the desired contact forces $\boldsymbol{\lambda}$. When adding a contact force task for the arm with $\boldsymbol{\lambda}_{des} = \mathbf{0}$, then both $\mathbf{J}_s^T$ and $\boldsymbol{\lambda}$ have to be adapted, but the vector computed by $\mathbf{J}_s^T\boldsymbol{\lambda}$ will remain the same. This means that a contact force task can be added and removed without any changes in control signal when the desired contact force for that limb is set to zero.

To take advantage of this the controller is designed such that the arm contact force task is added when a desired contact force unequal to zero is registered, and removed again when the desired contact force is back to zero. This desired force value can be a result of for example a planning algorithm, but can also be a direct result of real-time teleoperation input, such as through a joystick input. A simple first-order discrete-time filter of the form $\boldsymbol{\lambda}_{des}(t+\delta t) = \alpha\boldsymbol{\lambda}_{des}(t) + (1-\alpha)\boldsymbol{\lambda}_{des,input}(t+\delta t)$ is added in order to guarantee that the

desired contact force value will always progress smoothly from and to zero without causing jumps in the control signal, even when the input signal contains step changes. For a control period δt of 0.0025 s a value for α of 0.97 shows sufficient smoothing of input step changes.

The motion task for the arm's end-effector does not need to be removed from the task hierarchy when the contact force task is added. In fact, it is even desirable to leave it in. When no motion task is specified for the end-effector the controller will assume that the kinematic DOFs of the arm are available for other tasks, such as the COM motion task, which results in the computation of accelerations and corresponding joint torques for the arm. By inspection of the EOM (6.1) it can be seen that the robot's prescribed joint torques are a result of the desired generalized accelerations $\dot{\mathbf{u}}$, desired contact forces $\boldsymbol{\lambda}$, and of cancellation of dynamic effects $\mathbf{h}(\mathbf{q},\mathbf{u})$. In order to have the computed torques result in accurate tracking of the desired contact force it is important that the completion of the end-effector motion task does not require any joint torques. As discussed in Section 2.4.1, a typical motion tracking task is formulated as:

$$\begin{bmatrix}\mathbf{J}(\mathbf{q}) & \mathbf{0}\end{bmatrix}\boldsymbol{\xi} = \ddot{\mathbf{r}}_{des} - \dot{\mathbf{J}}(\mathbf{q})\mathbf{u} \tag{6.2}$$

$$\ddot{\mathbf{r}}_{des} = \ddot{\mathbf{r}}^{FF} + \mathbf{k}_d\left(\dot{\mathbf{r}}_{des} - \dot{\mathbf{r}}\right) + \mathbf{k}_p\left(\mathbf{r}_{des} \boxminus \mathbf{r}\right) \tag{6.3}$$

where for the end-effector motion tracking task $\ddot{\mathbf{r}}^{FF}$ and $\dot{\mathbf{r}}_{des}$ are set to zero, and $\dot{\mathbf{r}}_{des}$ corresponds to currently desired position and orientation of the end-effector. When the contact force tracking task is active, $\mathbf{r}_{des}$ can be set to be follow to the actual end-effector position by setting it equal to $\mathbf{r}$ such that $\ddot{\mathbf{r}}_{des}$ remains zero. This then corresponds to the no-motion task for the stance legs described in Chapter 3. Besides, when the contact force task is deactivated again, even when motion has occurred in the meantime, motion control can resume smoothly by ordering $\mathbf{r}_{des}$ to move away from $\mathbf{r}$ again.

In reality there are two reasons why extra attention has to be payed to the end-effector motion tracking task when the contact force task is activated. First, when a desired contact force is commanded while no actual rigid contact is present, high accelerations and velocities of the end-effector can result. These high velocities can be prevented by leaving the damping term $\mathbf{k}_d\left(\dot{\mathbf{r}}_{des} - \dot{\mathbf{r}}\right)$ in the computation of $\ddot{\mathbf{r}}_{des}$. The torques computed as a result of this part of the motion task will likely corrupt the execution of the contact force task, but will prevent excessive end-effector velocities. The second issue is that at the moment the desired contact force becomes non-zero and the contact force task is activated, it might be possible that there is a non-zero error $(\mathbf{r}_{des} \boxminus \mathbf{r})$. Immediately setting $\mathbf{r}_{des}$ to $\mathbf{r}$ would then result in an undesirable jump in the computed joint torques. In this case it is necessary to smoothly let $\mathbf{r}_{des}$ approach $\mathbf{r}$ before setting them to be equal.

The way this is dealt with is by computing the magnitudes of the position and orientation errors at the moment the contact force task is initiated and then ramping the desired position and orientation to the actual ones such that the error magnitudes smoothly decrease to zero within a specified time. For the ramping of the desired end-effector position $\mathbf{r}_{des}$ first the magnitude of the initial pre-ramping error e_{pre} is calculated when the contact force task is first activated at $t = 0$:

$$e_{pre} = \|\mathbf{r}_{des}(0) - \mathbf{r}(0)\| \tag{6.4}$$

From then on a ramping fraction α is calculated based on the time t since the contact force task is initiated, and a desired ramping time t_{ramp}:

$$\alpha = \frac{t}{t_{ramp}} \tag{6.5}$$

Then, at every next control period the desired position is clipped slightly closer to the actual position by scaling the normalized leftover error according to

$$\mathbf{r}_{des}(t) = \mathbf{r}(t) + (1-\alpha)e_{pre}\frac{\mathbf{r}_{des}(t) - \mathbf{r}(t)}{\|\mathbf{r}_{des}(t) - \mathbf{r}(t)\|} \tag{6.6}$$

When α equals one, or the error between the desired and actual end-effector position falls below a certain threshold, the desired position is set to be equal to the actual position for as long as the contact force task is still active. For a smooth but rapid ramping t_{ramp} is set to 0.5 s. The same procedure is followed for the end-effector desired orientation, except that the minus sign to compute the error in 6.4 and 6.6 has to be replaced by the boxminus operator $\boxminus$ [13], which makes it possible to express the orientation error as a vector with the axis-angle expression.

The setup on which the experiments were conducted is shown in Figure 6.2. A dynamical model of a heavy 50 kg door with a fixed door handle is added to the simulation. For most experiments the door is unable to move and functions as a rigid part of the environment onto which forces can be exerted by the robot. For the experiments where the door is able to move, the hinge joint limits of the door are set such that it is considered "shut" in the configuration shown in Figure 6.2, and can be pulled "open" in the direction of the robot. The hinges have a considerable friction of 10 N m. Because the hand at the end of the arm is not actuated in the simulation, the grasping of the door handle by the hand needs to be mimicked. This is done by adding a "fixed joint" in the simulation software between the hand and the door handle, thereby locking the hand's 6 DOF pose to that of the handle, similar to when the hand would actually have a tight grip on the door handle. The forces in this additional joint are obtained from the simulation software to measure the actual contact forces between the robot and the door handle.

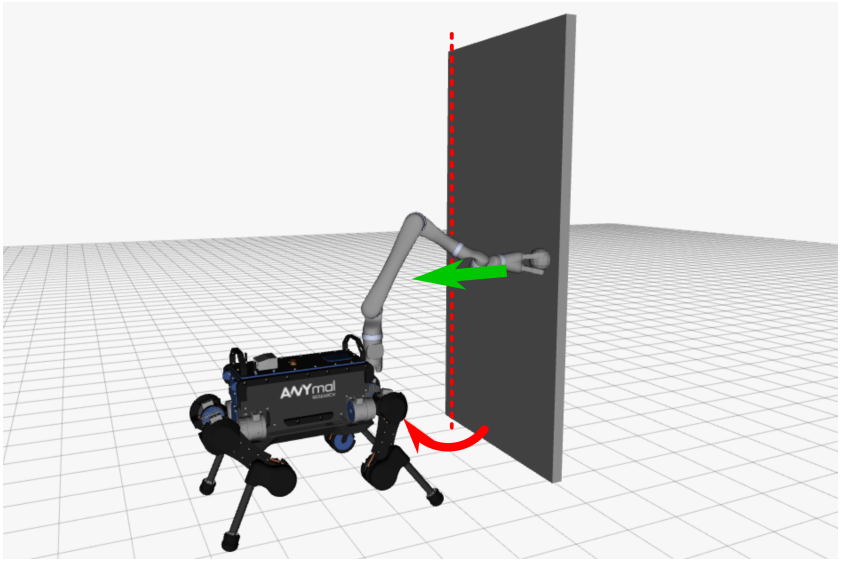


Figure 6.2: Setup in simulation for experiments on end-effector contact force control. The robot's end-effector is rigidly attached to the door handle by adding a fixed joint between the two objects in simulation. This allows the end-effector to transfer contact forces (green vector) onto the door. The door is "shut" in the current configuration but can be pulled open along the direction of the red vector. The door hinges can also be locked for the door to function as a rigid wall.

6.3. Results

For the first experiments the door's hinge is fixed such that the door functions as a rigid unmovable object at which forces can be exerted. As input for the desired contact forces at the hand a ~1 second step input was given for a desired contact force in x-, y-, and z-direction. Figure 6.3 shows the results of this experiment. The envelope of the desired contact force signal is a result of the first order filter that is applied to the step input signal such that jumps in the control signal are avoided. The plots show that the desired contact forces can all be tracked nicely without interfering with with the measured contact forces in the orthogonal directions.

To demonstrate the effect of the placement of the contact force task in the task hierarchy a situation was created where the foot friction cone limits prohibit the generation of the ground reaction forces required for the desired contact force at the hand. The design of the inequality task which enforces ground reaction forces at the feet to lie within the appropriate friction cone is discussed in Section 2.5.2. This task is based on an assumed friction coefficient μ and in order for the task to be fulfilled, the ratio between the tangent and normal components of the contact forces relative to the contacted plane needs to be smaller than the value of μ:

$$\mu \geq \frac{\text{tangent contact force component}}{\text{normal contact force component}} \tag{6.7}$$

The value of μ is normally set at a value of 0.5. However, for the experiment this value was reduced to 0.05. With these much "thinner" friction cones, the magnitude of the allowable tangent components of the contact forces at the feet is drastically decreased. Figure 6.4 shows what happens when a desired horizontal contact

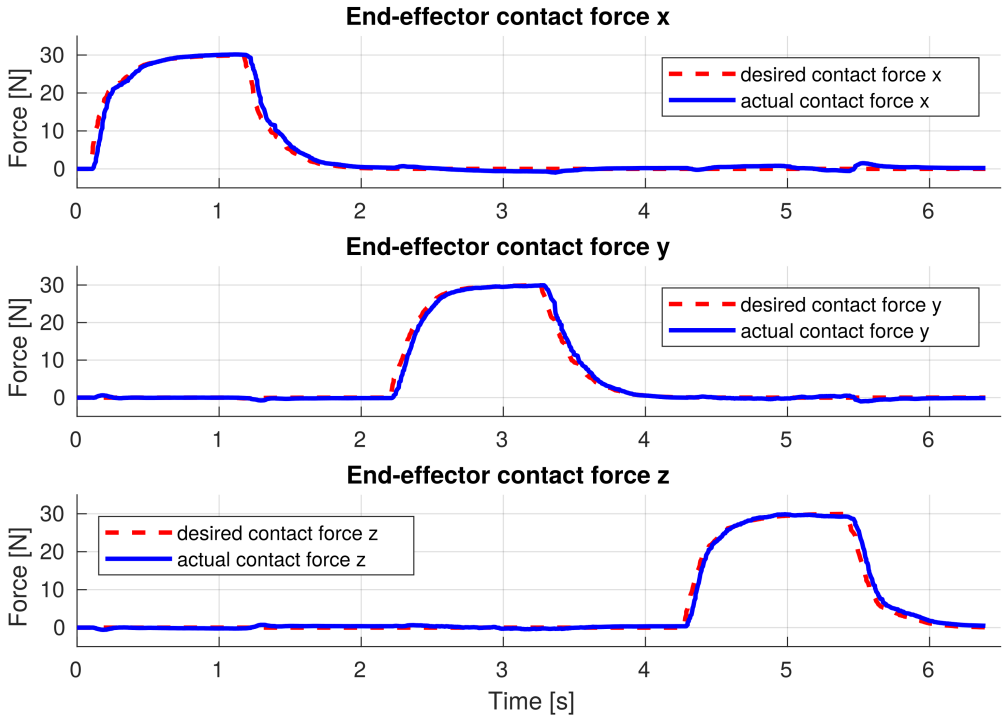


Figure 6.3: End-effector desired contact force tracking. These plots show that when the end-effector is in fixed contact with a rigid object any desired contact forces can be accurately tracked for each individual Cartesian direction.

force of 30 N between the hand and the door is prescribed. The top panel shows the desired and actual contact force of the end-effector along the x-axis. The bottom panel shows the ratio between the frontal and normal components of the ground reaction force measured at the front left foot. As the desired contact force at the hand increases, the contact force ratio also increases because the required compliance with the EOM task demands that the x-direction contact forces at the feet must equal the x-direction contact force at the end-effector. When the maximum ratio - indicating the edge of the friction cone - is reached, the friction cone inequality task becomes active, which results in a limitation of the contact force that is exerted at the end-effector. In other words, the end-effector contact force task is executed as well as possible without violating any of the higher priority tasks including the friction cone limit task. If the end-effector contact force task would be placed at any of the higher priority levels then part or all of the action required to avoid the friction cone limit would be through acceleration of the COM of the robot. Since this is not a sustainable strategy and quickly results in crashing of the robot, this would be highly undesirable.

A similar demonstration can be given when taking into account the joint torque limits. For this experiment the robot's arm is positioned such that the wrist flexion/extension joint has a considerable moment arm with respect to horizontal contact forces at the end-effector by having the hand point down while it is connected to the unmovable door. In this case high desired contact forces at the end-effector result in saturation of the torque that the actual wrist joint can deliver, which is 6.8 N m [4]. Because the joint torque limits are integrated into the top level of the task hierarchy, the controller will limit the execution of the contact force task such that the joint limits are respected. This can be seen by the results presented in Figure 6.5. The desired contact force at the arm is tracked properly until the wrist joint limit is reached. At that point the contact force at the end-effector levels off to the maximum achievable magnitude without violating the joint torque limit.

For the final experiment the door is set to allow motion by pulling it towards the robot, as indicated by Figure 6.6. This experiment will highlight the smooth transition between the motion and contact force tasks. First the robot's end-effector is moved towards the door handle, where then a rigid connection is formed between the hand and handle in order to simulate the door handle being grasped. At this point a motion for the end-effector is commanded in all three Cartesian directions, which moves the end-effector's desired position away

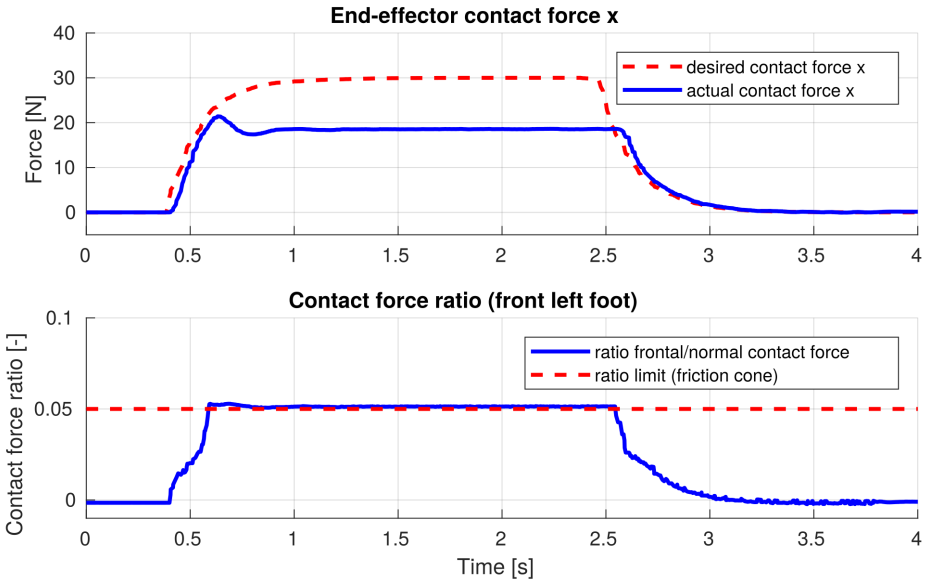


Figure 6.4: Commanding horizontal end-effector contact force while friction cone limits are reached. The bottom plot shows the ratio between the frontal and normal components of the contact force at one of the robot's feet. This ratio increases when the contact force at the end-effector increases. For this a "thin" friction cone assuming a low friction coefficient of $\mu = 0.05$ is designed. Because the friction cone task is of higher priority than the contact force task, the desired contact force is no longer tracked when this is impossible without violating the friction cone task.

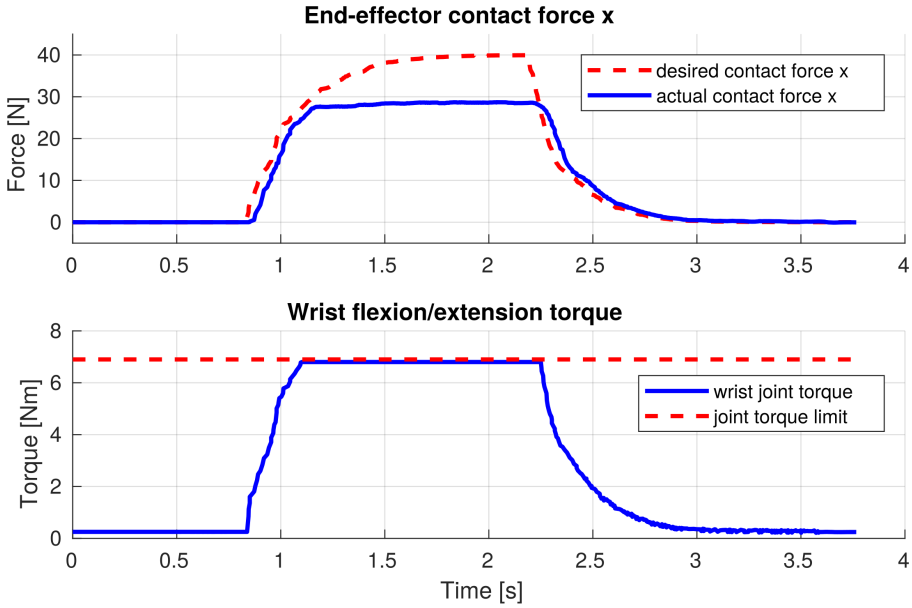


Figure 6.5: Commanding horizontal end-effector contact force while an actuator torque limit is reached in the wrist flexion/extension joint. The top plot shows the desired and actual contact force for the end-effector, while the bottom plot shows the wrist joint torque. Because the of the higher priority of the joint torque limit task, the desired contact force is only tracked as well as is possible without exceeding the joint torque limit.

from its actual position, and results in the robot pushing against the door because hand motion is restrained (Figure 6.6, left panel). This can be seen in the first 0.5 seconds of the plots in Figure 6.7. The first three plots show the actual and the deviating desired positions for the end-effector. The fourth plot shows the total position error magnitude. The final plot shows the desired contact force and the actually measured contact force between the robot and the door. It can be clearly seen that the end-effector's motion task which attempts to move the end-effector towards its desired position results in a considerable pushing contact force against

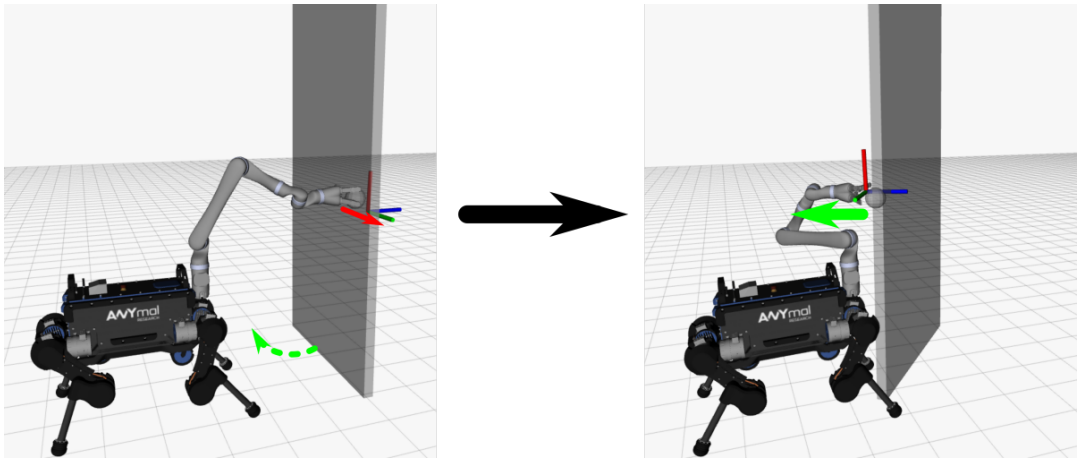


Figure 6.6: Experimental setup for demonstration of smooth transition between end-effect motion and contact force tasks. First the end-effector is rigidly connected in simulation to the door handle. Subsequently a motion of the end-effector is commanded, resulting in an error (red vector) between the actual end-effector pose and its desired reference frame. Next, an end-effector contact force is commanded (green vector) which results in motion of the door. When the contact force task is activated the desired end-effector pose is linearly ramped towards the actual end-effector pose such that the motion tracking task does not generate torques that interfere with the contact force task.

the rigid contact with the door. After 0.5 seconds a desired pulling contact force of 30 N m is commanded. This non-zero desired contact force results in the contact force task being added to the task hierarchy. The duration for which this task is active is indicated by the green boxes. As soon as the contact force task is active, the desired position of the end-effector is linearly being ramped towards the actual position, according to the methods described in the previous section. For demonstrational purposes this ramping time is set to a relatively long value of 1.5 s for this experiment. The contact force plot at the bottom shows that when the position error decreases, the contact force at the door starts moving towards its desired value. At around 1.4 seconds the contact force on the door becomes large enough to pull the door into motion, which can be seen by the change in actual position of the end-effector x-, and y-positions. The period during which the door is in motion is indicated by the darker green boxes. Despite the fact the the actual end-effector positions are changing, the ramping control law ensures that the total difference between desired and actual positions still decreases linearly, after which the desired positions are set to equal the actual positions such that the motion task no longer disturbs the execution of the contact force task.

The apparent irregularities in the linearly decreasing position error norm plot can be attributed to the fact that the simulation software does not always manage to run completely in real-time while the logging software assumes that it does and therefore uses the computer's time. The reason that the actual contact force in the bottom plot can be seen to never reach the desired contact force is because the remaining damping term in the end-effector motion task results in opposite joint torques which slow down the motion of the end-effector while the door is being pulled open. This prevents the end-effector from reaching undesirably high velocities as a result of controlling the contact force between the end-effector and a movable object. The behavior for desired and actual end-effector orientations during the initiation of a contact force task are similar to the results presented here for positions.

Finally it is interesting to note that by just specifying a desired contact force for the end-effector along the x-direction (while removing the position error) the robot ends up opening a heavy door. This behavior is possible because the robot does not experience drawbacks from the fact that the door's motion constrains the end-effector motion path because the end-effector is controlled not to track any motion, but just to produce a desired force. A similar task is considerably harder to execute using position control of the end-effector because in that the case prescribed end-effector motion needs to closely match the constrained motion path allowed by the door.

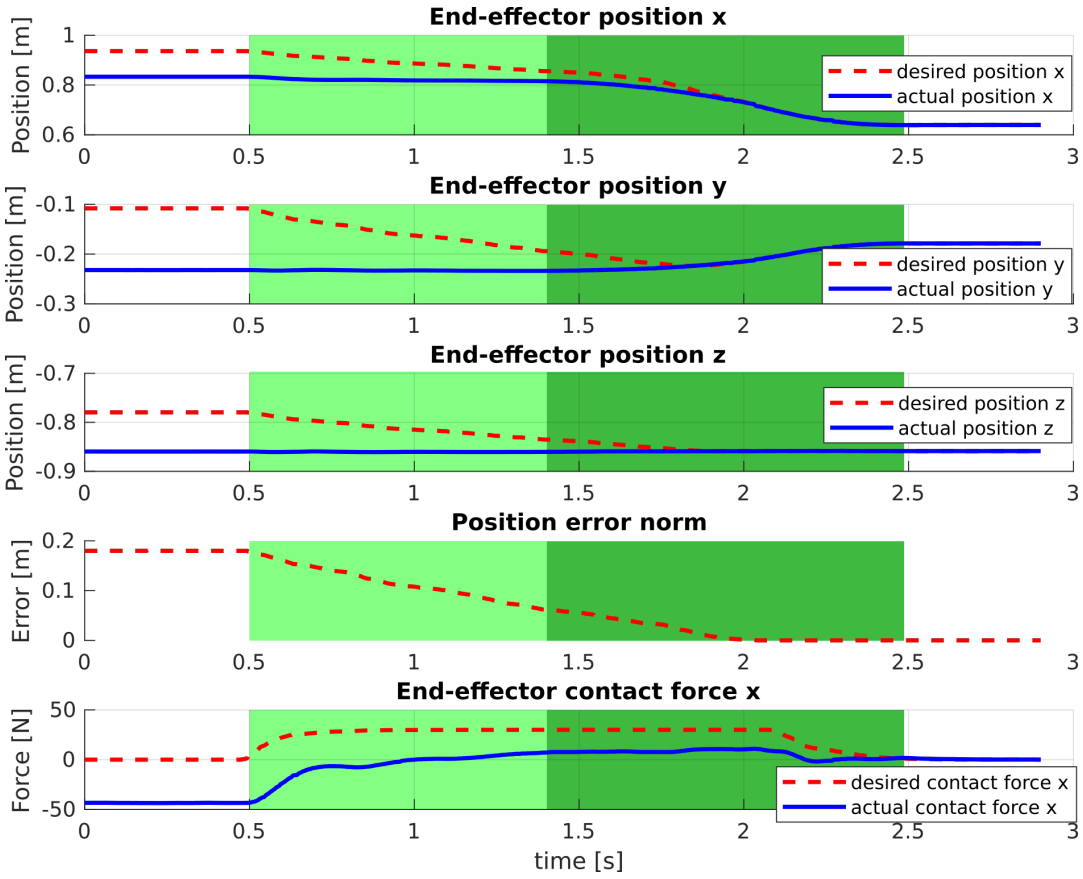


Figure 6.7: Switching smoothly to an end-effector contact force task with a present position tracking error. The top three plots show how the end-effector initially has a position tracking in all three Cartesian directions. As soon as the contact force task is activated (light and dark green boxes) the desired end-effector position is ramped towards the actual end-effector position. This is done in such a way that the total position error norm reduces linearly to zero over 1.5 seconds, even when the actual end-effector position starts to change because of motion of the door (dark green box). The reason that the desired contact force is not being reached is initially because of the conflicting position tracking error. Later it is due to the fact that the end-effector is in motion and results in motion damping action by the motion tracking task.

6.4. Discussion

In this chapter an implementation of the hierarchical optimization control framework is presented which enables real-time contact force control at the end-effector for manipulation purposes. By filtering the desired contact force value input and by ramping down the motion task errors, a smooth transition between the motion and contact force task for the end-effector can be achieved. The placement of the contact force task in the task hierarchy guarantees that the execution of this task is only performed as well is as is possible without corrupting the performance of any of the higher priority tasks. This task hierarchy mainly addresses three of the seven controller requirements listed in Section 2.6. Most importantly it allows for end-effector contact force control, which has not been addressed yet in the previous chapters. Secondly, it allows for simple real-time control of force-based manipulation tasks through online input on the desired contact force direction and magnitude. Finally it can be concluded that the robot's whole-body potential is used. The inclusion of the high-priority EOM task results in the contact force distribution being implicitly solved while taking the full robot dynamics and all joint torques into account.

Interesting to mention is that although the inequality tasks are added at high priority levels, they still need to be added explicitly at every priority level's QP formulation as explained in Chapter 2. Experiments show that when large horizontal contact forces are commanded at the end-effector, that this can result in a conflict with the friction cone task because that task does not allow for the required ground reaction forces. This conflict has shown to sometimes result in unstable behavior with jumps in the computed contact forces that the controller computes for the feet because of activation and deactivation of the friction cone limit tasks

for each of the feet. This behavior can be removed by setting the arm contact force task at a weight factor of 0.0001 with respect to the friction cone task. When a conflict between the two tasks then emerges, almost all focus of the controller will lie on satisfying the friction cone constraints thereby preventing the unstable behavior that originates when the two tasks actively compete at the same weight level.

The results shown here were obtained by a setup where the robot's end-effector was rigidly connected in simulation to a rigid object in the world. Although not very realistic, it served well to show the performance of the controller implementation. In real situations the robot will need to rigidly grasp an object such as a door handle to be able to display similar behavior. Performance might in those cases be limited by backlash in the grip or by a limited gripping strength of the hand. Also, force-based manipulation does not always coincide with a fixed grip on an unmovable object, or by only controlling contact forces without controlling motion. When considering for example a window cleaning task it would be desirable to control contact forces in the direction of the surface, and control motion of the end-effector tangential to the surface. This corresponds to hybrid force-motion control such as first described in [30]. Similar to the methods described in that paper the control framework described in this thesis also allows for decoupled force and motion control. Because both the motion and contact force task can be active at the same time it is possible to prescribe motion of the end-effector in a certain direction, while prescribing contact forces in an orthogonal direction.

For proper execution of motion and contact force tasks, these two tasks should never prescribe action with components in the same direction. Doing so anyway will result in neither of the two tasks being executed properly because it is physically impossible to simultaneously control both motion and force along the same direction. This is also the problem with the current method of velocity limiting control of the end-effector during a contact force task. As explained, upon activation of the contact force task the position error of the end-effector will ramp to zero such that the motion task does not result in the computation of position-regulating joint torques. However, the damping component of the motion task is still left intact. This ensures that when motion of the end-effector is possible, the execution of a contact force task will not result in excessive velocities of the end-effector along that direction. In practice this approach satisfies this requirement. However, the end-effector motion is not exactly damped as prescribed by the motion task because of the existence of the contact force task, while the exerted contact force changes both magnitude and direction as a result of the activation of the damping component of the motion task. It would be more logical to explore ways to modulate the desired contact force in order to regulate the end-effector's velocity, while the motion task is adapted such that it does not result in the computation of additional joint torques. This could for example be done by limiting the amount by which the desired contact force is allowed to differ from the actually measured contact force. This way, when the contacted object is not able to deliver a reaction force and moves instead, the desired contact force is reduced such that high accelerations are prevented. Another option would be to limit the prescribed desired contact force value as a function of the measured end-effector velocity, thereby also creating a damping-like behavior.

Contact force distribution for torque-controlled robots is an ongoing field of research. An influential paper on the topic of contact force distribution for torque-controllable quadrupedal robots suggested the inclusion of inequality constraints that describe the friction cones which need to be respected [42]. Recently literature has focused on solving the contact force distribution problem while taking into account the full system dynamics and desired motions. In [21] a method is presented to combine numerical contact force distribution with an analytical hierarchical inverse dynamics controller. In [10] and [23] a control method similar to the one used in this thesis is used to solve the contact force distribution even during changes in the number of contacts with the environment. However, all these reported methods focus only on computing a contact force distribution that satisfies a set of inequality tasks. In addition, the implementation presented here also takes into account direct contact force control through the implementation of a contact force equality task. Furthermore the work presented here presents a method for smooth transitioning between motion-based and force-based manipulation.

7

Balance disturbance rejection

7.1. Introduction

For a quadrupedal robot without manipulation capabilities it is common to assume that gravity is the only balance disturbing force, and therefore the robot's posture is controlled such that the effects of gravity do not result in loss of balance. This is done by making sure that the position of the robot's center of pressure (COP) remains near the middle of the robot's support polygon, which is formed by the feet in contact with the ground. Having the COP on an edge of the support polygon corresponds with inability of the robot to create a moment to counteract that of gravity, resulting in the robot tipping over. When interacting with the environment through force-based manipulation, the robot is subject to additional external forces. These forces, which can for example result from pushing or puling by the robot on the environment, can result in a shift in position of the COP and therefore in loss of balance. In the previous chapter it was shown how contact forces at the hand could be controlled in real-time, but the effect on the robot's balance was not addressed. In order to examine the possibilities for robustly balanced force-based manipulation a part of this thesis is focused on estimating and effectively dealing with external disturbing forces. Although the relatively large base of support of this quadrupedal platform allows for considerable external forces to be exerted on it, it is still possible for the robot to lose balance as a result of it, or arrive in a state where the margin between being stable and falling becomes undesirably small. Therefore a controller was developed that exploits the mobility of the robot's base and the reasonable tolerance for external forces in order to make the robot more robust to large external forces.

The controller presented in this chapter is developed to generate a desired COM position reference for the COM motion task. This position reference is computed based on the measured balance-disturbing effects of an external force. First, contact forces at the feet are estimated using the leg joint torques, because the actuators provide accurate torque readings, while no accurate contact force measurements are available directly at the feet. Based on these measured contact forces the position of the COP of the robot is computed. When this position deviates from the position where the COP is expected to be based on the position and acceleration of the COM, an external force must be acting on the robot. Finally, this difference between the expected and actual COP position is used to command a shift in the COM position such that the actual COP remains near the middle of the robot's support polygon. This way the robot reactively maintains balance when large external forces are exerted on the robot, either due to a deliberate manipulation task, or due to unexpected external disturbances such as pushes to the robot.

7.2. Controller setup

This section will present the exact design of the components of the disturbance rejection controller. First, a method to estimate contact forces based on joint torques is presented. Second a control law is presented which uses information on the state of COM and the position of the COP to estimate and correct for external

disturbing forces.

Knowledge of the position of the COP is crucial to assessing the robot's state of balance. Therefore the contact forces at the feet need to be known. The ANYmal robot does currently not possess accurate contact force sensors at the feet, but does have actuators which provide very accurate torque readings. The torque readings, in combination with part of the robot's EOM can be used to estimate the contact forces at the feet. To do this, assumptions can be made in order to prevent doing full dynamic state estimation of the robot, which can be complex and sensitive to producing errors. The assumption made here is that each of the four legs is attached at the hip to a fixed point in space instead of to a floating base. This assumption means that any joint torques $\boldsymbol{\tau}$ would only result in either leg joint (lj) accelerations $\dot{\mathbf{u}}_{lj}$, compensation for dynamic effects at the legs $\mathbf{h}_{lj}$, and contact forces $\boldsymbol{\lambda}$. Computationally this can be represented by extracting the leg joint-related parts of the full EOM and rewriting the equation to solve for the contact forces:

$$\boldsymbol{\lambda} = \mathbf{J}_{s,lj}^{T+}\left(-\boldsymbol{\tau} + \mathbf{M}_{lj}\dot{\mathbf{u}}_{lj} + \mathbf{h}_{lj}\right) \tag{7.1}$$

Here $\mathbf{J}_{s,lj}$ is the part of the support Jacobian that relates only the leg joint velocities to foot velocities, and $\mathbf{M}_{lj}$ is the 12×12 square part of the mass matrix that corresponds to the robot's leg joints. With this knowledge of the actual contact forces at the feet, the position of the robot's COP ($\mathbf{x}_{COP}$) can be computed as the sum of the positions of the feet, weighted by the vertical component of their respective contact forces.

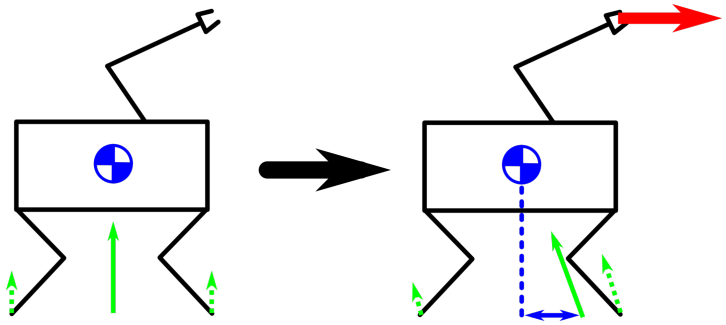


Figure 7.1: When the robot is not moving, its COP will lie underneath the COM, which is typically controlled to remain above the center of the support polygon such that the COP will be there too (left illustration). When an external force acts on the robot (red vector in right illustration), the robot's COP will shift away from the projection of the COM onto the ground. This shift of the COP towards one of the edges of the support polygon decreases the balance margin of the robot.

When the robot is in rest and gravity and the ground reaction forces at the feet are the only external forces acting on the robot, then the COP will be below the robots COM. This state is depicted in the left panel of Figure 7.1. Therefore, the COM is typically controlled to remain above the center of the support polygon, such that the COP also remains near the center of the support polygon, resulting in a stable configuration. However, when there are additional external forces acting on the robot, such as indicated by the red vector in the right panel of Figure 7.1, the COP position will shift and will no longer lie beneath the robot's COM. To compensate, the COM position should move such that the COP position will move towards the center of the support polygon again, for optimal balance.

The controller presented here assumes that there is already a reference position for the COM which guarantees balance when no disturbing forces are acting on the robot. When an error $\mathbf{e}_{COP}$ is detected between where the expected and actual COP positions (blue arrow in Figure 7.1), then an offset for the desired COM position reference is computed which compensates for this error and brings the COP back towards a position which corresponds with stability of the robot. To define the expected COP position, the zero-moment point (ZMP) is used [50]. This is the point on the ground with respect to which the moment of the inertial and gravitational forces has no component along the horizontal axis. This point, which can be expressed with the robot's state, is proven to coincide with the robot's COP when it falls within the robot's support polygon and no other forces are acting on the robot. In [51] it is shown that for a quadrupedal robot the system can be modelled as point-mass to simplify the computation of the ZMP. The computation of the ZMP in 2-D (in this case along the x-axis) can then be done according to:

$$x_{ZMP} = x_{COM} - \frac{z_{COM}\ddot{x}_{COM}}{\ddot{z}_{COM} + g} \tag{7.2}$$

where x and z indicate positions along the x-, and z-axis respectively. The y-coordinate of the ZMP can be computed similarly. Note that when the robot is in rest, and $\ddot{x}_{COM}$ and $\ddot{z}_{COM}$ are equal to zero and x_{ZMP} becomes equal to x_{COM}. Previous research has shown that planning COM motion trajectories based on this model (7.2) results in actual COP positions close to the planned ZMP positions, such that they fall inside the predefined support polygons during various quadrupedal walking gaits [51][11].

Using this definition of the ZMP, the error between the expected and actual position of the COP can then be computed as $\mathbf{e}_{COP} = \mathbf{x}_{ZMP} - \mathbf{x}_{COP}$. This gives a measure of the balance disturbing effect of any external force or moment on the robot. A heavy first-order filter is applied to $\mathbf{e}_{COP}$ in order to have the controller ignore very high-frequency variations in the ZMP and COP signals. With a filter constant $\alpha = 0.99$ and a control period $\delta t = 0.0025$ s, the cop error is computed as:

$$\mathbf{e}_{COP}(t) = \alpha \mathbf{e}_{COP}(t - \delta t) + (1 - \alpha)(\mathbf{x}_{ZMP}(t) - \mathbf{x}_{COP}(t)) \tag{7.3}$$

Based on this cop error $\mathbf{e}_{COP}$ a shift $\boldsymbol{\delta}\mathbf{r}_{COM}$ of the desired COM position is calculated with the following control law:

$$\boldsymbol{\delta}\mathbf{r}_{COM} = k_p \max(0,\ \|\mathbf{e}_{COP}\| - \theta_{COP})\ \frac{\mathbf{e}_{COP}}{\|\mathbf{e}_{COP}\|} \tag{7.4}$$

where θ_{COP} is the cop error threshold after which the controller becomes active, which is in this case set to 0.02 m. This results in a behavior where the robot retains its original compliant behavior for small external forces, and only starts actively compensating for external forces once they get larger. Besides that, in combination with the low-pass filter in (7.3) it ensures that the controller will also not become activated when high but only short-lasting disturbing forces are exerted on the robot.

7.3. Results

The effectiveness of the described method for contact force estimation is examined with the following experiment. While standing still a forward and then backward pitch change of the robot's torso is commanded while estimating and measuring the contact forces at the feet. Figure 7.2 shows the results of this experiment. In the top panel the red dotted line shows the vertical component of the contact force at the front left foot which is measured in the simulation software. The solid blue line shows the estimated vertical contact force at the same foot. The bottom panel shows the desired (red dotted) and actual (blue solid) torso pitch angle. In order to generate the angular acceleration of the torso, the contact forces at the foot change over time. It can be seen that the estimated contact forces are a good approximation of the actually measured contact forces. It is important to note that in this experiment the values of the joint torques which are used for the estimation are the known joint torques which are applied by the simulation software. In practice the data of the measured actuator torques might be less representative of the actual torques. However, the ANYdrive actuators in the robot's legs have a relatively high torque measurement resolution and have shown precise torque tracking [14]. Some preliminary experiments on the actual robot have shown that this contact force estimation method does indeed appear to yield very useable results.

In order to demonstrate the behavior of the COP-controller, an experiment is set up such as displayed in Figure 7.3. An unmovable door which functions as a rigid wall is added to the simulation. The robot's hand is connected rigidly to it in simulation. The green vector between the feet represents the COP and the sum of contact forces measured at the feet. The middle panel shows that when the COP-controller is not active and a pulling force is commanded (indicated by the horizontal green vector), the robot's COP shifts towards the front edge of the support polygon, while the robot retains its posture. The right panel shows the situation where the COP-controller is activated. An error between the COP and ZMP is detected and as a result the COM is commanded to move (indicated by the blue vector). Because of this COM motion the COP moves towards the center of the support polygon again, resulting in a more stable position for the robot under influence of the external force.

A detailed overview of the results from this experiment is shown in Figure 7.4. The plots on the left and right plots respectively show the robot's behavior without and with the COP-controller being implemented. For both cases the desired contact force at the hand is linearly ramped up over 1 second to a value of 40 N. This value remains constant for 2 seconds after which the desired contact force is ramped down to zero 0 N again. When the COP-controller is not implemented the COP of the robot shifts forward by almost 10 cm.

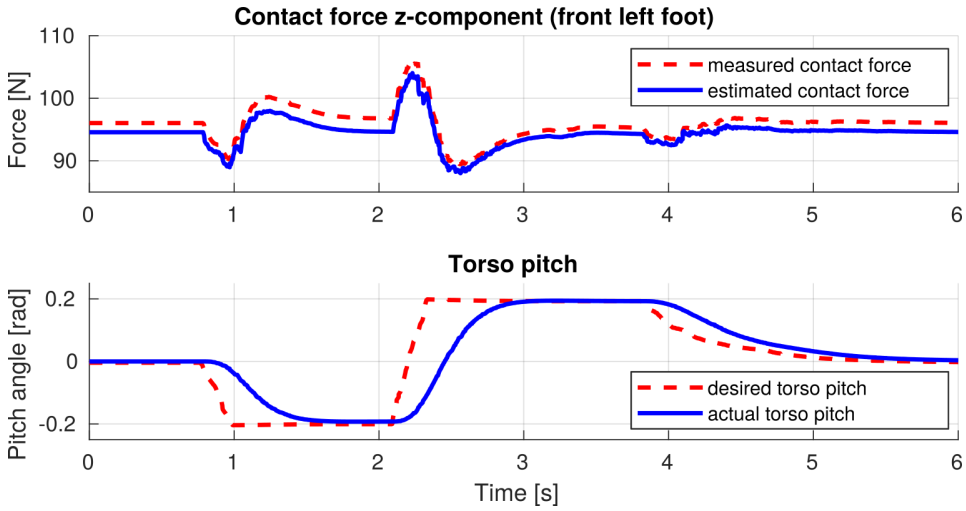


Figure 7.2: Contact force estimation (at the front left foot) during torso motion. While the torso is commanded to pitch forward and then backward, the contact forces at the feet change. The top plot shows that the contact forces estimated based on joint torques are highly similar to the actually measured contact forces.

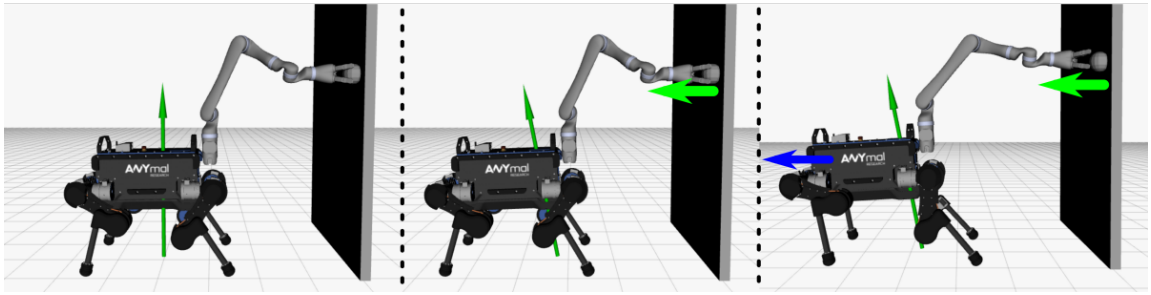


Figure 7.3: Exerting contact forces with the end-effector on a rigid object without and with COP-controller. The leftmost picture shows the robot's end-effector in rigid contact with the environment without exerting forces. The robot's COP, indicated by the origin of the green vector between the legs, remains underneath the COM. The middle picture shows the robot pulling on the environment (horizontal green arrow). As a result, the COP shifts towards the front of the support polygon. In the rightmost picture the COP-controller is active. Because an error between the robot's ZMP and COP are detected, the COM is commanded to move (blue vector) such that the COP moves towards the center of the support polygon.

During this the COM, and therefore also the ZMP, stay at a constant position. The plots on the right show that when the COP-controller is implemented, the error between the COP and ZMP results in a shift of the desired and actual COM position. As the COM moves, the COP moves back closer to its original position at 0.0 m which is in the middle of the support polygon. Because the COM shows no considerable accelerations, the ZMP x-position corresponds strongly with the COM x-position. When the contact force at the hand is lowered again the COP temporarily overshoots the middle of the support polygon because the COM is still shifted backwards with respect to the polygon center. Finally, while the external force at the hand goes back to zero, the error between the COP and ZMP position also goes to zero, resulting in the desired COM position to move back to its original position above the center of the support polygon. A clear difference between the two situations can be seen for the maximum distance of the COP from the support polygon's center. When the COP-controller is active, an equal external force at the hand results in almost a 50% smaller maximum deviation of the COP, thereby keeping the robot in a more stable configuration.

7.4. Discussion

This chapter showed the design and implementation of a controller that estimates the balance-disturbing effects of external forces based on estimated foot contact forces, and controls the desired COM position such that these balance-disturbing effects are mitigated. This works to compensate for balance-disturbing effects

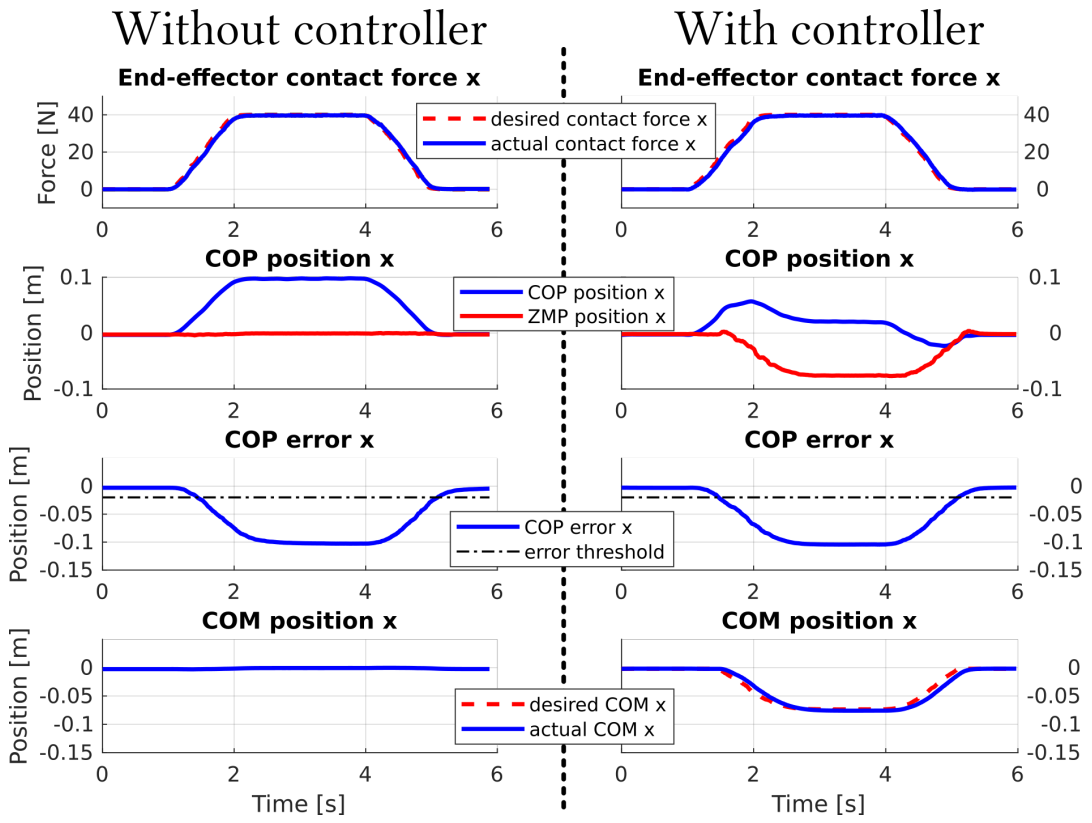


Figure 7.4: Response of the robot to external forces without (left) and with (right) the COP-controller. For both situations the end-effector is connected to the environment while over a time of 1 s the contact force is ramped up to 40 N and then ramped back down to 0 N. When the COP-controller is not implemented, this external force results in a considerable shift of the COP while the COM remains at the same place. When the COP-controller is activated, the COP error triggers a shift in desired COM position, which then results in the COP shifting back towards it's original stable position.

caused by any planned or unplanned external forces that are exerted on the robot. It specifically generates elegant behavior when external contact forces are actively commanded between the hand and the environment. In this case all of the robot's DOFs are used to adapt the posture of the robot such that balance is improved while the desired contact forces are generated at the hand. Of the seven control requirements listed in Section 2.6, two are specifically addressed. First, the controller presented in this chapter is focused on maintaining the robot's balance, even under large external disturbing forces. Second, the method described here is reactive in nature and therefore still allows simple real-time teleoperation of the desired contact forces at the end-effector.

The large base of support of the robot allows for the small initial disturbance of balance which then triggers this balance-restoring posture change. Therefore it is possible to remotely and in real-time command relatively high desired contact forces at the hand, without requiring complex and delaying planning methods to ensure balance of the robot. Similar behavior and teleoperation would not be possible for bipedal robots, which due to their typically small base of support require careful planning of COM motion to retain balance. Without careful planning, real-time contact force control at an arm would then quickly result in the contact force pushing or pulling the robot into a fall if locomotion is not initiated. Most wheeled mobile manipulators have a considerably larger base of support than humanoid platforms. However, they often lack to ability to move the robot's base/torso with respect to the contact points on the floor. This severely limits the possibilities to adapt the COM position to adapt for the balance-disturbing effect of external forces. In [9] it is reported that wheeled mobile manipulators, although expected to be inherently stable, are still prone to falling as a result of force-based interaction with the environment.

In [48] a method similar to the one in this chapter is presented for the balance control of a humanoid robot. It depends on integrating the error between the desired and actual COP position of the robot and using that as a measure for shifting the COM position accordingly. However, it is only shown to be effective in keeping

the robot balanced when small masses are slowly incrementally added to the robot's hand. The same research group later reports for a similar method applied on a humanoid robot that its effectiveness is limited to handling only small external forces, or slowly changing larger forces [19]. Besides, the risk for oscillating responses is mentioned due to not taking into account the robot's dynamics [48]. The method described in this chapter does take some of the system dynamics into account by using the ZMP in the control law. This term, which takes into account the accelerations of the COM, will result in a smaller control action when part of the registered COP shift can be attributed to COM acceleration instead of a reaction to external forces. Furthermore, the inclusion of the ZMP allows for a fast response to new external forces. This is because such a force will first result in acceleration of the COM, which is reflected by the ZMP position. Only after position errors have built up, the COP will start to shift to generate the forces required to fight the position errors.

The COP-controller shown here is a simple approach that generates balance restoring COM motion based on the symptoms of balance-disturbance which is generated by external forces and moments. This has shown to be effective and illustrated the ability of the robot the improve its balance reactively when large external disturbances are present. However, improvements could be made by making use of an external disturbance observer such that appropriate balance restoring motion can be computed based on the actual position and nature (force vs torque) of the external disturbance.

Like previous chapters, the implementation shown in this chapter has only been demonstrated while assuming the robot standing still. In reality, the robot will often be walking, and some force-based tasks such as fully opening a door are not even possible without the robot taking a few steps. Adapting the robot's walking behavior to external forces will be a considerably more challenging task. Currently COM motion of a quadruped during walking is typically computed through planning of a ZMP trajectory through the upcoming support polygons of the next steps [28][51][11]. When an external force is detected then this COM motion trajectory could for example be given an offset similar to the way done in the here presented COP-controller. Alternatively the positions of the footholds could be shifted instead. Finally, the COM motion and foothold positions could be designed such that a relatively large margin of error of the COP is allowed along the direction of the perceived external force before the COP reaches one of the edges of the support polygon. This would be a very interesting topic for future work.

8

Discussion

The objective of this thesis was to explore specific implementations of a hierarchical optimization-based inverse dynamics controller for the control of a quadrupedal manipulator. This was done with a focus on combining the benefits of both the control framework as well as those of the robotic platform, such that the list of specific requirements presented in Section 2.6 could be met. In the individual Discussion sections of the previous chapters it was discussed to what extent the prespecified objectives were met by the presented controller implementations, and in which direction improvements can be made in future work. This chapter will provide a general discussion of this thesis, its results, limitations, applications and topics of interest for future research.

Besides a general implementation of the hierarchical optimization controller, several other extended versions of this basic implementation were presented. Each of these extended implementations focused on meeting one or more of the specific control requirements that were identified for this robotic platform. Combined, the behavior of these specific implementations cover the entire list of requirements, being: end-effector pose control; end-effector contact force control; maintaining balance under external disturbances; exploiting whole-body potential; avoiding/handling singularities; collision avoidance; simple real-time teleoperation. The fact that these approaches followed a similar logic, such as rating end-effector and COM motion control as higher priority than torso orientation control, enables all the individually presented controller implementations to be combined into a single task hierarchy, which is shown in Figure 8.1. This is a combination of the basic implementation (Chapter 3), the joint limit-based whole-body reaching implementation (Chapter 4), the posture optimization implementation (Chapter 5), and the force-based manipulation implementation (Chapter 6). The functionalities of the individual controller implementations do not necessarily conflict and therefore combining these implementations is possible in order to combine the benefits that each of these implementations offer individually.

In the previous chapters implementation-specific limitations were discussed. Besides these, there are also several global limitations that should be mentioned. Firstly, the experiments presented in this report have all been conducted in simulation. Previous results from this physics simulation in combination with an accurate dynamic model of the ANYmal robot have shown good transferability to behavior on the actual robot. However, it is unknown if this would also be the case for the here presented robotic platform consisting of the ANYmal robot in combination with a 6-DOF Kinova arm. Possible complications for implementation on a physical robot include inaccuracies in the dynamic and actuator models, sensor data and actuator torque control. Besides that, other unmodelled consequences could complicate implementation on the actual robot, such as actuator heat accumulation as a result of high torque demands.

Secondly, an important factor that has not been addressed in this work is that of the computational demands that the hierarchical optimization control framework brings. Compared to other types of multi-task operational-space controllers, the control framework used in this thesis offers several benefits, but is also one of the most computationally expensive. This is due to the fact that it allows for a hierarchy of priority levels, and solves for the specified tasks numerically as a opposed to analytically. Indeed, various authors mention

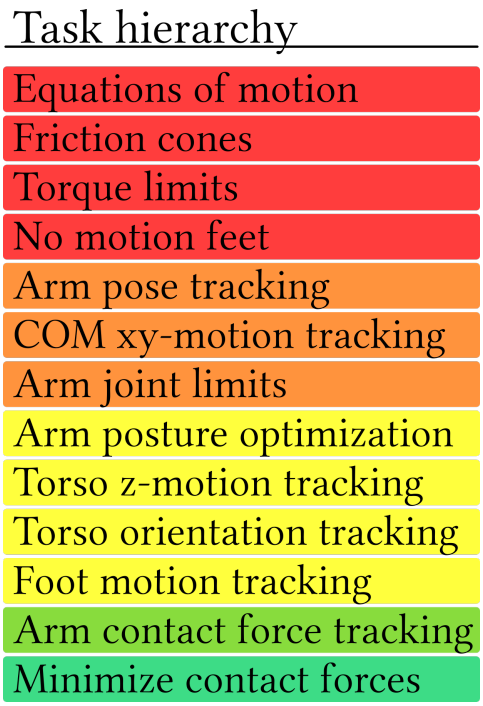


Figure 8.1: Combined task hierarchy containing all individual implementations presented in this thesis. This task hierarchy allows for whole-body reaching motion, arm posture optimization, and end-effector contact force-control.

the need for computationally fast algorithms to make implementation of this type of optimization-based whole-body control feasible for the control of a high-DOF robot at a desirable ms-level control rate [17][22]. The reason that computational demands were not explicitly considered in this thesis is because the focus was chosen to be on adapting the control framework to generate useful behavior of the robot. Furthermore, the computer on which the experiments in this thesis were performed lacked the computational strength to run the control framework at speeds that the embedded computer on the actual robot would be able to achieve. However, the challenge of keeping the controller computationally tractable was not ignored. As mentioned in [23] computation time mainly depends on the number of DOFs of the robot, and the size and composition of the task hierarchy. Previous literature reports on successfully implementing the framework on physical robots, such as the 400 Hz control of a 16 DOF quadruped robot [10], and the 1000 Hz control of a 14 DOF bottom half of a humanoid robot [22]. The latter research group also mentions attaining maximum control periods of only 3 ms for the control of a 25 DOF full humanoid in simulation. These results suggest that real-time control of a 18 DOF quadrupedal manipulator should also be feasible, especially considering ongoing improvements in computer processing power and algorithmic efficiencies. For the design of the various task hierarchies presented in this thesis special attention has been payed to prevent unnecessary computational load. This has been done by grouping tasks on priority levels wherever possible in order to arrive at a minimum number of priority levels. Furthermore, the total DOF count of all tasks combined was kept to a minimum by preventing conflicting tasks and additional inequality tasks which are not strictly necessary. This does however not guarantee that extra adaptations will not be necessary in order to get the presented individual or combined task hierarchies running in real-time on the actual robot.

The presented controller implementations have all been tested in experiments to show that they are functional and achieve desirable behavior of the robot. However, because of the exploratory nature of this thesis, non of these implementations have been investigated truly in-depth. The reason for the exploratory nature of this thesis is that the field of real-time whole-body control for fully torque-controllable legged manipulators is currently still in development. To date only few fully torque-controllable platforms exist, and research on them is mostly not yet focused on general mobile manipulation, but instead on specific topics such as contact force distribution, balance, or locomotion [21][23][10]. Research on whole-body control of a torque-controllable quadrupedal manipulator has to date not been published.

There are two main topics identifiable as general topics for future work on the control of a quadrupedal manipulator. First is the topic of online model estimation. The control framework presented in this thesis is highly dependent on an accurate dynamic model of the robot. It depends on this model to compute accurate joint torques to achieve compensation for dynamic effects such as gravity and Coriolis forces, and to achieve accurate inverse dynamics-based motion tracking. However, the robot is equipped with manipulation capabilities in order to interact with the environment. A likely task for the robot would be to pick up, carry and/or handle objects in its environment. This adds unmodeled inertia to the system, which can result in serious performance deterioration [35]. One simple way of dealing with additionally carried mass is to command a desired contact force at the end-effector to compensate for the gravity acting on the object [48]. The drawbacks of this approach are that the additional mass is not taken into account by the balance-preserving COM motion task and that the effects of the object's inertia during relatively fast arm motions are ignored. Furthermore, it requires the addition of the arm contact force task to the task hierarchy, which adds extra complexity and computational load to the controller. An alternative would be to update the robot's dynamic model. This could be done either be done through e.g. visual identification of the object when the object's inertial properties are known, or through online model estimation of the object. The latter approach would offer the benefit of being more generally applicable. Future research will indicate whether it would suffice to only update the gravity compensation terms in the EOM, or whether the translational or even rotational inertias of the carried object should be taken into account.

The second, and most important topic for future research is that of locomotion. To create a manageable scope, this thesis has focused only on standing manipulation. However, in order to exploit the excellent mobility capabilities of the robot, special attention will have to be payed to integrate the robot's legged locomotion with its manipulation capabilities. When locomotion is the only objective, then this should not be a considerably more difficult problem for a quadrupedal manipulator compared to only a quadrupedal base. Indeed, some exploratory experiments during the research for this thesis showed that the methods previously used for locomotion on the ANYmal robot, could relatively easily be adapted to also generate legged locomotion for the quadrupedal manipulator. However, the real challenge lies in coordinating locomotion with the robot's manipulation capabilities in order to greatly improve the robot's manipulation performance. This can be done on two levels: motion and force. In terms of motion a smart algorithm could for example be be used to trigger appropriate locomotion when this is needed to increase the arm's workspace reachability. This way the robot's torso can almost literally be viewed as a free-floating base that can be commanded to stay at a certain position relative to the arm's end-effector. More complex approaches could focus on using the manipulator directly for locomotion control by engaging in additional contact with the environment, or by using the manipulator for balance in a tail-like fashion by commanding changes in the angular momentum. In terms of force a different challenge lies ahead for future research. Current methods for quadruped locomotion assume the only forces acting on the robot to be gravity and the ground reaction forces, or short balance-disturbing pushes. When manipulation tasks require high forces on the environment, such as during the opening of a spring-loaded door, these locomotion controllers are likely to fail in keeping the robot balanced under the effect of the external forces. In that case it will be necessary to develop methods that generate locomotion behavior which is adapted to the continuous external forces acting on the robot.

The controller implementations presented in this thesis were specifically designed for the quadrupedal manipulator introduced in Chapter 2. However, this knowledge can be assumed to be generally applicable on most quadrupedal manipulators. First, most quadrupedal robots share the morphology of possessing 3 DOF legs with point-feet, e.g. [24][26][28][44][6]. In order to give the manipulator full control over its pose, it is an obvious choice to give it 6 DOF, such as is also done in [40]. When more DOFs are present in any of the limbs such as in the 4 DOF legs of BigDog [8] then an additional task can be specified, such as orientation tracking or posture optimization to resolve this redundancy. In case fewer DOFs are present, such as in the 5 DOF arm on SpotMini [6], the dimension of the task set should be reduced accordingly. For this example the choice could be made to formulate a 3 DOF translational motion tracking task, but only a 2 DOF orientation tracking task.

9

Conclusion

The objective of this thesis was to implement an optimization-based hierarchical inverse dynamics control framework for the control of a simulated torque-controlled quadrupedal manipulator, with a specific focus on the design of prioritized sets of control tasks which allow to exploit the characteristic benefits offered by both the control framework and robotic platform. For the design of the sets of control tasks special attention was paid to a set of control objectives that were identified specifically for this robotic platform. Initially a basic controller implementation is presented which satisfies the most crucial control requirements such as end-effector motion control and maintaining balance by controlling the position of the robot's center of mass. Subsequently three extended controller implementations are presented. First a method is presented that depends on the inclusion of kinematic limit tasks and a specific task hierarchy in order to prevent kinematic singularities and self-collision, and trigger whole-body reaching motion when only arm motion is not sufficient to reach the desired end-effector pose. Secondly an implementation is presented which is focused on using motion of some of the torso's DOFs to optimize the arm's posture according to a desired criterion. It is shown that this can for example be used to avoid kinematic singularities, self-collision, and to let whole-body reaching motion emerge. Thirdly a controller implementation is presented which enables active contact force control at the end-effector for force-based manipulation, while implicitly solving the dynamic contact force distribution problem. It is demonstrated how such contact force control can be used to perform tasks such as opening a door, which are known to be complicated for traditional position-controller robots. Finally, in addition to that implementation, a controller is presented which enables the robot to detect and reactively respond to external forces acting on the robot in order to mitigate the balance-disturbing effects of these forces. These different controller implementations do not exclude each other and allow to be implemented simultaneously in order to combine the individual benefits that they offer. Furthermore all these implementations still allow for simple real-time teleoperation such as controlling the end-effector motion or contact force with a joystick. These results show that these sets of control tasks allow for completion of all of the specified control objectives which are listed in Section 2.

It can be concluded that the investigated whole-body control framework can be successfully used for the control of a simulated quadrupedal manipulator. Furthermore, using carefully designed sets of prioritized tasks appears to be a promising approach for exploiting the potential benefits that a torque-controllable quadrupedal manipulator offers. In particular, it is shown that the motion control of the torso's orientation and height can be relaxed in order to create kinematic redundancy in the system. These additional degrees of freedom can then be used by other control tasks. Because of the exploratory nature of this research the controller implementations presented in this report have not been individually investigated in-depth and can benefit from further research for which various suggestions are made. Most notably this includes the investigation of the transferability of the controller implementations presented here for simulation to the control of a physical robot.

Bibliography

[1] ANYdrive, https://www.anybotics.com/anydrive/, last accessed: 2-3-2018.

[2] Gazebo simulator, http://gazebosim.org, last accessed: 2-3-2018.

[3] Kinova Jaco2 6DOF-S, http://www.kinovarobotics.com/innovation-robotics/products/robot-arms/#jaco1, last accessed: 2-3-2018, .

[4] Kinova actuators, http://www.kinovarobotics.com/innovation-robotics/products/actuators/, last accessed: 2-3-2018. .

[5] Rethink Robotics: Sawyer, http://www.rethinkrobotics.com/sawyer/, last accessed: 28-01-2018.

[6] Boston Dynamics: SpotMini, https://www.bostondynamics.com/spot-mini, last accessed: 28-01-2018.

[7] LBR iiwa, https://www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa, last accessed: 28-01-2018.

[8] Yeuhi Abe, Benjamin Stephens, Michael P Murphy, and Alfred A Rizzi. Dynamic whole-body robotic manipulation. In *Unmanned Systems Technology XV*, volume 8741, page 87410V. International Society for Optics and Photonics, 2013.

[9] Christopher G Atkeson, B P W Babu, N Banerjee, D Berenson, C P Bove, X Cui, M DeDonato, R Du, S Feng, and P Franklin. What happened at the darpa robotics challenge, and why. *DRC Finals Special Issue of the Journal of Field Robotics*, 1, 2016.

[10] C Dario Bellicoso, Christian Gehring, Jemin Hwangbo, Péter Fankhauser, and Marco Hutter. Perception-less terrain adaptation through whole body control and hierarchical optimization. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 558–564. IEEE, 2016. ISBN 1509047182.

[11] C Dario Bellicoso, Fabian Jenelten, Peter Fankhauser, Christian Gehring, Jemin Hwangbo, and Marco Hutter. Dynamic Locomotion and Whole-Body Control for Quadrupedal Robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2017)*. IEEE, 2017.

[12] Arjun Bhasin, Rekha Raja, and Ashish Dutta. Non-holonomic mobile manipulator kinematic control using hybrid simulated annealing. In *Electrical and Computer Engineering (WIECON-ECE), 2015 IEEE International WIE Conference on*, pages 435–438. IEEE, 2015. ISBN 146738786X.

[13] Michael Bloesch, Hannes Sommer, Tristan Laidlow, Michael Burri, Gabriel Nuetzi, Péter Fankhauser, Dario Bellicoso, Christian Gehring, Stefan Leutenegger, and Marco Hutter. A primer on the differential calculus of 3D orientations. 2016.

[14] Karen Bodie, C Dario Bellicoso, and Marco Hutter. ANYpulator: Design and control of a safe robotic arm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1119–1125. IEEE, 2016. ISBN 1509037624.

[15] Alexander Dietrich, Thomas Wimbock, Alin Albu-Schaffer, and Gerd Hirzinger. Integration of reactive, torque-based self-collision avoidance into a task hierarchy. *IEEE Transactions on Robotics*, 28(6):1278–1293, 2012. ISSN 1552-3098.

[16] Johannes Englsberger, Alexander Werner, Christian Ott, Bernd Henze, Maximo A Roa, Gianluca Garofalo, Robert Burger, Alexander Beyer, Oliver Eiberger, and Korbinian Schmid. Overview of the torque-controlled humanoid robot TORO. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 916–923. IEEE, 2014. ISBN 147997174X.

[17] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028, 2014. ISSN 0278-3649.

[18] Péter Fankhauser, Marko Bjelonic, Dario Bellicoso, Takahiro Miki, and Marco Hutter. Robust Rough-Terrain Locomotion with a Quadrupedal Robot. In *IEEE International Conference on Robotics and Automation (ICRA 2018)*. ETH Zurich, 2018.

[19] Siyuan Feng, X Xinjilefu, Christopher G Atkeson, and Joohyung Kim. Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 1028–1035. IEEE, 2015. ISBN 1479968854.

[20] Michele Focchi, Andrea Del Prete, Ioannis Havoutis, Roy Featherstone, Darwin G Caldwell, and Claudio Semini. High-slope terrain locomotion for torque-controlled quadruped robots. *Autonomous Robots*, 41(1):259–272, 2017. ISSN 0929-5593.

[21] Bernd Henze, Máximo A Roa, and Christian Ott. Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios. *The International Journal of Robotics Research*, 35(12):1522–1543, 2016. ISSN 0278-3649.

[22] Alexander Herzog, Ludovic Righetti, Felix Grimminger, Peter Pastor, and Stefan Schaal. Experiments with a hierarchical inverse dynamics controller on a torque-controlled humanoid. *Available online: http://arxiv.org/abs/1305.2042*, 2013.

[23] Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludovic Righetti. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, 40(3):473–491, 2016. ISSN 0929-5593.

[24] Marco Hutter, Christian Gehring, Michael Bloesch, Mark A Hoepflinger, C David Remy, and Roland Siegwart. StarlETH: A compliant quadrupedal robot for fast, efficient, and versatile locomotion. In *Adaptive Mobile Robotics*, pages 483–490. World Scientific, 2012.

[25] Marco Hutter, Hannes Sommer, Christian Gehring, Mark Hoepflinger, Michael Bloesch, and Roland Siegwart. Quadrupedal locomotion using hierarchical operational space control. *The International Journal of Robotics Research*, 33(8):1047–1062, 2014. ISSN 0278-3649. doi: 10.1177/0278364913519834.

[26] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, and Michael Bloesch. Anymal-a highly mobile and dynamic quadrupedal robot. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 38–44. IEEE, 2016. ISBN 1509037624.

[27] Dominic Jud, Gabriel Hottiger, Philipp Leemann, and Marco Hutter. Planning and Control for Autonomous Excavation. *IEEE Robotics and Automation Letters*, 2(4):2151–2158, 2017. ISSN 2377-3766.

[28] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry, and Stefan Schaal. Fast, robust quadruped locomotion over challenging terrain. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2665–2670. IEEE, 2010. ISBN 1424450381.

[29] Shinji Kawatsuma, Mineo Fukushima, and Takashi Okada. Emergency response by robots to Fukushima-Daiichi accident: summary and lessons learned. *Industrial Robot: An International Journal*, 39(5):428–435, 2012. ISSN 0143-991X.

[30] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987. ISSN 0882-4967.

[31] Twan Koolen, Sylvain Bertrand, Gray Thomas, Tomas De Boer, Tingfan Wu, Jesper Smith, Johannes Englsberger, and Jerry Pratt. Design of a momentum-based control framework and application to the humanoid robot atlas. *International Journal of Humanoid Robotics*, 13(01):1650007, 2016. ISSN 0219-8436.

[32] Jeongsoo Lim and Jun-Ho Oh. Backward ladder climbing locomotion of humanoid robot with gain overriding method on position control. *Journal of Field Robotics*, 33(5):687–705, 2016. ISSN 1556-4967.

[33] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994. ISBN 0849379814.

[34] Keiji Nagatani, Tomonobu Hirayama, Akio Gofuku, and Yutaka Tanaka. Motion planning for mobile manipulator with keeping manipulability. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 2, pages 1663–1668. IEEE, 2002. ISBN 0780373987.

[35] Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, 2008. ISSN 0278-3649.

[36] Hae-Won Park, Patrick M Wensing, and Sangbae Kim. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. 2015. ISSN 2330-7668.

[37] Jonghoon Park, Wankyun Chung, and Youngil Youm. Computation of gradient of manipulability for kinematically redundant manipulators including dual manipulators system. *Transactions on Control, Automation and Systems Engineering*, 1(1):8–15, 1999.

[38] Oscar Efrain Ramos Ponce. Generation of the whole-body motion for humanoid robots with the complete dynamics, 2014.

[39] Gill Pratt and Justin Manzo. The darpa robotics challenge. *IEEE Robotics & Automation Magazine*, 20(2): 10–12, 2013. ISSN 1070-9932.

[40] Bilal Ur Rehman, Michele Focchi, Jinoh Lee, Houman Dallali, Darwin G Caldwell, and Claudio Semini. Towards a multi-legged mobile manipulator. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3618–3624. IEEE, 2016. ISBN 1467380261.

[41] Bilal Ur Rehman, Darwin G. Caldwell, and Claudio Semini. Centaur Robots-a Survey. In *Human-centric Robotics-Proceedings Of The 20th International Conference Clawar 2017*, page 247. World Scientific, 2017. ISBN 981323105X.

[42] Ludovic Righetti, Jonas Buchli, Michael Mistry, Mrinal Kalakrishnan, and Stefan Schaal. Optimal distribution of contact forces with inverse-dynamics control. *The International Journal of Robotics Research*, 32(3):280–298, 2013. ISSN 0278-3649.

[43] Claude Samson, Bernard Espiau, and Michel Le Borgne. *Robot control: the task function approach*. Oxford University Press, 1991. ISBN 0198538057.

[44] Claudio Semini, Jonas Buchli, Marco Frigerio, Thiago Boaventura, Michele Focchi, Emanuele Guglielmino, Ferdinando Cannella, Nikos G Tsagarakis, and Darwin G Caldwell. HyQ-A dynamic locomotion research platform. In *International Workshop on Bio-Inspired Robots, Nantes (France)*, 2011.

[45] Luis Sentis. *Synthesis and control of whole-body behaviors in humanoid systems*. Stanford university USA, 2007. ISBN 0549246045.

[46] Luis Sentis and Oussama Khatib. Control of free-floating humanoid robots through task prioritization. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1718–1723. IEEE, 2005. ISBN 078038914X.

[47] Anthony Stentz, Herman Herman, Alonzo Kelly, Eric Meyhofer, G Clark Haynes, David Stager, Brian Zajac, J Andrew Bagnell, Jordan Brindza, and Christopher Dellin. Chimp, the cmu highly intelligent mobile platform. *Journal of Field Robotics*, 32(2):209–228, 2015. ISSN 1556-4967.

[48] Benjamin J Stephens and Christopher G Atkeson. Dynamic balance force control for compliant humanoid robots. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1248–1255. IEEE, 2010. ISBN 1424466768.

[49] Nikos G Tsagarakis, Stephen Morfey, Gustavo Medrano Cerda, Li Zhibin, and Darwin G Caldwell. Compliant humanoid coman: Optimal joint stiffness tuning for modal frequency control. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 673–678. IEEE, 2013. ISBN 1467356433.

[50] Miomir Vukobratović and Branislav Borovac. Zero-moment point—thirty five years of its life. *International journal of humanoid robotics*, 1(01):157–173, 2004. ISSN 0219-8436.

[51] Alexander W Winkler, Carlos Mastalli, Ioannis Havoutis, Michele Focchi, Darwin G Caldwell, and Claudio Semini. Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5148–5154. IEEE, 2015. ISBN 1479969230.

[52] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985. ISSN 0278-3649.