University of Trento

Master's Degree in Mechatronics engineering

Robust satisfaction of joint position and velocity bounds for robot manipulators

Author:
Erik Zanolli *(198852)*

Supervisor:
Dr. Andrea del Prete

March 17, 2021

Abstract

This thesis deals with the problem of robust control of robotic systems, with a focus on industrial manipulators. One of the main challenges in the control of these systems comes from the fact that they are *highly constrained*. For instance, they typically have bounded joint positions, velocities, accelerations and torques. Velocities and accelerations may be not directly bounded, but their limits may be the consequence of other bounds, such as on the motor current, or the gear-box torque. Moreover, these robotic systems are subject to *disturbances*, which may arise from modeling errors, sensor noises or communication delays.

In order to tackle the above-mentioned challenges, this work presents mathematical and computational tools that account for constraints and disturbances in the control of robot manipulators. In particular, we have faced two different versions of this problem.

In the first version we consider a system subject to *bounded additive disturbances* on the control inputs, with constant joint position, velocity and *acceleration bounds*. For this system, we compute the *robust viability kernel*, which is the set of states such that, starting from any such state, it is possible to avoid violating the constraints in the future, despite the presence of the disturbances. Then we develop an efficient algorithm for computing the range of feasible accelerations that allows the state to remain inside the viability kernel. Tests are performed in simulation with a single joint and a *6-DOF* robot manipulator (i.e. the *Baxter* robot arm), demonstrating the effectiveness of the proposed approach compared to other state-of-the-art methods.

In the second version of the problem we have replaced the joint acceleration bounds with *joint torque bounds*. This leads to a more realistic, but also more challenging problem. Constant torque bounds correspond to state-dependent acceleration bounds, which make the computation of the viability kernel extremely hard. Given the complexity introduced by the joint torque bounds, we have analyzed the non-robust version of the problem. We have started from a well-known method [6] to compute inner approximations of 1-step reachable sets. The method is based on the iterative discretization of the state space in hyper-boxes, followed by the use of interval arithmetic to conservatively verify reachability of a given target set. We have extended and improved this method to approximate the viability kernel and to be able to work with a flat target set (which was not possible in the original version). We have tested the developed algorithm on a *single-joint* system (i.e. a simple pendulum), showing its capability to quickly compute an inner approximation of the viability kernel. Despite its preliminary nature, this work is

a valuable contribution towards the ambitious goal of computing viability kernels for arbitrary high-dimensional nonlinear systems, such as robot manipulators.

Contents

Chapter 1

Introduction

1.1 Overview

One of the main challenges of robotic systems such as industrial manipulators is the fact that they are highly constrained. Robot joints typically have bounded positions. Moreover, actuators have bounded velocity, acceleration and torque/current. This thesis focuses on the control of joints with constant bounds on position, velocity and either acceleration or torque. Even though velocity and acceleration bounds may be not constant, they are often approximated by constant values [30, 37, 17, 23, 34, 11, 35].

An additional challenge is that these systems are subject to disturbances, which may arise from modeling errors, sensor noise or communication delays. Therefore this manuscript deals with the problem of robust control, considering the presence of bounded additive disturbances on the system inputs. The problem of non-robust control of system with constant acceleration bounds has been already investigated in the literature [11][12][35]. In particular, the algorithm of [12] is the starting point of our work. The main contribution of this thesis is to extend this algorithm introducing robustness to bounded additive disturbances on the control inputs. This method is presented in Chapter 2.

Even if the developed method guarantees a robust control in the presence of disturbances, it does not tackle a limitation that affects also previous work. Such robust control is obtained on systems with constant joint acceleration bounds. A real robot has instead state-dependent acceleration bounds, which are often well captured by constant joint torque bounds. Therefore the main focus of Chapter 3 is on guaranteeing the satisfaction of joint position and velocity bounds for a robot manipulator with constant joint torque bounds.

The problem of satisfying the system constraints can be easily tackled if one is able to compute the so-called *viability kernel*, which is the set of all states,

starting from which, the system can avoid violating the constraints. When this set is too hard to compute, one could target a slightly easier problem, which is the computation of a *control invariant* set. This is a set such that, if the state starts inside the set, it is possible to keep it inside the set. Control invariant sets give us sufficient (but not necessary) conditions to guarantee constraint satisfaction. The viability kernel instead, which is the largest control invariant set, gives us a condition that is both sufficient and necessary.

Computing these sets is in general a challenging problem, that has received considerable attention in the literature [16][6]. We model our system (e.g., a robot manipulator) as a nonlinear system subject to simple linear bounds on state and control. Starting from the algorithm of [6] we propose an extension that is able to deal with our problem. The proposed method is tested with a simple system (i.e. a 1-*DOF* pendulum). The manuscript ends with a discussion of the known limitations of the presented techniques and possible future directions for improvement.

1.2 State of the art

This section reviews the relevant state of the art on the subject of viability, with a specific focus on robot manipulators. First, we discuss the problem assuming constant joint acceleration bounds. Then, we discuss the more challenging problem of constant joint torque bounds.

1.2.1 Viability with joint acceleration bounds

First, let us introduce the notation that will be used throughout the thesis:

- δt is the time-step duration of the controller.
- $q(t), \dot{q}(t), \ddot{q}(t) \in \mathbb{R}$ are the joint position, velocity and acceleration at time t.
- q^{min}, q^{max} are the joint position boundaries.
- $\dot{q}^{max}$, $\ddot{q}^{max}$ are the maximum velocity and acceleration.
- τ is the torque applied at the joints bounded in $[\tau^{min}, \tau^{max}]$.
- $M(q), h(q, \dot{q})$ are respectively the mass matrix and the bias forces of a robotic system.

Assuming a constant acceleration throughout each time step, the future position q^+ and velocity $\dot{q}^+$ are functions of the current acceleration $\ddot{q}$:

$$\begin{aligned} q^+ &= q + \delta t\, \dot{q} + \frac{1}{2}\delta t^2 \ddot{q} \\ \dot{q}^+ &= \dot{q} + \delta t\, \ddot{q} \end{aligned} \tag{1.1}$$

A naive approach to bound accelerations is to compute the maximum and minimum $\ddot{q}$ such that q^+ and $\dot{q}^+$ are within their bounds:

$$\begin{aligned} \ddot{q} &\leq \min\left(\quad \ddot{q}^{max}, \frac{1}{\delta t}(\quad \dot{q}^{max} - \dot{q}), \frac{2}{\delta t^2}(q^{max} - q - \delta t\, \dot{q}) \right) \\ \ddot{q} &\geq \max\left(-\ddot{q}^{max}, \frac{1}{\delta t}(-\dot{q}^{max} - \dot{q}), \frac{2}{\delta t^2}(q^{min} - q - \delta t\, \dot{q}) \right) \end{aligned} \tag{1.2}$$

This simple approach is unsatisfactory because the resulting bounds may be incompatible, leading to an unfeasible problem [30]. Several improvements have been proposed, but none of them have completely solved the problem.

A common approach [30, 37] is to use a larger value of δt in (1.2), which results (most of the time) in stricter bounds. This helps reducing the acceleration when approaching a bound, but does not guarantee constraint compatibility.

Another approach [17] is to bound velocity with a hand-tuned linear function of the distance to the position limit. Reducing the velocity as you get close to a position bound is surely a sensible idea, but this method does not explicitly account for acceleration limits. The same method was later used in [39].

Other inverse-dynamics control frameworks [40, 22] simply do not model joint position-velocity bounds, but rather rely on the design of reference trajectories that ensure their satisfaction. This approach works well as long as the actual trajectories stay close to the reference ones, but it cannot guarantee that the controller reaction to a disturbance will not lead to violating a bound.

Control barrier functions provide a general framework for handling constraints with arbitrary relative degree [29, 32]. However, these methods do not deal with constraint conflicts, which are the key issue when considering bounds on position and acceleration [34].

To the best of our knowledge, Decre et al. [11] have been the first ones trying to provide formal guarantees of satisfaction of position, velocity and acceleration bounds in robotics. They computed the future position trajectory assuming maximum deceleration, and imposed the satisfaction of the position bound for all the future states. This strategy allows them to bound the joint velocity with a nonlinear function of the joint position. Their method does not require any hand tuning, but it has two critical issues. First, they assumed constant velocity throughout the time step, so they do not really bound the acceleration, but a pseudo-acceleration, defined as $(\dot{q}^+ - \dot{q})/\delta t$ (the real acceleration is always zero except for the instants where the velocity changes). Second, their method may lead to conflicts between velocity and acceleration limits when getting close to the position bounds. This second issue was later addressed by Rubrecht et al. [34, 35]. However, they introduced some conservatism in the solution, which they suggest to be beneficial in case of measurement errors. We believe that the robustness issue should be tackled explicitly, as we do in this thesis, and not through some arbitrary conservatism.

Moreover, also this work dealt with pseudo-acceleration bounds.

Other relevant results have been proposed in the field of online trajectory generation. Kroeger et al. [23] presented online trajectory-generation algorithms that can guarantee velocity, acceleration and jerk bounds. While these algorithms are fast enough to run at 1 kHz, they treat a different problem from the one treated in this thesis. First, they do not consider position limits. Second, they compute minimum time joint-space trajectories, while we compute the minimum and maximum accelerations that ensure the possibility to satisfy the bounds in the long term. This means that, for instance, motion can be generated according to a task-space criterion, such as following a trajectory with the end-effector, while exploiting the manipulator redundancy to satisfy the robot constraints.

Viability method: robustness

The viability-based approach developed in [12] bounds accelerations with theoretical guarantees in term of constraint violations. However, it is a non-robust method, so it cannot deal with the presence of disturbances. The viability-based approach has some good properties

- it is exact and does not introduce any type of arbitrary conservatism;
- it assumes a constant acceleration between consecutive time step, so it bounds the real acceleration (as opposed to other methods that consider pseudo-accelerations).

Considering these properties and its performances, we aim to develop a robust version of the viability-based approach. As we are going to explain in a rigorous manner in the next chapter, we can prove that ensuring the satisfaction of constraints for an extreme value of disturbance corresponds to ensuring the constraint satisfaction for all the disturbance values in between the bounds. This is because the extreme disturbance values represent the worst-case conditions for the system.

1.2.2 Viability with joint torque bounds

The problem of dealing with constraints can be tackled via computation of a control invariant set. Computation of such sets is well discussed in the literature from both linear [15, 16, 36, 27] and nonlinear [6, 7, 14, 19] system. A robotic system with constant joint torque bounds can be modeled as a linear system with nonlinear constraints, i.e. having the dynamics of the system in the constraints:

$$
\begin{aligned}
q^+ &= q + \delta t\, \dot{q} + \frac{1}{2}\delta t^2 \ddot{q} \\
\dot{q}^+ &= \dot{q} + \delta t\, \ddot{q} \\
&\text{subject to} \\
& q^{min} \leq q \leq q^{max} \\
& \dot{q}^{min} \leq \dot{q} \leq \dot{q}^{max} \\
& \tau^{min} \leq M(q)\ddot{q} + h(q,\dot{q}) \leq \tau^{max}
\end{aligned}
\tag{1.3}
$$

This implies that methods that require linear constraint, such as [15, 27], do not apply to our problem. For linear systems with nonlinear constraints there exists an iterative method [16] able to compute an invariant set for a specific linear controller. However, if the constraints are not convex (which is our case), the method requires solving non-convex optimization problems, which is not possible in general. Since it is not possible to ensure that the dynamics of a generic robot manipulator is a convex function, we cannot rely on this method.

Alternatively, the robotic system can be modeled as a nonlinear system with linear bounds on inputs (joint torques) and state.

$$
\begin{aligned}
q^+ &= q + \delta t\, \dot{q} + \frac{1}{2}\delta t^2 M(q)^{-1}(\tau - (h(q,\dot{q}))) \\
\dot{q}^+ &= \dot{q} + \delta t\, M(q)^{-1}(\tau - (h(q,\dot{q}))) \\
&\text{subject to} \\
& q^{min} \leq q \leq q^{max} \\
& \dot{q}^{min} \leq \dot{q} \leq \dot{q}^{max} \\
& \tau^{min} \leq \tau \leq \tau^{max}
\end{aligned}
\tag{1.4}
$$

The computation of invariant sets for constrained nonlinear systems is an open research problem.

As stated in [25, 18] there is a close relationship between viability theory and *constrained reachability* theory. The viability kernel, as previously defined, is the set of all states, starting from which, the system can avoid violating the constraints. Reachability analysis instead provides methods for simulating all possible trajectories of a dynamic system under all the admissible input range. To perform this reachability analysis it is possible to start considering the set of initial states and follow this set forward in time under the flow of the system dynamics to compute the so-called *forward reachable set*. Alternatively, it is possible to start from the terminal set and follow the flow of the system dynamics backward in time to compute the so-called *backward reachable set*. [25, 18] reformulate the definition of the *finite horizon* viability kernels in terms of the backward reachable set over

one discrete time step, using as the basis for their algorithm [38]. Computing the *infinite reachable* sets allows us to find also the *infinite horizon* viability kernel.

This connection between the viability kernel and the reachable sets applies to any system, such as those with nonlinear dynamics and/or non-convex state constraints, however [25, 18] developed a method for linear systems with convex input and state constraint sets, therefore not directly suitable for our system.

The computation of 1-step reachable and infinite reachable sets for nonlinear system is obtained by mean of a recursion algorithm in [3, 2, 6]. [2, 1, 26] rather than studying the nonlinear system $\dot{x} = f(x)$, it proposes to study an *approximation* $\dot{x} = g(x)$ that is easier to handle. Such $g(x)$ approximation consists in modeling the initial system as a *piecewise linear* system. Moreover, in order to be conservative, it adds a bounded input to take into account the interpolating error. The accuracy can be tuned by choosing a finer mesh on which to perform the linearization. The aim of this linearization is to use the already developed and well-known methods for linear system.

The method of [6] does not require any linearization and develops a branch-and-bound algorithm able to provide in a computationally efficient way an approximation of the so-called *one-step reachable set* for nonlinear system. This algorithm can therefore be used recursively to find an inner (i.e. conservative) approximation of the infinite reachable set. This conservative behavior is obtained tackling the problem with the *intervals arithmetic*. The algorithm formulation cannot be directly applied to a robotic system like ours for reasons discussed later in Section 3.2.2.

However, from this starting point, with the relaxation of certain hypotheses and utilizing some viability-related previous discussed methods, it is possible to build an improved branch-and-bound algorithm able to compute an inner approximation of the viability kernel of the system. We choose this algorithm [6] as the basis for our work because:

- It is generic, namely it does assume anything about the system dynamics.
- It has the *potential* to scale to system with many degrees of freedom.
- Its modified implementation is computationally efficient. Its branch-and-bound structure prune the non-viable zones in a fast way.

Chapter 2

Robust viability with joint acceleration limits

2.1 Notation

Let us introduce the notation that will be used throughout the chapter:

- $\wedge$ and $\vee$ denote the logical quantifiers AND and OR.
- $t \in \mathbb{R}^+$ denotes time.
- $i \in \mathbb{N}$ denotes discrete time steps.
- δt is the time-step duration of the controller.
- w_t is the disturbance applied considering a continuous time system, while w_i is the disturbance applied to the discrete system kept constant for the whole i-th time step.
- $q(t), \dot{q}(t), \ddot{q}(t) \in \mathbb{R}$ are the joint position, velocity and acceleration at time t.
- $q_i \triangleq q(i\delta t), \quad \dot{q}_i \triangleq \dot{q}(i\delta t), \quad \ddot{q}_i \triangleq \ddot{q}(i\delta t)$.
- q^{min}, q^{max} are the joint position boundaries.
- $\dot{q}^{max}$, $\ddot{q}^{max}$ are the maximum velocity and acceleration, implying the assumption of symmetrical velocity and acceleration bounds.

The method can be easily extended and generalized to an asymmetrical acceleration bound, which is often the case in practice, but it would make more verbose the analysis without adding true benefits. For the sake of clarity, we prefer to hold the hypothesis of a symmetrical acceleration bound $\ddot{q} \in [-\ddot{q}^{max}, \ddot{q}^{max}]$ in our examples.

2.2 Problem statement

2.2.1 Feasible states

Considering a robot whose joints have position limits and acceleration limits, it is possible to define, as in [12], the set of feasible states for a single joint as:

$$X = \{(q,\dot{q}) \in \mathbb{R}^2 : q^{min} \leq q \leq q^{max}, |\dot{q}| \leq \dot{q}^{max}\} \tag{2.1}$$

As previously stated, our control inputs are the joint accelerations. The accelerations are bounded: $|\ddot{q}| \leq \ddot{q}^{max}$. It is important to notice that w_i does not influence the acceleration bounds; $\ddot{q}$ is the desired acceleration applied to the joint, to which we add the disturbance. The problem of finding the maximum and minimum allowable accelerations such that the constraint on position and velocity can be satisfied in the future can be formulated as an infinite-horizon optimal control problem

$$\begin{aligned}
\ddot{q}_0^{max} = \underset{\ddot{q}_0,\ddot{q}_1,\dots}{\text{maximize}} \quad & \ddot{q}_0 \\
\text{subject to} \quad & q(i\delta t + t) = q_i + t\,\dot{q}_i + \frac{t^2}{2}(\ddot{q}_i + w_i) && \forall i \geq 0, t \in [0,\delta t], w_i \in [-\bar{w},\bar{w}] \\
& \dot{q}(i\delta t + t) = \dot{q}_i + t\,(\ddot{q}_i + w_i) && \forall i \geq 0, t \in [0,\delta t], w_i \in [-\bar{w},\bar{w}] \\
& (q(t),\dot{q}(t)) \in X && \forall t > 0 \\
& |\ddot{q}_i| \leq \ddot{q}^{max} && \forall i \geq 0 \\
& (q(0),\dot{q}(0)) \quad \text{fixed}
\end{aligned} \tag{2.2}$$

The problem, as we can see in the constraints of (2.2), is affected by w_i because it contains the system dynamics, which is affected by the disturbance. The q and $\dot{q}$ behavior inside the time step is fundamental because the joints limits must be satisfied also in continuous time inside the time step and not only in discrete time. It is important to state that the discrete time laws in (2.2) are special cases of the continuous time laws defined inside the time step with $t = \delta t$. This problem has an infinite number of constraints and cannot be solved. Our approach to the solution is to extend the result obtained by [12] taking into account the disturbances, recompute the viability kernel and rewrite the position and velocity inequalities starting from the new viability kernel.

2.3 Problem solution

The concept of viability will help us reformulate the problem in (2.2). A state is defined as viable if starting from that state there exists a sequence of control inputs

that allows for the satisfaction of all constraints in the future. The robust viability kernel $\mathscr{V}$ is defined as:

$$
\begin{aligned}
(q(0),\dot{q}(0)) \in \mathscr{V} \quad \Leftrightarrow \quad \exists(\ddot{q}_i)_{i=0}^{\infty} : q(i\delta t + t) &= q_i + t\,\dot{q}_i + \frac{t^2}{2}(\ddot{q}_i + w_i) && t \in [0,\delta t], \forall w_i \in [-\bar{w},\bar{w}] \\
\dot{q}(i\delta t + t) &= \dot{q}_i + t\,(\ddot{q}_i + w_i) && t \in [0,\delta t], \forall w_i \in [-\bar{w},\bar{w}] \\
(q(t),\dot{q}(t)) &\in X && \forall t > 0, \forall w_i \in [-\bar{w},\bar{w}] \\
|\ddot{q}_i| &\leq \ddot{q}^{max} && \forall i \geq 0
\end{aligned}
\tag{2.3}
$$

The main interest in the introduction of the robust viability kernel $\mathscr{V}$ is that ensuring the existence of a feasible future trajectory (i.e. our original problem) is clearly equivalent to ensuring that the next state belongs to $\mathscr{V}$. However, this definition of $\mathscr{V}$ does not immediately provide practical utility: verifying its membership amounts to finding an infinite sequence of accelerations that results in a feasible trajectory, which is too computationally demanding. In the following, we derive an equivalent definition of $\mathscr{V}$ that allows us to check membership easily. Thanks to this, we reformulate the hard problem of satisfying the position-velocity-acceleration limits as the simple problem of ensuring that the next state is viable.

2.3.1 Continuous time control

In the beginning, let us assume to deal with a continuous control in which we can change $\ddot{q}$ at any instant, and consequently w_t can changes its value at any instant. This results in a set of viable states $\mathscr{V}^C$ that is a superset of the previous one, so $\mathscr{V} \subset \mathscr{V}^C$. It is obvious that a viable state is also feasible, but not all the feasible states are viable. For instance, if the system is approaching a bound with a large velocity even the maximum deceleration may not be enough to stop it and prevent a violation of the position constraint. For a given initial position q_0, we can find the maximum initial velocity $\dot{q}_M^{\mathscr{V}}$ that allows us to satisfy the position limits in the future:

$$
\begin{aligned}
\dot{q}_M^{\mathscr{V}} = \underset{\dot{q}_0,\ddot{q}(t)}{\text{maximize}} \quad & \dot{q}_0 \\
\text{subject to} \quad & q(t) = q_0 + t\,\dot{q}(t) + \frac{t^2}{2}(\ddot{q}(t) + w_t) && \forall t \geq 0, \forall w_t \in [-\bar{w},\bar{w}] \\
& \dot{q}(t) = \dot{q}_0 + t\,(\ddot{q}(t) + w_t) && \forall t \geq 0, \forall w_t \in [-\bar{w},\bar{w}] \\
& (q(t),\dot{q}(t)) \in X && \forall t \geq 0, \forall w_t \in [-\bar{w},\bar{w}] \\
& |\ddot{q}(t)| \leq \ddot{q}^{max} && \forall t \geq 0 \\
& q(0) = q_0, \quad \dot{q}(0) = \dot{q}_0
\end{aligned}
\tag{2.4}
$$

In the case without disturbance the solution of this problem is rather intuitive: the maximum initial velocity is such that, if we constantly apply the maximum deceleration, we end up exactly at q^{max} with zero velocity. Taking into account the disturbance w_t, that is an unknown value in the $[-\bar{w}, \bar{w}]$ interval, we can consider applying the maximum deceleration $-\ddot{q}^{max}$ and the maximum value of our disturbance $\bar{w}$: this condition represents the worst-case maximum deceleration that the system can exert, i.e. the maximum deceleration that the motor can produce is reduced by the worst-case disturbance. Verifying it with the bound value of our w_t range means verifying the condition for every $w_t \in [-\bar{w}, \bar{w}]$. So we

1. write the position trajectory for the maximum deceleration and maximum disturbance $\ddot{q}(t) = -\ddot{q}^{max} + \bar{w}$,
2. compute the time at which the velocity of this trajectory is zero $t^0 = \dot{q}_0/(\ddot{q}^{max} - \bar{w})$,
3. compute the initial velocity such that $q(t^0) = q^{max}$

Following these steps we find:

$$\dot{q}_M^{\mathscr{V}}(q) = \sqrt{2(\ddot{q}^{max} - \bar{w})(q^{max} - q_0)} \tag{2.5}$$

As we can see in (2.5) the disturbance w_t decreases the value of $\dot{q}_M^{\mathscr{V}}(q)$. Following the same steps, we can define also the minimum velocity to ensure viability

$$\dot{q}_m^{\mathscr{V}}(q) = -\sqrt{2(\ddot{q}^{max} - \bar{w})(q_0 - q^{min})} \tag{2.6}$$

So the set $\mathscr{V}^C$ of the viable states can be rewritten as

$$\mathscr{V}^c = \{(q, \dot{q}) : (q, \dot{q}) \in X, \dot{q}_m^{\mathscr{V}}(q) \leq \dot{q} \leq \dot{q}_M^{\mathscr{V}}(q)\} \tag{2.7}$$

This definition of $\mathscr{V}^c$ allows us to check easily the viability of a state by just verifying three inequalities.

In Fig. 2.1 it is possible to notice how the state-space varies with the presence of disturbances. As expected, the forbidden values are in the corners of the state-space plot so where the value of the $(q_i, \dot{q}_i)$ are both near the respective bounds. Fig. 2.1 also shows the difference between the state-space area generated by a no-disturbance case and with a disturbance of $w_t = \frac{1}{3}\ddot{q}^{max}$.

2.3.2 Discrete-time control

The main difference between the continuous-time control and discrete-time is the fact that in continuous case we can change the value of the acceleration at any instant and also the disturbance can change its value at any instant, while in discrete

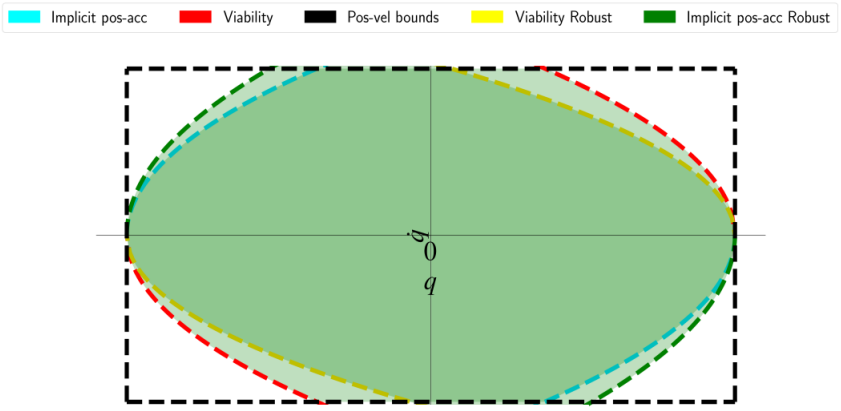


Figure 2.1: State-space with $w_t = 0$ and with $w_t = \bar{w}$. The bounds on position and velocity are $q_{min} = -0.5, q_{max} = 0.5, |\dot{q}_{max}| = 10$ and $\ddot{q}^{max} = 5$ for acceleration.

time control we must keep a constant value for both acceleration and disturbance for the whole time step. Neglecting for a moment w_t, this means that if we reach the q^{max} position with zero velocity with continuous control, we can switch to zero also the $\ddot{q}$ value. Instead, if reach the q^{max} position with zero velocity at the end of a time step with discrete control, we must apply a constant deceleration for the whole time step. Theoretically, this can lead to a violation of the lower position-velocity bounds. Let us understand in which condition this is possible. The worst case is represented by the system reaching the state $(q^{max}, 0)$ a moment after the beginning of the time step with the maximum deceleration $\ddot{q} = -\ddot{q}^{max}$ and the disturbance $w_i = -\bar{w}$ that increases the deceleration. The resulting deceleration is kept for the whole time step, resulting in:

$$\begin{cases} q_1 = q^{max} - 0.5\delta t^2(\ddot{q}^{max} + \bar{w}) \\ \dot{q}_1 = -\delta t(\ddot{q}^{max} + \bar{w}) \end{cases} \tag{2.8}$$

First of all, it is necessary that the value of $\dot{q}_1$ does not violate the velocity bound, so:

$$\dot{q}^{max} \leq -\delta t(\ddot{q}^{max} + \bar{w}) \tag{2.9}$$

We can see this bound also as a bound on $\bar{w}$

$$\bar{w} \leq \frac{\dot{q}^{max} - \delta t \ddot{q}^{max}}{\delta t} \tag{2.10}$$

After the first time step the joint is approaching the lower position bound with velocity $\dot{q}_1$. To ensure that the lower bound is not going to be reached, it is necessary

to obtain an inversion of the sign of $\dot{q}$. The best thing that the controller can do to stop the joint is to apply maximum acceleration, while the worst-case error that can occur is $-\bar{w}$, which decreases the acceleration.

$$\begin{cases} q_2 = q_1 + \dot{q}_1 \delta t + 0.5\delta t^2 (\ddot{q}^{max} - \bar{w}) \\ \dot{q}_2 = \dot{q}_1 + \delta t (\ddot{q}^{max} - \bar{w}) \end{cases} \tag{2.11}$$

After the substitution of $q_1, \dot{q}_1$:

$$\begin{cases} q_2 = q^{max} - 0.5\delta t^2 (\ddot{q}^{max} + 2\bar{w}) \\ \dot{q}_2 = -2\delta t \bar{w} \end{cases} \tag{2.12}$$

Due to the nonzero disturbances $-\bar{w}$, the velocity is still negative. So we apply again maximum acceleration, with disturbance $w_i = -\bar{w}$, and analyze in first place the resulting velocity:

$$\begin{cases} \dot{q}_3 = -2\delta t \bar{w} + \delta t (\ddot{q}^{max} - \bar{w}) \\ \dot{q}_3 = \delta t (\ddot{q}^{max} - 3\bar{w}) \end{cases} \tag{2.13}$$

After 3 time steps the joint velocity could be positive, depending on the value of $\bar{w}$. In the following we are going to assume that this is the case. This is not necessary, but it is a reasonable assumption because it results in a rather large upper bound for $\bar{w}$:

$$\delta t (\ddot{q}^{max} - 3\bar{w}) \geq 0 \quad \Rightarrow \quad \bar{w} \leq \frac{1}{3} \ddot{q}^{max} \tag{2.14}$$

The assumption of a positive $\dot{q}$ after 3 time steps leads to accept as valid only disturbances that are not greater than one third of the maximum acceleration. In theory it would be possible to deal with greater values of $\bar{w}$ considering a switch of velocity sign after a higher number of time steps. Thanks to this assumption, we know that in the third time step the sign of $\dot{q}$ is going to change. Analysing the $\dot{q}$ trajectory in continuous time in the third time step we can calculate the time when the velocity is null, i.e. the time when the position reaches its minimum value.

$$\begin{gathered} \dot{q}_3(t) = \dot{q}_2 + t_0 (\ddot{q}^{max} - \bar{w}) = 0 \\ t_0 = \frac{-\dot{q}_2}{\ddot{q}^{max} - \bar{w}} \Rightarrow t_0 = \frac{2\delta t \bar{w}}{\ddot{q}^{max} - \bar{w}} \end{gathered} \tag{2.15}$$

We can now substitute t_0 in the position equation and obtain the minimum value of the trajectory. This value is defined as q^{discr} and represents the maximum value of the lower bound such that we can neglect the fact that the controller operates in discrete time.

$$q_3(t_0) \triangleq q_{discr} = q_2 - \frac{4\delta t \bar{w}}{\ddot{q}^{max} - \bar{w}} \delta t \bar{w} + \frac{\delta t^2 \bar{w}^2}{\ddot{q}^{max} - \bar{w}} (\ddot{q}^{max} - \bar{w}) \tag{2.16}$$

The lower position bound q^{min} must be less than or equal to q^{discr}:

$$q_{discr} \triangleq q^{max} - \delta t^2 (\ddot{q}^{max} + 2\bar{w} + \frac{\bar{w}^2}{\ddot{q}^{max} - \bar{w}}) \geq q^{min} \tag{2.17}$$

$$q^{max} - q^{min} \leq \delta t^2 (\ddot{q}^{max} + 2\bar{w} + \frac{\bar{w}^2}{\ddot{q}^{max} - \bar{w}}) \tag{2.18}$$

Looking at (2.18) it is possible to notice that when the disturbance $\bar{w} = 0$, the expression is exactly the one found in [12], as expected. At a glance we can notice that the effective acceleration of the system $(\ddot{q}^{max} + 2\bar{w} + \frac{\bar{w}^2}{\ddot{q}^{max} - \bar{w}})$ contains both $\ddot{q}^{max}$ and $\bar{w}$ and must be greater or equal to the $(q^{max} - q^{min})/\delta t^2$, but there is no substantial difference in terms of effective acceleration in how much $\ddot{q}^{max}$ and $\bar{w}$ influence the final value, i.e. we can both respect the condition (2.18) for example with a large value of $\ddot{q}^{max}$ and a small value for $\bar{w}$ or a small value of $\ddot{q}^{max}$ and a large value of $\bar{w}$. Now we want to find the maximum value of w_t that satisfies that expression. It is possible to rewrite (2.18) in the form:

$$a\bar{w}^2 + b\bar{w} + c \geq 0, \tag{2.19}$$

where:

- $a = \delta t^2$
- $b = q^{min} - q^{max} - \delta t^2 \ddot{q}^{max}$
- $c = (q^{max} - q^{min} - \delta t^2 \ddot{q}^{max}) \ddot{q}^{max}$

It is reasonable to express the maximum disturbance as a percentage of $\ddot{q}^{max}$:

$$\bar{w} = \alpha \ddot{q}^{max}, \tag{2.20}$$

where $\alpha \in \mathbb{R}^+$ is a positive scalar. Substituting it in (2.19):

$$\delta t^2 \ddot{q}^{max2} \alpha^2 + (q^{min} - q^{max} - \delta t^2 \ddot{q}^{max}) \ddot{q}^{max} \alpha - \delta t^2 \ddot{q}^{max2} + \ddot{q}^{max} q^{max} - q^{min} \ddot{q}^{max} \geq 0 \tag{2.21}$$

Eq. (2.21) is satisfied for the set of $\alpha : \alpha_1 \geq \alpha \vee \alpha \geq \alpha_2$, where the two extreme values α_1 and α_2 are computed as:

$$\alpha_{1,2} = \frac{k^2 + \ddot{q}_{max} q^r \pm \sqrt{(k^2 + \ddot{q}^{max} q^r)^2 - 4k^2 (\ddot{q}^{max} q^r - k^2))}}{2k^2} \tag{2.22}$$

where we have defined:

$$\begin{cases} k \triangleq \ddot{q}^{max}\delta t \\ q^r \triangleq q^{max} - q^{min} \end{cases} \tag{2.23}$$

However, we can simplify the expression of the range because for our hypothesis in (2.14) the disturbance must be $\bar{w} \leq \frac{1}{3}\ddot{q}^{max}$ and so $\alpha \leq \frac{1}{3}$. At the same time, negative alpha values are not admissible. Since the bound of α imposed by (2.14) can be more strict than both $\alpha_{1,2}$, we must find a general form to compute the allowable range. First of all we can notice that

$$\alpha_2 = \frac{k^2 + \ddot{q}_{max}q^r + \sqrt{(k^2 + \ddot{q}^{max}q^r)^2 - 4k^2(\ddot{q}^{max}q^r - k^2))}}{2k^2} \tag{2.24}$$

has in the numerator k^2 plus a positive value (the square root plus a positive value $\ddot{q}^{max}q^r$), and in the denominator $2k^2$:

$$\alpha_2 = \frac{k^2+n}{2k^2}, \quad n \in \mathbb{R}^+ \Leftrightarrow \quad \alpha_2 \geq \frac{1}{2} \tag{2.25}$$

As shown in (2.25),$\alpha_2 \geq \frac{1}{2}$, so greater than the maximum α allowable in our analysis due to (2.14), which is $\alpha = \frac{1}{3}$. This lead us to discard always the range $\alpha \geq \alpha_2$, and keep only the range

$$\alpha \in \left[0, \min\left(\alpha_1, \frac{1}{3}\right)\right] \tag{2.26}$$

The results obtained so far derive from the position analysis and the definition of q^{discr}. Another bound on the maximum allowable α value comes rewriting the velocity constraint (2.10)

$$\alpha \leq \frac{\dot{q}^{max} - \delta t\ddot{q}^{max}}{\delta t\ddot{q}^{max}} \triangleq \alpha_3 \tag{2.27}$$

We have now considered all the contributions to define our final range for the allowable α:

$$\alpha \in \left[0, \min\left(\alpha_1, \frac{1}{3}, \alpha_3\right)\right] \tag{2.28}$$

If α belongs to this range, then in our analysis we can neglect the fact that the controller can only change the acceleration at discrete instants. This greatly simplifies the problem, and we expect this assumption to be verified in most practical cases. Therefore, In the following we will assume to be in the condition where the discrete-time and the continuous-time robust viability kernels are equivalent.

2.3.3 Numerical example

It is possible to better understand the meaning of (2.22) and the previously stated limit on the disturbance set in (2.14) showing a numerical example. We consider a joint whose limits and δt are

$$q^{min} = -0.1rad \quad q^{max} = 0.1rad \quad \delta t = 0.1s \quad \ddot{q}^{max} = 10\frac{rad}{s^2} \tag{2.29}$$

We are initially omitting on purpose the bound on maximum velocity α_3 ,assuming that it is satisfied, to focus on the α_1, α_2 computation. Looking at (2.18) it is possible to set the maximum effective acceleration that the system can take into account as

$$\ddot{q}^{allow} = \frac{q^{max} - q^{min}}{\delta t^2} \Rightarrow \ddot{q}^{allow} = 20 \tag{2.30}$$

It is important to notice that $\ddot{q}^{allow}$ is the sum of an acceleration given by the motor and an acceleration driven by disturbance w_i, but we do not care about the internal allocation of this resultant acceleration, we simply define the sum of all contributions as $\ddot{q}^{allow}$. It is also important to notice that this is not the maximum acceleration that the joint can provide, but the maximum acceleration that the joint can reach without violations of bounds. From (2.22) it is known that the admissible range of alpha values are $\alpha_1 \geq \alpha \vee \alpha \geq \alpha_2$. $\alpha_{1,2}$ value are reported for our numerical case in (2.31):

$$\begin{cases} \alpha_1 = 0.38 \\ \alpha_2 = 2.62 \end{cases} \tag{2.31}$$

In addition it is known from (2.14) that our final allowable range must be of the form of (2.26), so knowing that we can reject a priori the range greater than α_2, and end up with an allowable α range of $\alpha \in [0, \frac{1}{3}]$, since also α_1 is greater than $\frac{1}{3}$. We can now remove the hypothesis of $\ddot{q}^{max} = 10$ from the initial data of our particular case and express the upper bound α_1 as a function of $\ddot{q}^{max}$. This allows us to represent the allowable disturbance such that the viability kernel remains the same in continuous and discrete time as a function of the maximum acceleration that joint can generate $\ddot{q}^{max}$. A graphical representation is shown in Fig. 2.2. As previously stated, the maximum allowable $\ddot{q}^{max}$ corresponds to $\ddot{q}^{allow}$. Such condition implies that no disturbance is tolerated by our joint and so the value of α_1 is zero. In the neighborhood of $\ddot{q}^{allow}$ the range limit $\alpha \in [0, min(\alpha_1, \frac{1}{3})]$ is constrained by the value of α_1. With lower values of $\ddot{q}^{max}$ instead it is constrained by (2.14).

We can set the bound on our velocity $\dot{q}^{max} = 1\frac{rad}{s}$. Definitions of α_1, α_2 do not change. As it is shown in Fig. 2.3 the velocity bound become more strict than the position bound.

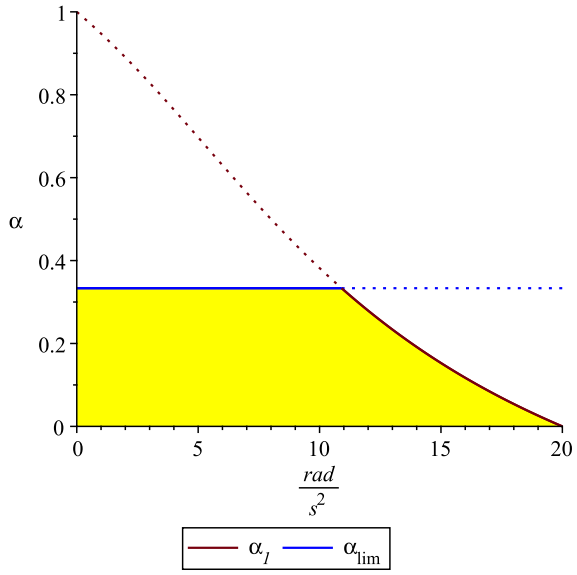


Figure 2.2: Maximum allowable α value varying $\ddot{q}^{max}$ value.

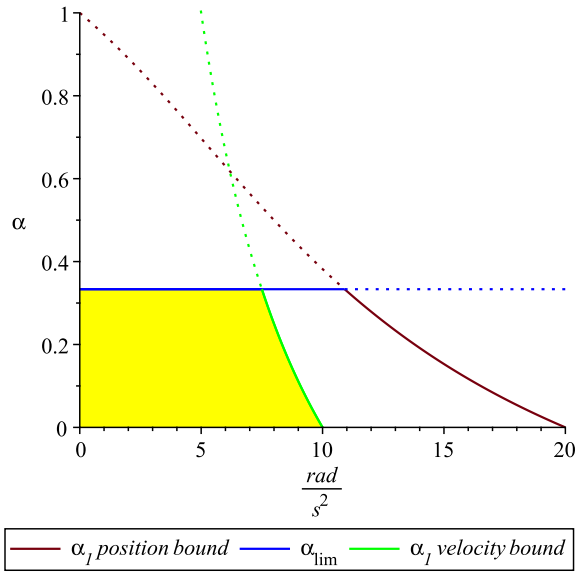


Figure 2.3: Maximum allowable α value with velocity bound over $\ddot{q}^{max}$ value.

As we can notice from this numerical example, the classic working conditions of a joint are quite far from the constraints derived here. Even assuming a very large $\delta t = 0.1$, it requires either $\dot{q}^{max}$ to be more than 10 times smaller than $\ddot{q}^{allow}$, or $(q^{max} - q^{min})$ to be at least 100 times smaller than $\ddot{q}^{allow}$. Since these conditions are of limited practical utility, in the following we assume that (2.21) is verified, and so the viability kernels in discrete time and continuous time coincide: $\mathscr{V} = \mathscr{V}^C$.

2.3.4 Reformulation in terms of viability

We have now obtained a formulation of $\mathscr{V}$ so we can reformulate the problem (2.3). Starting from the current state $(q(0), \dot{q}(0)) \in \mathscr{V}$ we need to compute the maximum value of $\ddot{q}$ such that:

1. the next state $(q(\delta t), \dot{q}(\delta t)) \in \mathscr{V}$.
2. the entire trajectory leading to the next state belongs to the feasible set X.

The first condition alone is not sufficient, similarly to what happens in (2.15), because the trajectory between two viable states may violate a constraint. Viability only ensures the existence of future feasible trajectory, so starting from a viable state we can violate a constraint. We can reformulate (2.3) as:

$$
\begin{aligned}
\ddot{q}_0^{max} = \underset{\ddot{q}}{\text{maximize}} \quad & \ddot{q} \\
\text{subject to} \quad & q(0) = q, \quad \dot{q}(0) = \dot{q} \\
& q(\delta t) = q + t\,\dot{q} + \frac{t^2}{2}(\ddot{q} + w_i) \qquad t \in [0, \delta t], \forall w_i \in [-\bar{w}, \bar{w}] \\
& \dot{q}(\delta t) = \dot{q} + t\,(\ddot{q} + w_i) \qquad t \in [0, \delta t], \forall w_i \in [-\bar{w}, \bar{w}] \\
& (q(t), \dot{q}(t)) \in \mathscr{F} \qquad \forall t \in [0, \delta t], \qquad \forall w_i \in [-\bar{w}, \bar{w}] \\
& \dot{q}_m^{\mathscr{V}}(q(\delta t)) \leq \dot{q}(\delta t) \leq \dot{q}_M^{\mathscr{V}}(q(\delta t)) \\
& |\ddot{q}| \leq \ddot{q}^{max}
\end{aligned}
\tag{2.32}
$$

We can notice that the problem is much simpler than the previous one: it has a single variable instead of a infinite sequence and its constraints concern only the trajectory in the $[0, \delta t]$ interval, rather then in $[0, \infty]$. However, problem (2.32) is still hard to solve because the constraints are infinitely many and nonlinear. We have already stated in (2.5) that the worst-case disturbance corresponds to an extreme value. If we guarantee the satisfaction of the constraints for the extreme value of w_i, we verify all the w_i in the range $[-\bar{w}, \bar{w}]$. For this reason, in the following we replace w_i with its bound values $\pm\bar{w}$.

2.3.5 Position inequalities

The position trajectory with disturbance is

$$q^{min} \leq q + t\dot{q} + \frac{1}{2}t^2(\ddot{q} \pm \bar{w}) \leq q^{max} \; \forall t \in [0, \delta t] \tag{2.33}$$

We use $\pm\bar{w}$ in (2.33) to represent both the critical cases at once. To be more clear and readable from now on in this section we suppose a positive $\bar{w}$ disturbance. The method explained hereafter can be easily applied also with a negative disturbance $-\bar{w}$. We want (2.33) to be satisfied for the whole time step and not only at the end with $t = \delta t$. If the velocity $\dot{q}$ does not change its sign during the time step it is actually enough to verify the condition only in $t = \delta t$. However, if the sign changes in the middle of the time step, the position inequality can be still satisfied in $t = 0$ and $t = \delta t$, but violated somewhere in the middle. Let us focus first on the upper bound. The choice to work with the upper bound is not accidental because it is already known from the previous analysis that this represents the worst-case situation for the violation of the upper-bound limit, i.e. a disturbance that pushes the system towards the upper bound. In the same manner a disturbance $-\bar{w}$ represents the worst case for the lower bound. We can rewrite the upper bound constraint in (2.33) as:

$$f(\ddot{q}) \leq q^{max} \tag{2.34}$$

where:

$$f(\ddot{q}) = \underset{t \in [0, \delta t]}{\text{maximize}} \, [q + t\dot{q} + 0.5t^2(\ddot{q} + \bar{w})] \tag{2.35}$$

Following a procedure similar to (2.15) we want to find the time at which the position reaches its maximum, so we set the time derivative of the position to zero and retrieve the corresponding value:

$$\dot{q} + t(\ddot{q} + \bar{w}) = 0 \rightarrow t^{ext} \triangleq -\dot{q}/(\ddot{q} + \bar{w}) \tag{2.36}$$

The extremum is a maximum and it is reached during the time step only if:

$$\dot{q} \geq 0, \; \ddot{q} \leq -\frac{\dot{q}}{\delta t} - w \triangleq \ddot{q}_1^M \tag{2.37}$$

If these conditions are satisfied the maximum position is:

$$\begin{aligned} f(\ddot{q}) &= q + t^{ext}\dot{q} + \frac{1}{2}(t^{ext})^2(\ddot{q} + \bar{w}) \\ f(\ddot{q}) &= q - \frac{\dot{q}^2}{2(\ddot{q} + \bar{w})} \end{aligned} \tag{2.38}$$

Substituting this expression in (2.34) we get:

$$\begin{aligned} \ddot{q} &\leq \frac{\dot{q}^2 - 2q\bar{w} + 2q^{max}\bar{w}}{2(q - q^{max})} \\ \ddot{q} &\leq \frac{\dot{q}^2}{2(q - q^{max})} - \bar{w} \triangleq \ddot{q}_2^M \end{aligned} \tag{2.39}$$

It is possible to notice that this bound is equivalent to the bound without disturbance found in [12], setting $\bar{w}$ to zero. If the conditions in (2.37) are not satisfied the maximum point of position trajectory is reached at the boundary of the time step and we must only ensure that the constraint is verified for $t = \delta t$:

$$\ddot{q} \leq \frac{2}{\delta t^2}(q^{max} - q - \delta t\dot{q}) - \bar{w} \triangleq \ddot{q}_3^M \tag{2.40}$$

As we can see from (2.39) and (2.40) for the position inequalities it is possible to rewrite the expressions isolating $\bar{w}$. The expressions are the same ones obtained with no disturbance, and adding the contribute of $\bar{w}$ as a last step. Practically this allows us to use the algorithm develop in [12] (Alg. 1) dealing with position inequalities without disturbances, and to add $\bar{w}$ at the end.

We can now put all of this together. If $\dot{q} \leq 0$ the acceleration upper bound is simply $\ddot{q}_3^M$. If $\dot{q} > 0$ things are more complicated instead: if (2.37) is satisfied then the upper bound is $\ddot{q}_2^M$, otherwise the upper bound is $\ddot{q}_3^M$. We may then rewrite (2.34) as:

$$(\ddot{q} \leq \ddot{q}_1^M \wedge \ddot{q} \leq \ddot{q}_2^M) \quad \vee \quad (\ddot{q} > \ddot{q}_1^M \wedge \ddot{q} \leq \ddot{q}_3^M), \tag{2.41}$$

or equivalently:

$$\ddot{q} \leq \min(\ddot{q}_1^M, \ddot{q}_2^M) \quad \vee \quad \ddot{q}_1^M < \ddot{q} \leq \ddot{q}_3^M \tag{2.42}$$

Depending on the values of $\ddot{q}_1^M, \ddot{q}_2^M$ and $\ddot{q}_3^M$, one among them is the *real* acceleration upper bound. We can repeat the analysis with the lower bound and the disturbance equal to $w = -\bar{w}$. The computation is summarized in Alg. 3.

2.3.6 Velocity inequalities

The velocity trajectory is a line and so we need to consider only the extremes of this trajectory, so verify that the bounds are satisfied for $t = \delta t$. As for the position inequalities case we can write

$$\dot{q}^{max} \geq |\dot{q} + \delta t(\ddot{q} \pm \bar{w})| \tag{2.43}$$

to represent both the $[-\bar{w},\bar{w}]$ cases. Since the $\bar{w}$ case is the worst case for the velocity upper bound, rearranging (2.43) we obtain for the upper bound:

$$\begin{aligned} \frac{\dot{q}^{max}-\dot{q}}{\delta t} &\geq (\ddot{q}+\bar{w}) \\ \frac{\dot{q}^{max}-\dot{q}}{\delta t} - \bar{w} &\geq \ddot{q} \end{aligned} \tag{2.44}$$

As for the position inequalities, adding a disturbance $\bar{w}$ for computing the acceleration limit for the upper bound corresponds to simply subtracting $-\bar{w}$ to the acceleration limit computed without disturbance. This analysis stands also for the lower bound, in that case it is sufficient to add $\bar{w}$ to the lower limit calculated without disturbance. In the end we can write the bounds as:

$$\frac{1}{\delta t}(-\dot{q}^{max}-\dot{q})+\bar{w} \leq \ddot{q} \leq \frac{1}{\delta t}(\dot{q}^{max}-\dot{q})-\bar{w} \tag{2.45}$$

2.3.7 Viability inequalities

Let us consider the upper bound of the viability inequality.

$$\dot{q}(\delta t) \leq \sqrt{2(\ddot{q}^{max}-\bar{w})(q^{max}-q(\delta t))} \tag{2.46}$$

As already stated, we are going to deal with the worst-case disturbance, which for the upper bound is $\bar{w}$:

$$\dot{q}+\delta t(\ddot{q}+\bar{w}) \leq \sqrt{2(\ddot{q}^{max}-\bar{w})(q^{max}-q-\delta t\dot{q}-0.5\delta t^2(\ddot{q}+\bar{w}))} \tag{2.47}$$

the constraint is clearly nonlinear, but it is possible to derive an algorithm to reformulate it as a simple upper bound. Since the right-hand-side (RHS) of (2.46) is a square root, it is always positive, so if the left-hand-side (LHS) is negative (2.46) is always satisfied:

$$\dot{q}+\delta t(\ddot{q}+\bar{w}) \leq 0 \;\Leftrightarrow\; \ddot{q} \leq -\frac{\dot{q}}{\delta t}-\bar{w} \triangleq \ddot{q}^1 \tag{2.48}$$

Instead if the LHS is positive we can take the square of both sides:

$$(\dot{q}+\delta t(\ddot{q}+\bar{w}))^2 \leq 2(\ddot{q}^{max}-\bar{w})(q^{max}-q-\delta t\dot{q}-\frac{1}{2}\delta t^2(\ddot{q}+\bar{w})) \tag{2.49}$$

that can be reformulated as $a\ddot{q}^2+b\ddot{q}+c \leq 0$ where:

$$\begin{cases} a = \delta t^2 \\ b = 2\delta t(\dot{q}+\delta t\bar{w})+\delta t^2(\ddot{q}^{max}-\bar{w}) \\ c = (\dot{q}+\delta t\bar{w})^2-2(\ddot{q}^{max}+\bar{w})(-\frac{1}{2}\delta t^2\bar{w}-\delta t\dot{q}-q+q^{max}) \end{cases} \tag{2.50}$$

If $\Delta = b^2 - 4ac \geq 0$ this parabola is equal to zero in two points $\ddot{q}_2$ and $\ddot{q}_3$ (which coincide if $\Delta = 0$):

$$\ddot{q}_2 \triangleq \frac{-b - \sqrt{\Delta}}{2a}, \qquad \ddot{q}_3 \triangleq \frac{-b + \sqrt{\Delta}}{2a} \tag{2.51}$$

Since $a > 0$ by definition, the inequality (2.49) is satisfied for $\ddot{q}_2 \leq \ddot{q} \leq \ddot{q}_3$. If instead $\Delta < 0$ then the parabola is always positive and there exists no value of $\ddot{q}$ that satisfies (2.49). Putting it all together we can rewrite (2.46) as:

$$(\ddot{q} \leq \ddot{q}_1) \quad \vee \quad [(\ddot{q} > \ddot{q}_1) \wedge (\Delta \geq 0) \wedge (\ddot{q}_2 \leq \ddot{q} \leq \ddot{q}_3)] \tag{2.52}$$

Notice that instead of using the formulation (2.46) and the coefficient a, b, c defined in (2.50), we could calculate the acceleration upper bound for the case without $\bar{w}$, and then add $-\bar{w}$ to the resulting value.

$$\dot{q} + \delta t \ddot{q} \leq \sqrt{2(\ddot{q}^{max} - \bar{w})(q^{max} - q - \delta t \dot{q} - 0.5\delta t^2 \ddot{q})} \tag{2.53}$$

This is also possible for the lower bound. We have to notice that the $\bar{w}$ disturbance that comes from the velocity inequality still remains. We can now derive again the a, b, c coefficient, find the roots and add $\pm\bar{w}$. All the aforementioned considerations on $\ddot{q}_1, \ddot{q}_2, \ddot{q}_3$ still remain valid. The a, b, c coefficients used in Alg. 2 derive from this case. Unlike the position inequalities and velocity inequalities, here it is necessary to rewrite the algorithm.

At this point we need to take into account in a unique algorithm all these conditions. As already mentioned, for position and velocity bounds we can simply add $\pm\bar{w}$ to the bounds computed by the algorithms presented in [12], as shown in Algorithm 3. The viability inequality instead needs a modified algorithm.

Fig. 2.4 and Fig. 2.5 show the state space divided in regions based on which acceleration upper bound dominates the others.

2.4 Tests

In this section we are going to analyze the behaviour of robotic systems with the aforementioned implemented robust algorithm, taking into account $w_i \in [-\bar{w}, \bar{w}]$ disturbances. First, we are going to see the behaviour of a simple joint and then the behaviour of the joints of a more complex system, the Baxter robot. A robot manipulator such as the Baxter robot has different position, velocity and acceleration bounds at each joint. Recalling the definition in (2.28), adding a generic disturbance w corresponds to adding a disturbance $w = \alpha \ddot{q}_j^{max}$, where the α value across the various joints re-scales the maximum acceleration $\ddot{q}_j^{max}$ that each joint

Algorithm 1 accBoundsFromPosLimits

Require: $q, \dot{q}, q^{min}, q^{max}, \delta t$
$\ddot{q}_1^M \leftarrow -\dot{q}/\delta t$
$\ddot{q}_2^M \leftarrow -\dot{q}^2/(2(q^{max}-q))$
$\ddot{q}_3^M \leftarrow 2(q^{max}-q-\delta t\,\dot{q})/(\delta t^2)$
$\ddot{q}_2^m \leftarrow \dot{q}^2/(2(q-q^{min}))$
$\ddot{q}_3^m \leftarrow 2(q^{min}-q-\delta t\,\dot{q})/(\delta t^2)$
if $\dot{q} \geq 0$ **then**
  $\ddot{q}^{LB} \leftarrow \ddot{q}_3^m$
  if $\ddot{q}_3^M > \ddot{q}_1^M$ **then**
    $\ddot{q}^{UB} \leftarrow \ddot{q}_3^M$
  else
    $\ddot{q}^{UB} \leftarrow \min(\ddot{q}_1^M, \ddot{q}_2^M)$
else
  $\ddot{q}^{UB} \leftarrow \ddot{q}_3^M$
  if $\ddot{q}_3^m < \ddot{q}_1^M$ **then**
    $\ddot{q}^{LB} \leftarrow \ddot{q}_3^m$
  else
    $\ddot{q}^{LB} \leftarrow \max(\ddot{q}_1^M, \ddot{q}_2^m)$
return $\{\ \ddot{q}^{LB}, \ddot{q}^{UB}\ \}$

Algorithm 2 accBoundsFromViability

Require: $q, \dot{q}, q^{min}, q^{max}, \ddot{q}^{max}, \delta t, \bar{w}$
$a \leftarrow \delta t^2$
$b \leftarrow \delta t(2\dot{q} + (\ddot{q}^{max} - \bar{w})\delta t)$
$c \leftarrow \dot{q}^2 - 2(\ddot{q}^{max} - \bar{w})(q^{max} - q - \delta t\,\dot{q})$
$\ddot{q}_1 \leftarrow -\dot{q}/\delta t$
$\Delta \leftarrow b^2 - 4ac$
if $\Delta \geq 0$ **then**
  $\ddot{q}^{UB} \leftarrow \max(\ddot{q}_1, (-b + \sqrt{\Delta})/(2a))$
else
  $\ddot{q}^{UB} \leftarrow \ddot{q}_1$
$b \leftarrow 2\delta t\,\dot{q} - (\ddot{q}^{max} - \bar{w})\delta t^2$
$c \leftarrow \dot{q}^2 - 2(\ddot{q}^{max} - \bar{w})(q + \delta t\,\dot{q} - q^{min})$
$\Delta \leftarrow b^2 - 4ac$
if $\Delta \geq 0$ **then**
  $\ddot{q}^{LB} \leftarrow \min(\ddot{q}_1, (-b - \sqrt{\Delta})/(2a))$
else
  $\ddot{q}^{LB} \leftarrow \ddot{q}_1$
$\{\ \ddot{q}^{LB}, \ddot{q}^{UB}\ \} \leftarrow \{\ddot{q}^{LB}, \ddot{q}^{UB}\} + (\bar{w}, -\bar{w})$
return $\{\ \ddot{q}^{LB}, \ddot{q}^{UB}\ \}$

Algorithm 3 Compute Joint Acceleration Bounds

Require: $q, \dot{q}, q^{min}, q^{max}, \dot{q}^{max}, \ddot{q}^{max}, \delta t, \bar{w}$
$\ddot{q}^{UB} \leftarrow [0, 0, 0, \ddot{q}^{max}]$
$\ddot{q}^{LB} \leftarrow [0, 0, 0, -\ddot{q}^{max}]$
$(\ddot{q}^{LB}[0], \ddot{q}^{UB}[0]) \leftarrow$ accBoundsFromPosLimits(...)+$(\bar{w}, -\bar{w})$
$\ddot{q}^{LB}[1] \leftarrow (-\dot{q}^{max} - \dot{q})/\delta t + \bar{w}$
$\ddot{q}^{UB}[1] \leftarrow (\dot{q}^{max} - \dot{q})/\delta t - \bar{w}$
$(\ddot{q}^{LB}[2], \ddot{q}^{UB}[2]) \leftarrow$ accBoundsFromViability(...)
return $\{\max(\ddot{q}^{LB}), \min(\ddot{q}^{UB})\}$

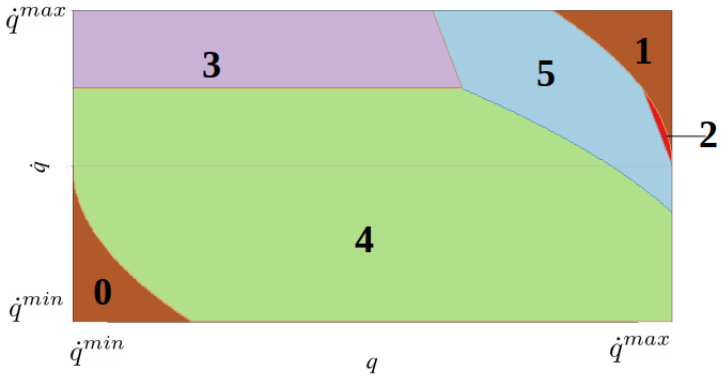


Figure 2.4: State space plot with $\bar{w} = 0$.

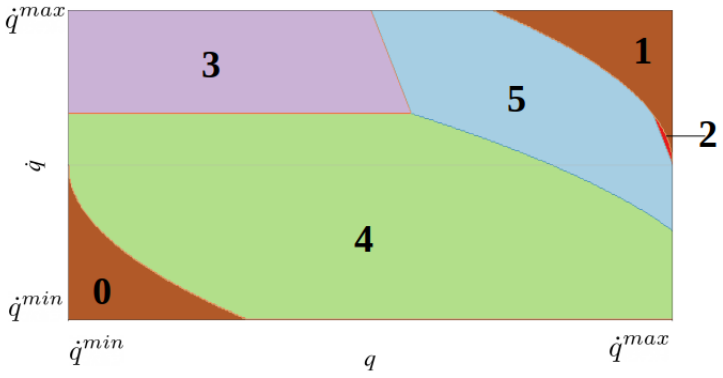


Figure 2.5: State space plot with $\bar{w} = \frac{1}{3}\ddot{q}^{max}$.

Feasible state space for $q^{max} = -q^{min} = 0.5$ rad, $\dot{q}^{max} = 2$ rad/s, $\ddot{q}^{max} = 10$ rad/s^2, and $\delta t = 0.1$s. Region 0 represents the unreachable space, while region 1 represents the space that is not viable. In each of the other four regions, a different acceleration upper bound dominates the others. In region 2 it is the one coming from the position inequality (2.33). In region 3 it is the one coming from the velocity inequality (2.45). In region 4 it is the acceleration upper bound $\ddot{q}^{max}$. In region 5 it is the one coming from the viability inequality (2.46)

can provide. For our tests we choose the same α value for all the joints to re-scale the maximum acceleration for each joint in the same manner. It is important to state the fact that the violation of a position constraint physically means hitting a physical component, a wall or simply reaching the maximum extension or rotation of a joint. The consequences may results in impacts, abrupt stop of the robotic system, vibrations and even failure of components. Our algorithms, plots and simulations do not try to reproduce these behaviours, but they simply allow bounds to be violated, with the purpose of comparing the performance of the various methods. The methods used for comparison are: i) the robust viability method, ii) the Naive method described in Section 1.2.1 and iii) the non-robust viability method described in Section 1.2.1. In all our tests we have verified that the conditions developed in Section 2.3.2, regarding the equivalence of discrete-time and continuous-time viability kernel, were satisfied. When the state is outside the viability kernel V, the robust and non-robust viability algorithms may give inconsistent values of lower and upper acceleration limits. Our main concern in these cases is to reach $\mathscr{V}$. Since the algorithm, as it is, may fail doing that, when a violation occurs we must add an escape condition, described in Alg. 4.

Algorithm 4 Dealing with non-viable states

Require: $q, \dot{q}, q^{min}, q^{max}, \dot{q}^{max}, \ddot{q}^{max}, \delta t, \bar{w}$
  if $IsStateViable(...) == False$ **then**
    if $(\dot{q} \geq 0 \wedge q \geq q^{min}) \quad \vee \quad q \geq q^{max})$ **then**
      $(\ddot{q}^{LB}, \ddot{q}^{UB}) \leftarrow (-\ddot{q}^{max}, -\ddot{q}^{max})$
    else
      $(\ddot{q}^{LB}, \ddot{q}^{UB}) \leftarrow (\ddot{q}^{max}, \ddot{q}^{max})$
    return$\{\ddot{q}^{LB}, \ddot{q}^{UB}\}$

Alg. 4 is based on the following reasoning.

- If the velocity is positive and we are above the lower position bound apply the maximum deceleration $\ddot{q} = -\ddot{q}^{max}$ to stop the joint as soon as possible.
- If the joint is above the upper position bound apply the maximum deceleration $\ddot{q} = -\ddot{q}^{max}$ to re-enter the viability kernel as soon as possible.
- If the velocity is negative and we are below the upper position bound apply the maximum acceleration $\ddot{q} = \ddot{q}^{max}$ to stop the joint as soon as possible.
- If the joint is below the lower position bound apply the maximum acceleration $\ddot{q} = \ddot{q}^{max}$ to re-enter the viability kernel as soon as possible.

2.4.1 Simple joint

This section deal with a robotic system composed by only one joint. These tests try to reach the upper position bound without exceeding it or violating the velocity limit. The limits used for this test are:

$$\begin{cases} q^{max} = 2[\text{rad}] \\ q^{min} = -2[\text{rad}] \\ |\dot{q}^{max}| = 5[\text{rad/s}] \\ \\ \delta t = 0.1[\text{s}] \end{cases}$$

In the first test we always apply to the joint the maximum acceleration allowed (as computed by the algorithms). The system is subject to a uniform distributed random disturbance w_i, bounded between $[-\bar{w}, \bar{w}]$, applied at every time step of the simulation. Fig. 2.6 shows that with the non-robust viability approach the position bound is violated. Fig. 2.7 shows instead the results with the robust viability approach: the joint never reaches the bound, but it comes close to it and then fluctuates, leading to no constraint violations.

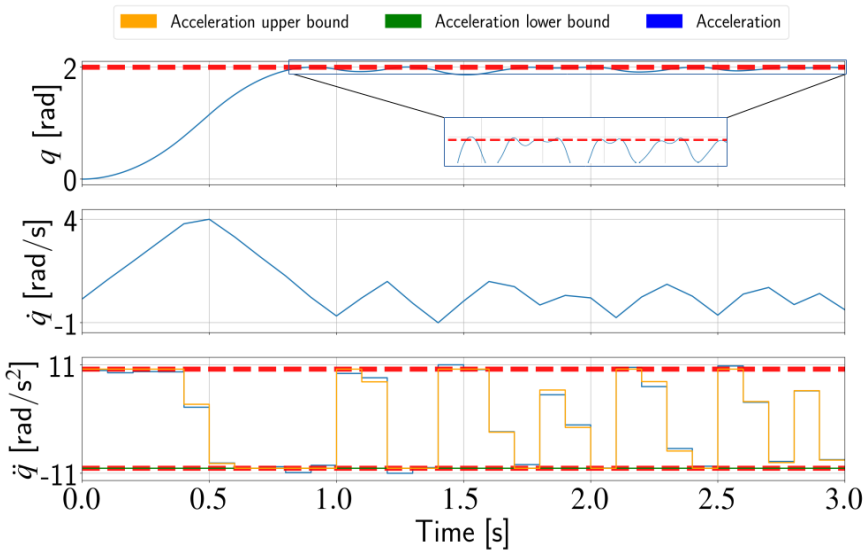


Figure 2.6: Test non-robust viability approach: joint have to reach the upper bound while subject to a random disturbance w_i.

In the second test, shown in Fig. 2.8 and Fig. 2.9, the maximum allowable disturbance $\bar{w}$ is applied at every time step. It is possible to notice that for the robust method the joint comes closer to the limit than it did in the previous test (Fig. 2.7),

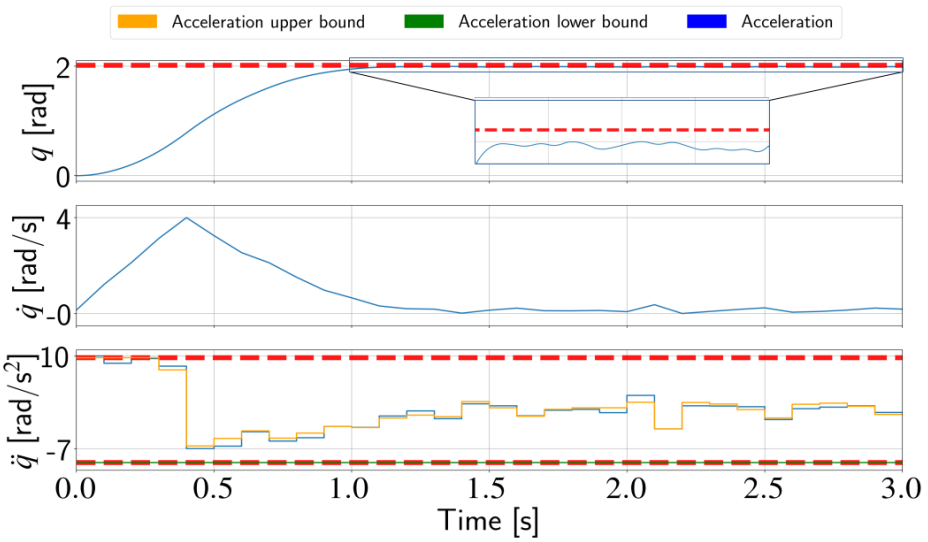


Figure 2.7: Test robust viability approach: joint have to reach the upper bound while subject to a random disturbance w_i.

but then it starts oscillating. The oscillating behaviour was also observed in the tests of the viability method in [12], and it is due to the discrete-time nature of the controller. The non-robust method instead comes to a clear violation of the constraint.

Another interesting test to prove the robustness of the algorithm is to apply at every time step the maximum values of the disturbance, $\bar{w}$ or $-\bar{w}$, depending on the velocity sign. We add $\bar{w}$ if we have a positive velocity, and we add $-\bar{w}$ with negative velocity: this leads to an unstable behaviour. If velocity is positive, the disturbance is going to increase the acceleration towards the upper bound, so increasing a positive acceleration given by the joint motor or reducing the deceleration, pushing the joint against the limit. In the same manner, when velocity becomes negative and the joint starts to move towards the lower limit, the disturbance switches sign and starts to push the joint against it. The test in Fig. 2.9 shows us no constraint violation.

2.4.2 6-DoF Baxter robot

In these tests we applied disturbances on every joint according to its own maximum allowable acceleration. In the following we show only the joints with the most relevant behaviours. Tests are divided in two categories:

- The joints have to reach a desired position in joint space where the position of the joints is unfeasible: $q^d \notin X$.

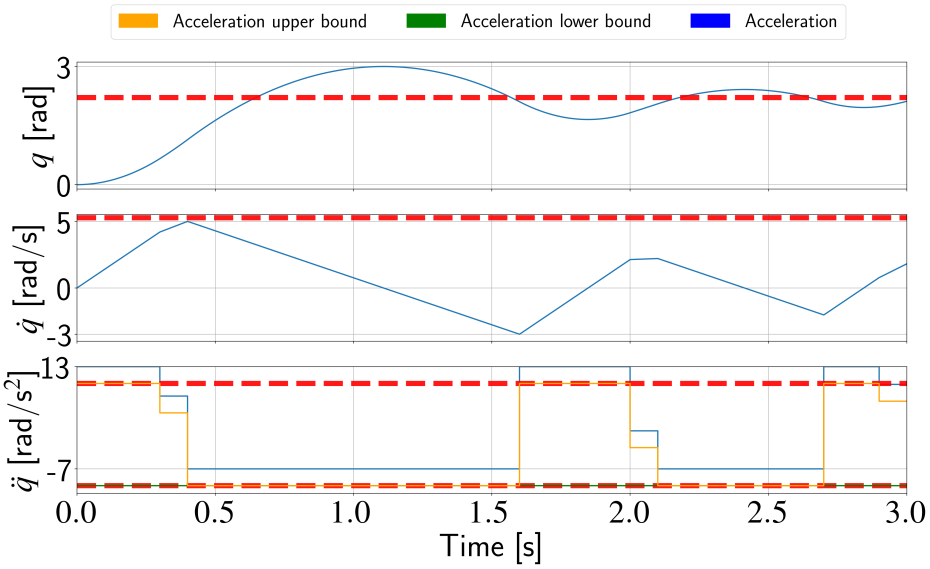


Figure 2.8: Test non-robust viability approach: the joint has to reach the upper bound while subject to a disturbance $\bar{w}$.

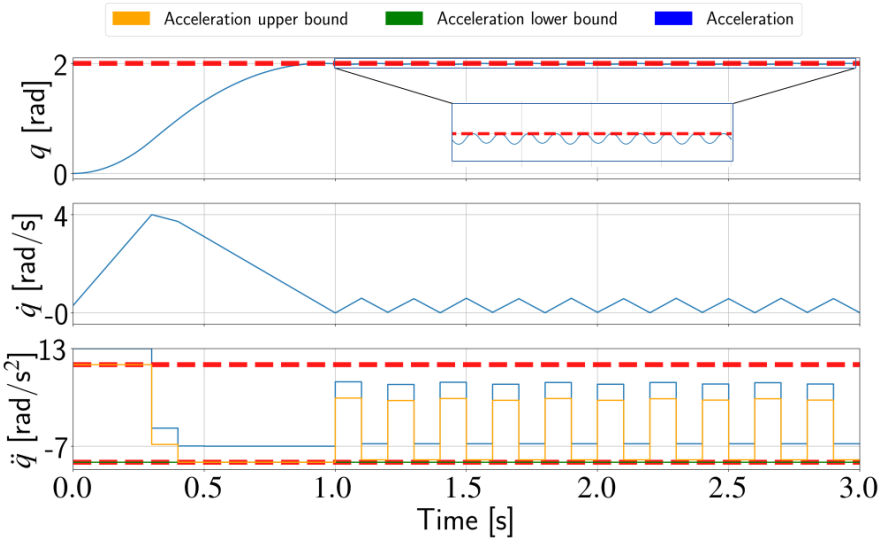


Figure 2.9: Test robust viability approach: the joint has to reach the upper bound while subject to a disturbance $\bar{w}$.

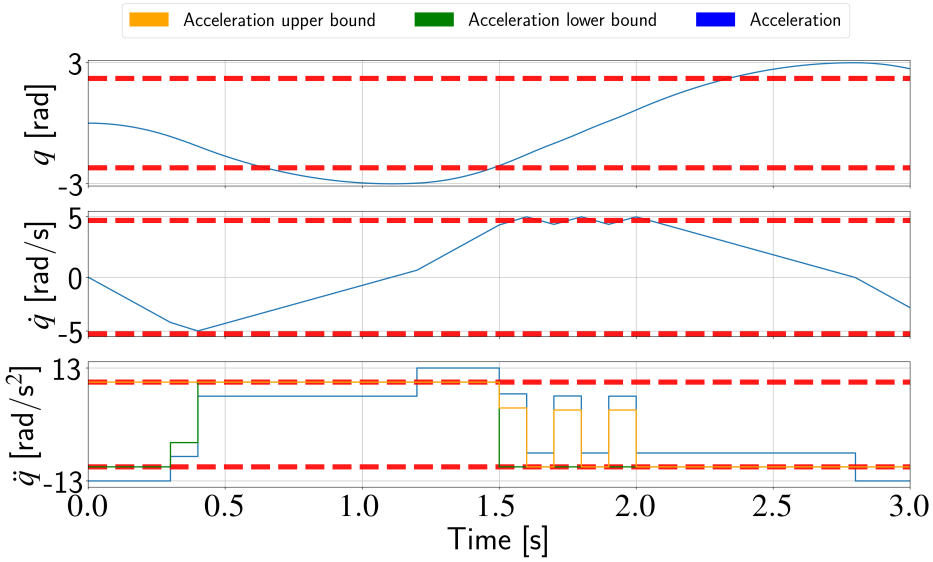


Figure 2.10: Test of the non-robust method with $w_t = \pm\bar{w}$ based on velocity sign for each time step.

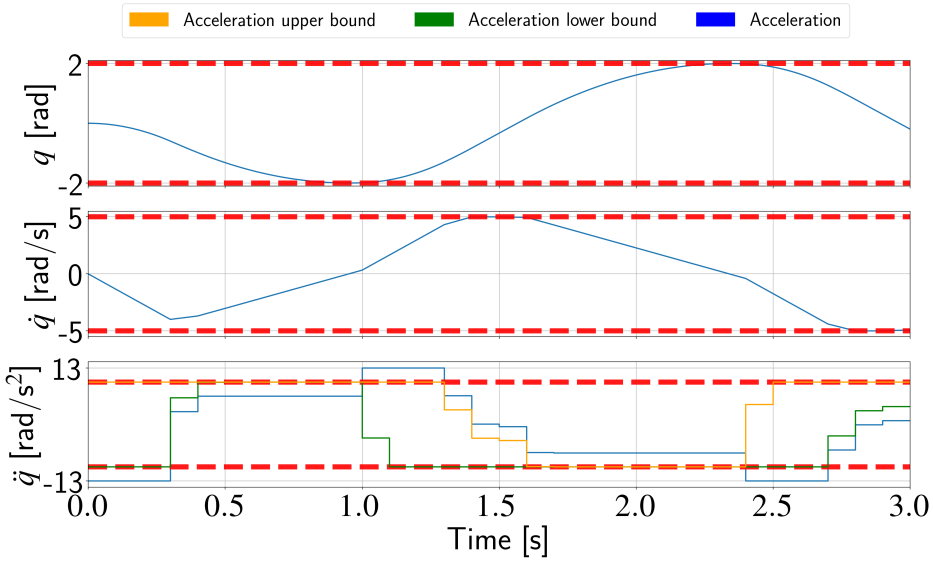


Figure 2.11: Test of the robust method with $w_t = \pm\bar{w}$ based on velocity sign for each time step.

- The end-effector has to reach a desired position in Cartesian space that is outside of its work-space.

Joint space tests

For these tests the desired position to reach is not feasible: $q^d \notin X$. In particular we set q^d for the first joint, referred from now on as *joint 0*, above its upper limit (i.e. 0.5 rad over the position limit). While the acceleration applied in the single-joint tests is directly the maximum allowable, for the Baxter robot the desired acceleration is given by a *PD* control law.

$$\ddot{q}^d = k_P(q^d - q) - k_D\dot{q} \tag{2.54}$$

where $k_P, k_D \in \mathbb{R}^+$. In second place the resulting acceleration (2.54) is compared with the limit generated by Alg. 3 and potentially truncated. This corresponds to computing the joint acceleration by solving this Quadratic Program (QP):

$$\begin{aligned} \underset{\ddot{q}}{\text{minimize}} \quad & ||\ddot{q}^d - \ddot{q}||^2 \\ \text{subject to} \quad & \ddot{q}^{lb} \leq \ddot{q} \leq \ddot{q}^{ub} \end{aligned} \tag{2.55}$$

The values used in our tests are $k_P = 1000$, $k_D = 2\sqrt{k_P}$, and $\delta t = 0.05s$. The disturbance w applied for these tests is the maximum allowable for each joint, so with $\alpha = \frac{1}{3}$. It is possible to notice in Fig. 2.12 and Fig. 2.13 that the use of the naive method leads to a violation of both velocity and position constraints, as expected, since this method is prone to violations even without disturbances. Also, the amplitude of fluctuations around the the upper position limit is strongly dependent to the k_p, k_D values.

Fig. 2.14 and Fig. 2.15 show the results obtained with the non-robust viability algorithm. It is possible to notice multiple violations of the constraints. The joint violates first the velocity limit, i.e. it leaves the viability kernel and so the control chosen with Alg. 4 tries to lead the joint again in the viability kernel. This generates fluctuation on velocity above the upper position limit in the first part of the graph. Later, the joint violates the position bound. Violation has a smaller value respect to the naive method and still fluctuates around the upper position limit.

Fig. 2.16 and Fig. 2.17 shows the results obtained with the robust viability algorithm. The bounds are not violated, but the oscillation on acceleration in the final part are greater than in the naive or non-robust case. This behaviour was expected, as it was documented also for the case without disturbance in the non-robust algorithm [12].

Fig. 2.18, 2.19, 2.20, 2.21, 2.23 and 2.22 reports tests where the disturbance was chosen according to the sign of velocity of each joint, as previously explained

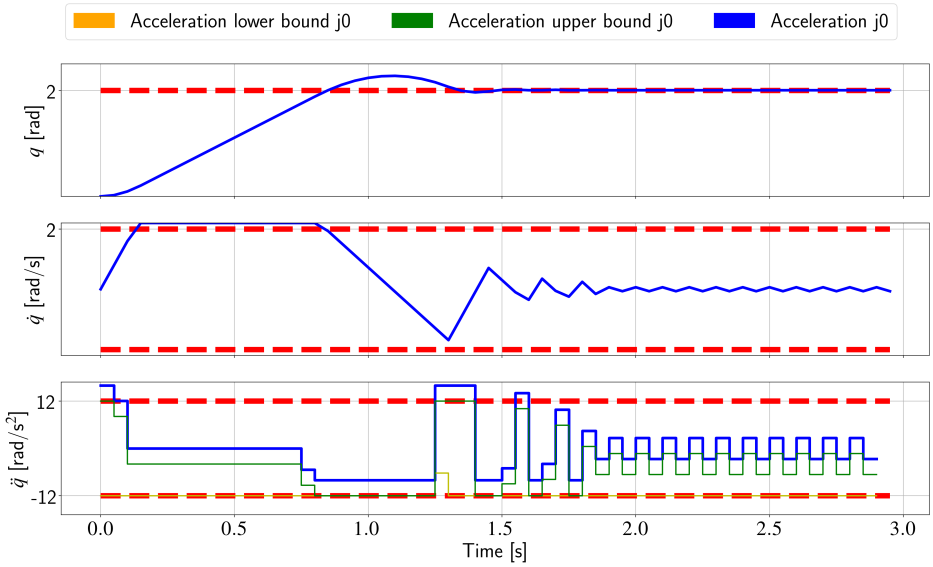


Figure 2.12: Naive method.

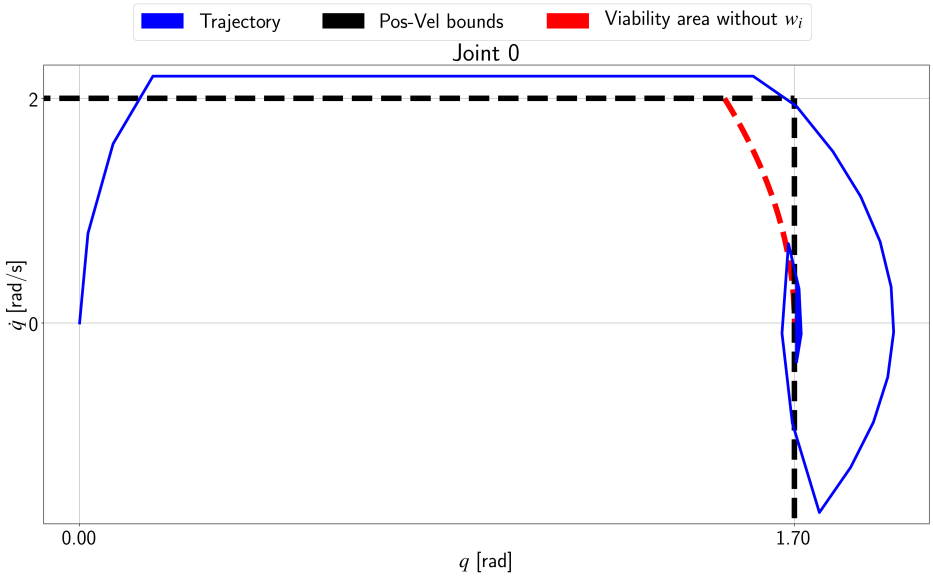


Figure 2.13: Naive state-space trajectory.

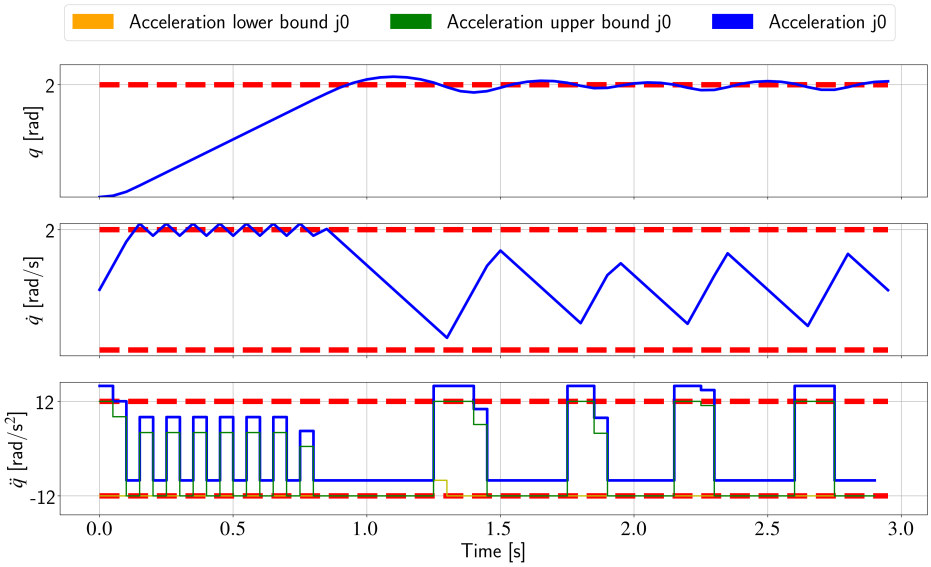


Figure 2.14: Viability non-robust method.

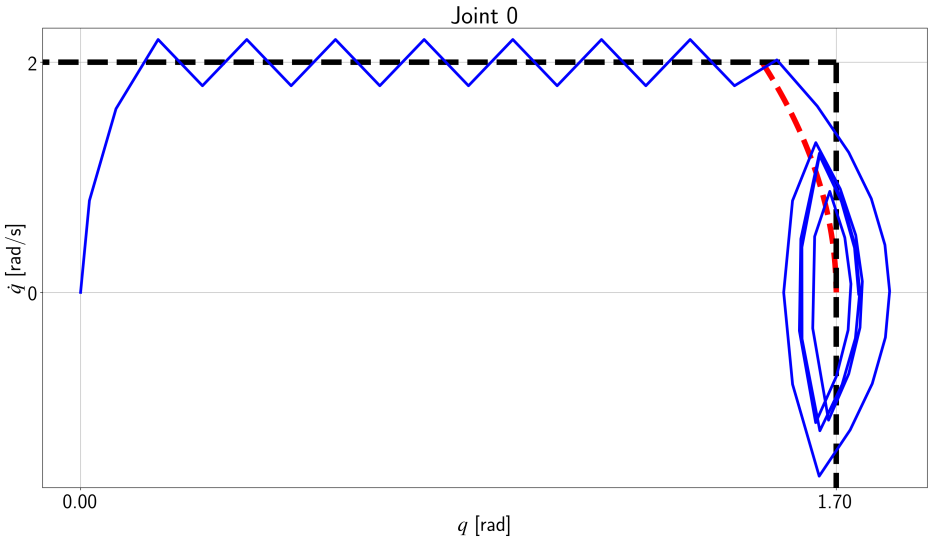


Figure 2.15: Viability non-robust method state-space trajectory.

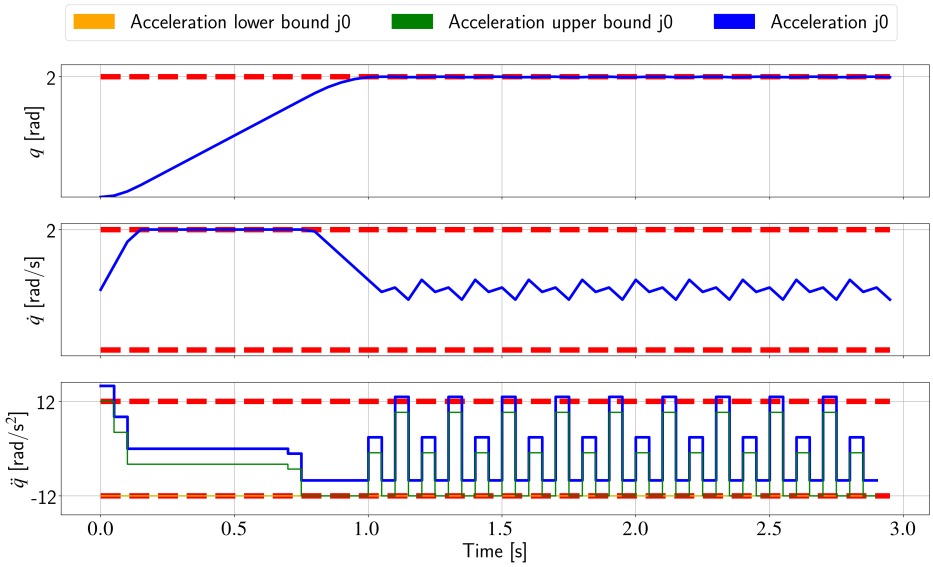


Figure 2.16: Viability robust method.

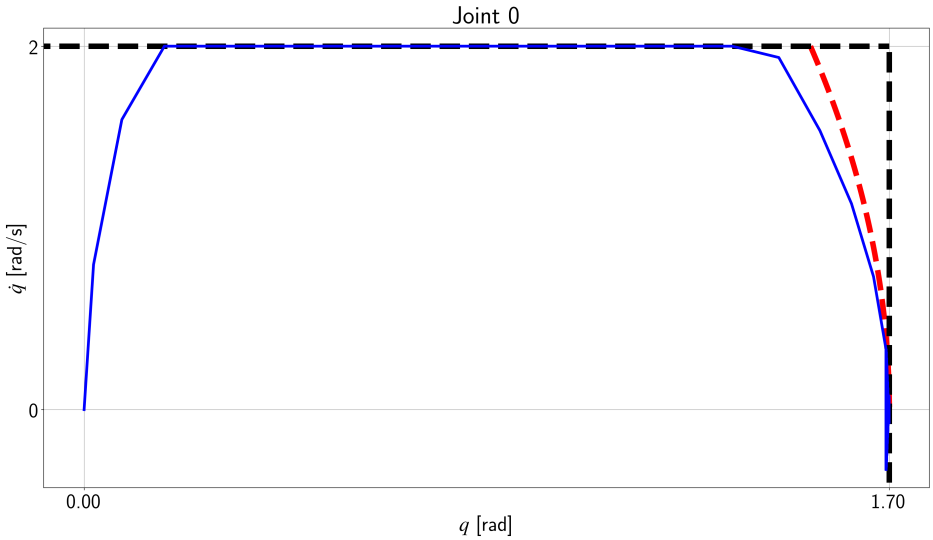


Figure 2.17: Viability robust method state-space trajectory.

in the single-joint case. As clearly shown, the Naive method cannot deal with these disturbances and once it violates the viability region it cannot come back. With the non-robust viability method we can clearly see that the position and velocity bounds are repeatedly violated, but still the algorithm tries to reach again the viable region and again violations have smaller value with respect to the naive case. The robust viability method instead does not violate any bound.

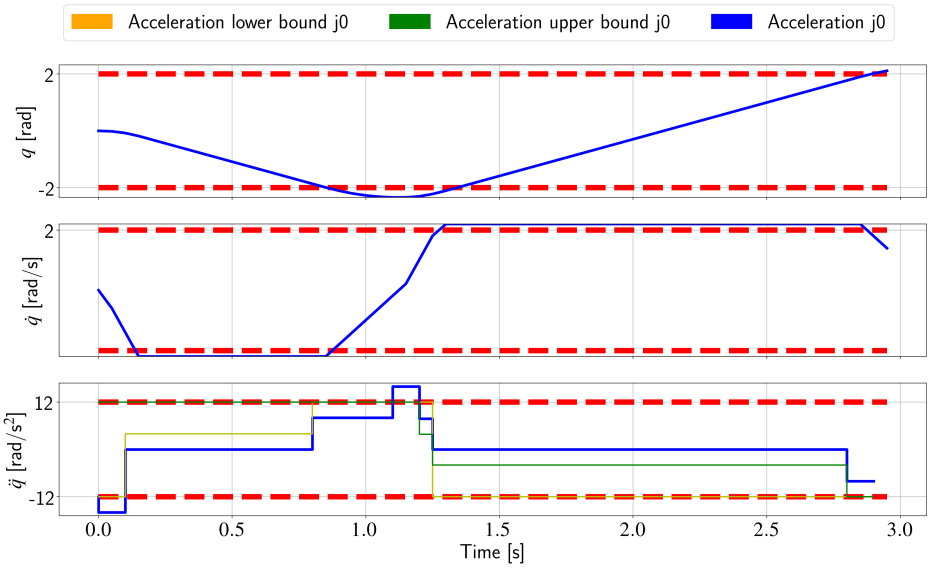


Figure 2.18: Test with naive method and $w_t = \pm\bar{w}$ based on velocity sign on Baxter robot joint.

Cartesian space tests

For these tests we set a desired configuration q^d outside of the work-space of the Baxter robot. The applied disturbance is $\bar{w}$ on each joint. We used a task-space inverse dynamics controller [13] to reach a desired configuration with the end-effector of Baxter's arm. At the same time, a lower-priority joint-space task is used to stabilize the null space of the end-effector. At each control loop we computed the desired joint torques by solving the following Quadratic Program:

$$\begin{aligned} \underset{\ddot{q}}{\text{minimize}} \quad & ||\dot{v}^d - J(q)\ddot{q} - \dot{J}(q,\dot{q})\dot{q}||^2 + a||\ddot{q}^p - \ddot{q}||^2 \\ \text{subject to} \quad & \ddot{q}^{lb} \leq \ddot{q} \leq \ddot{q}^{ub} \end{aligned} \tag{2.56}$$

where $(\mathbf{q}) \in \mathbb{R}^{6\times n}$ is the end-effector Jacobian, and $a \in \mathbb{R}$ is the postural-task weight. The desired end-effector acceleration and joint accelerations are com-

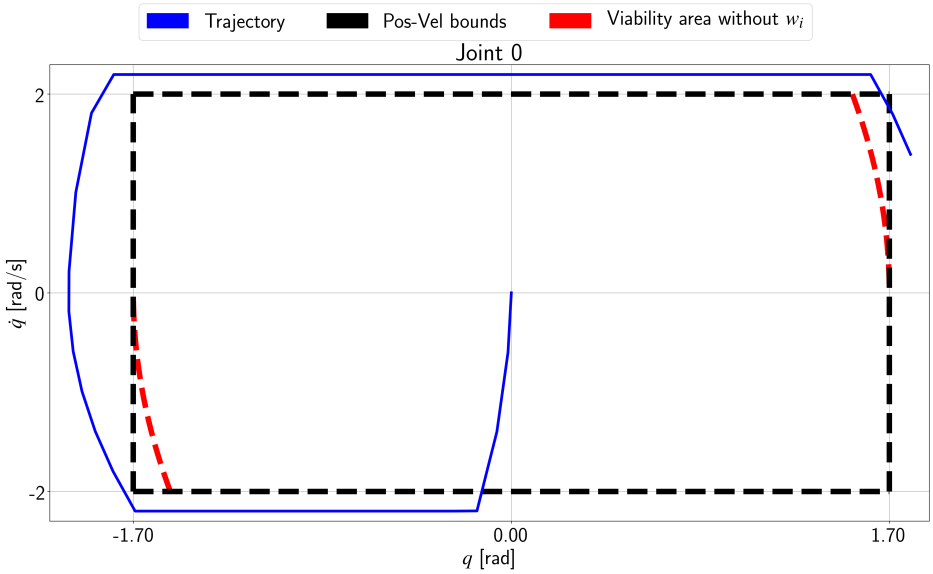


Figure 2.19: Joint state-space with naive method and $w_t = \pm\bar{w}$ based on velocity sign.

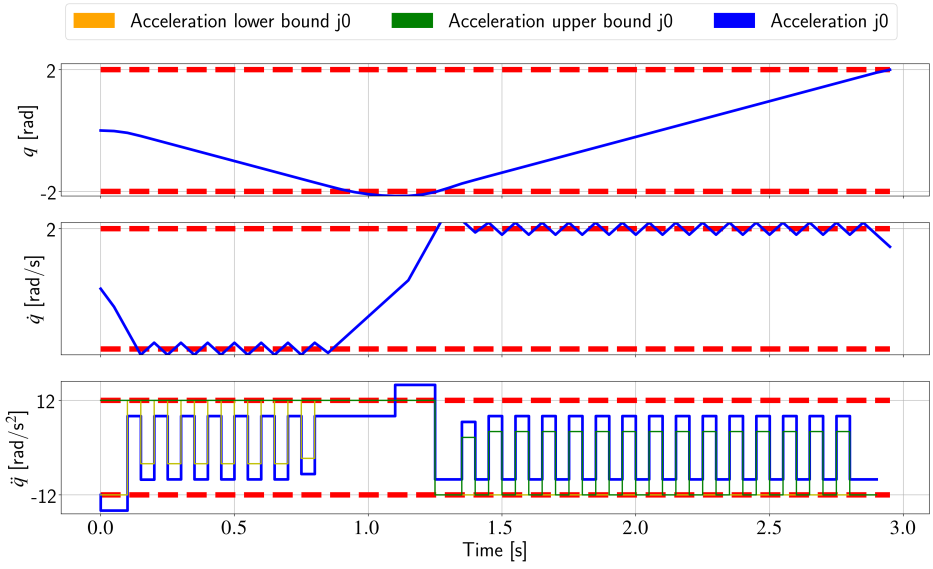


Figure 2.20: Test with viability non-robust method and $w_t = \pm\bar{w}$ based on velocity sign on Baxter robot joint.

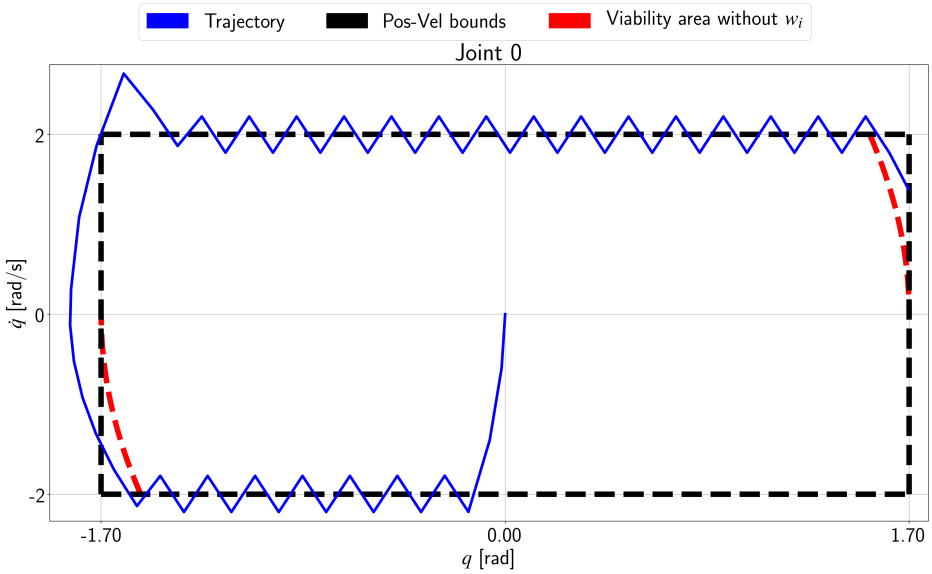


Figure 2.21: Joint state-space with viability non-robust method and $w_t = \pm\bar{w}$ based on velocity sign.

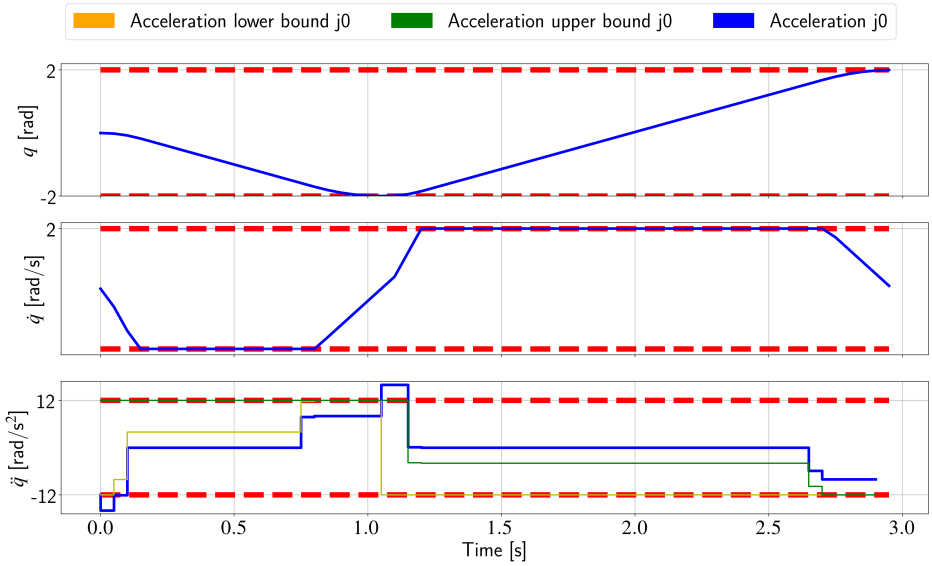


Figure 2.22: Joint state-space with viability robust method and $w_t = \pm\bar{w}$ based on velocity sign.

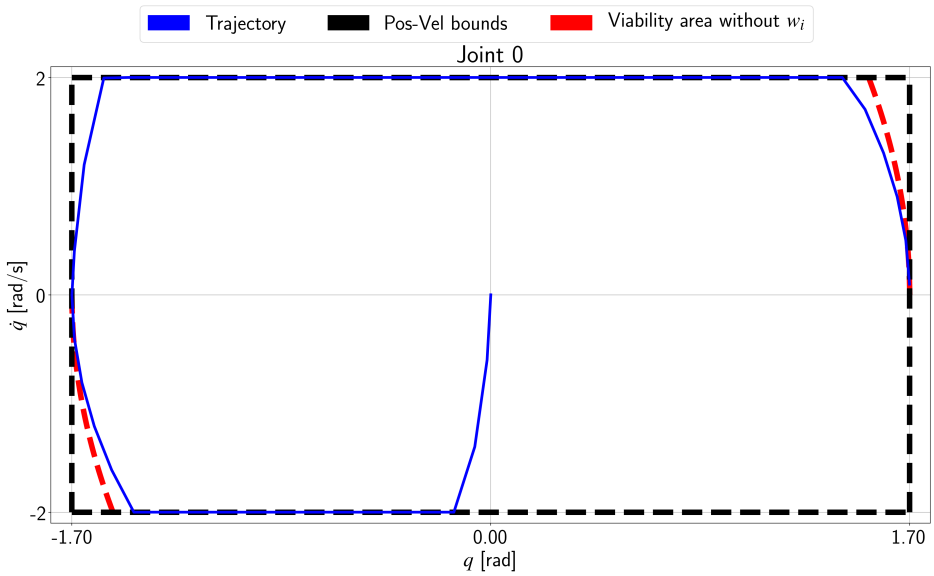


Figure 2.23: Joint state-space with viability robust method and $w_t = \pm\bar{w}$ based on velocity sign.

puted as:

$$\begin{aligned} \dot{v}^d &= k_p \log(H(q)^{-1} H^d) - k_d J(q)\dot{q} \\ \ddot{q}^p &= k_p(q^r - q) - k_d \dot{q}, \end{aligned} \tag{2.57}$$

where $k_p, k_d \in \mathbb{R}^+$ are the proportional and derivative gains, and $H, H^d \in SE(3)$ are respectively the measured and desired rigid transformation from world to end-effector frame. The values used for this simulations are $k_p = 10$, $k_d = 2\sqrt{k_p}$, $\delta t = 0.01$ s, $a = 10^{-3}$.

We can notice that in Fig. 2.24, Fig. 2.27 and Fig. 2.30 the end-effector cannot reach such configuration and it stops at the nearest spot in its work-space, fluctuating due to the disturbances. We can notice that in Fig. 2.25, Fig. 2.28 and Fig. 2.31 the desired acceleration in Cartesian space given by (2.57) and the effective acceleration limited due to our methods or due to the maximum acceleration that every joint can provide coming from (2.56). In Fig. 2.26, Fig. 2.29 and Fig. 2.32 instead we can see the behaviour of joint position, velocity and acceleration. It is possible to notice that both the naive and non-robust viability method lead to violations of the velocity limits.

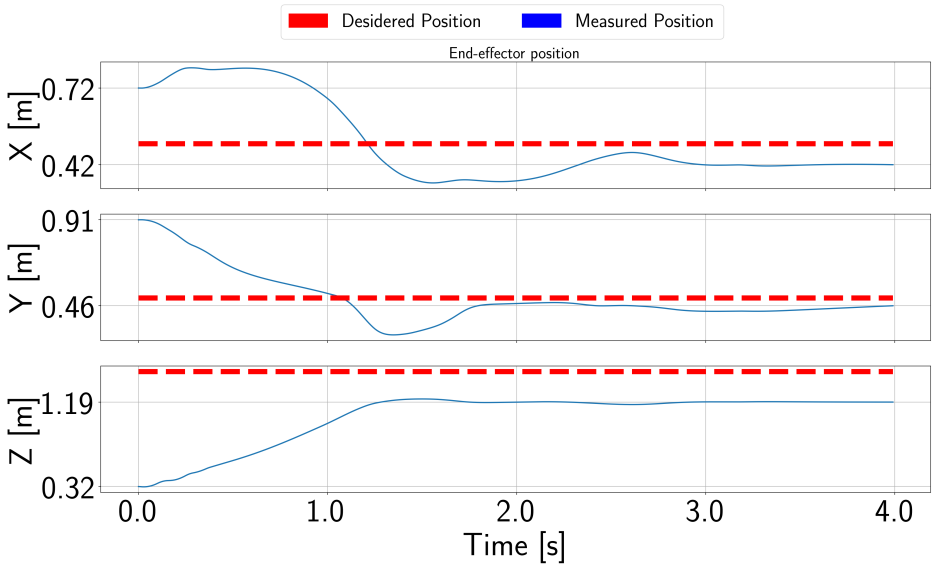


Figure 2.24: Cartesian space position of end-effector with $w_t = \pm\bar{w}$ based on velocity sign for Naive method.

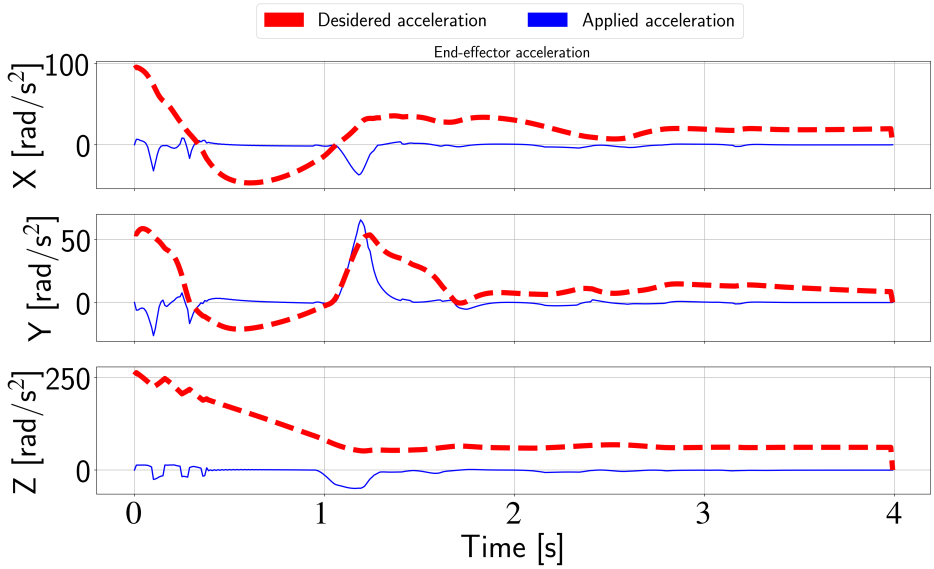


Figure 2.25: Cartesian space acceleration of end-effector with $w_t = \pm\bar{w}$ based on velocity sign for Naive method.

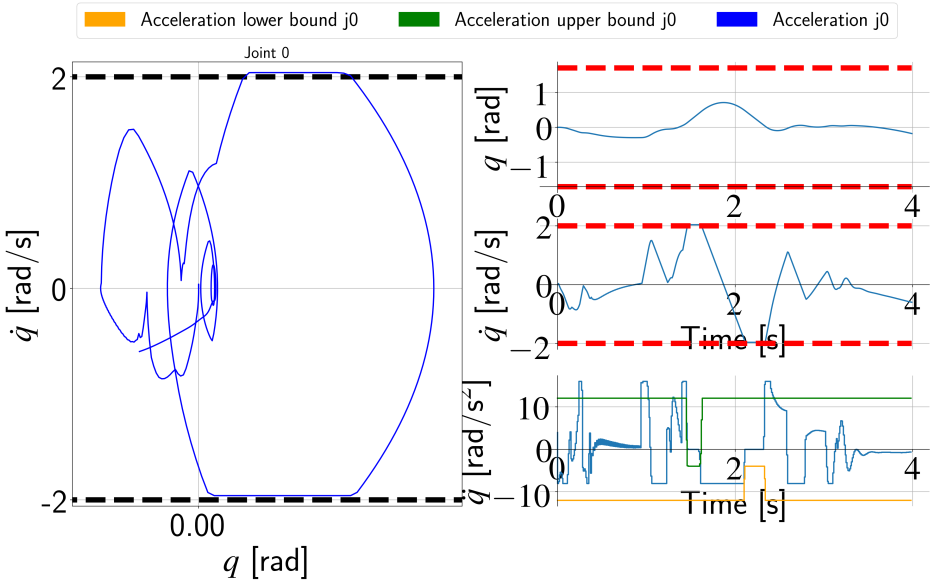


Figure 2.26: Joint state-space with $w_t = \pm\bar{w}$ based on velocity sign for Naive method.

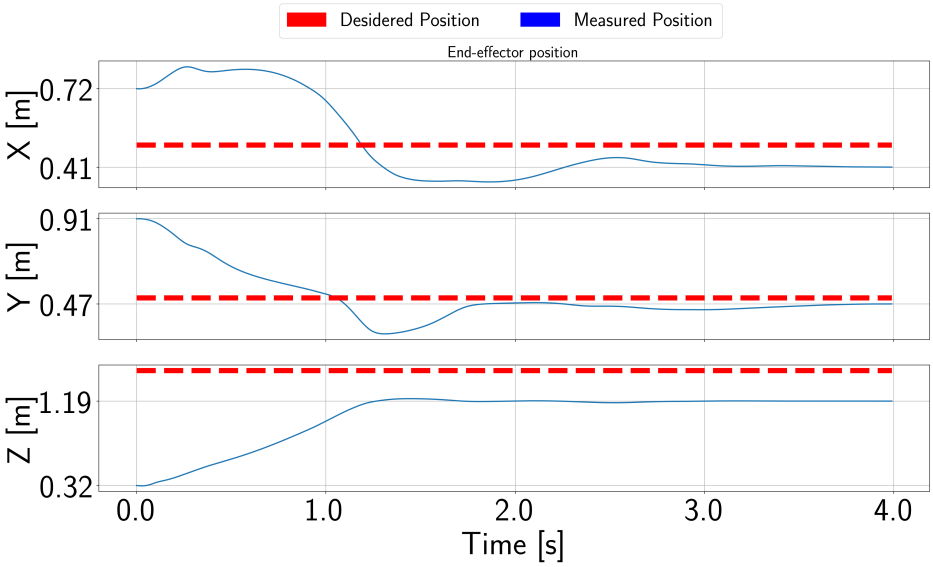


Figure 2.27: Cartesian space position of end-effector with $w_t = \pm\bar{w}$ based on velocity sign for non-robust method.

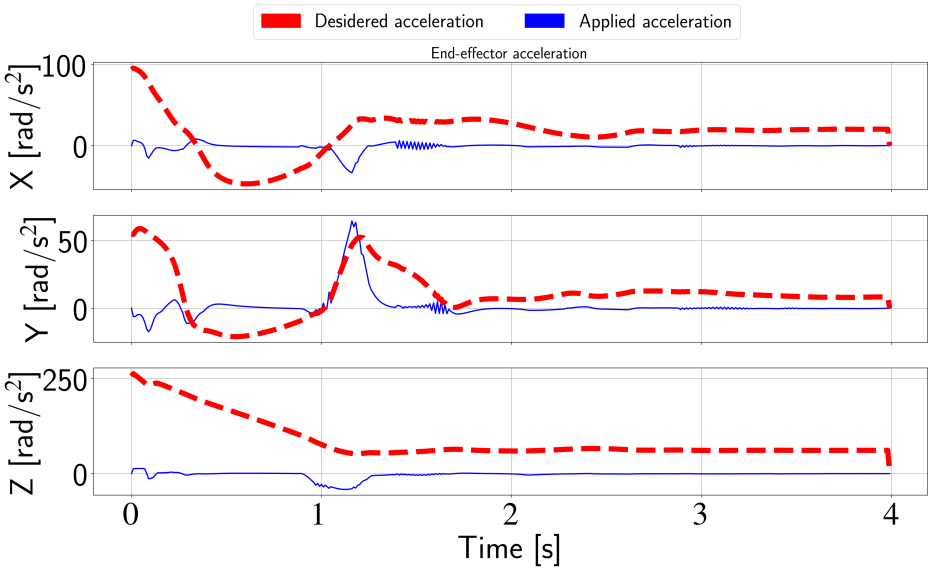


Figure 2.28: Cartesian space acceleration of end-effector with $w_t = \pm\bar{w}$ based on velocity sign for non-robust method.

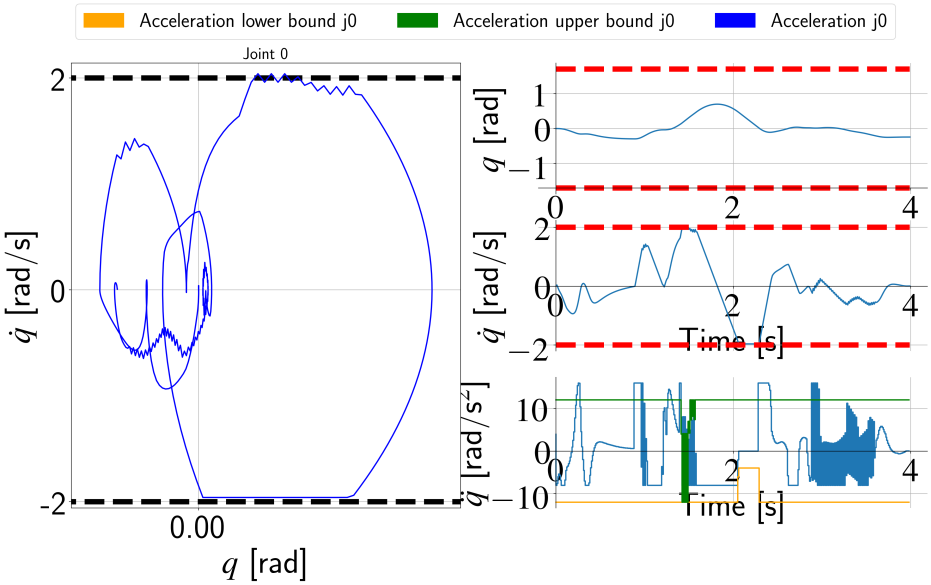


Figure 2.29: Joint state-space with $w_t = \pm\bar{w}$ based on velocity sign for non-robust method.

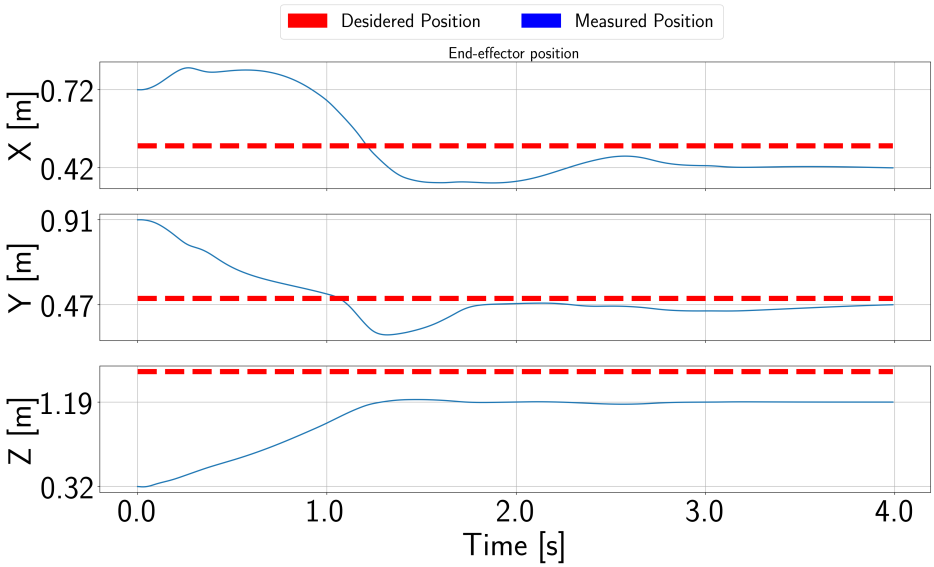


Figure 2.30: Cartesian space position of end-effector with $w_t = \pm\bar{w}$ based on velocity sign for robust method.

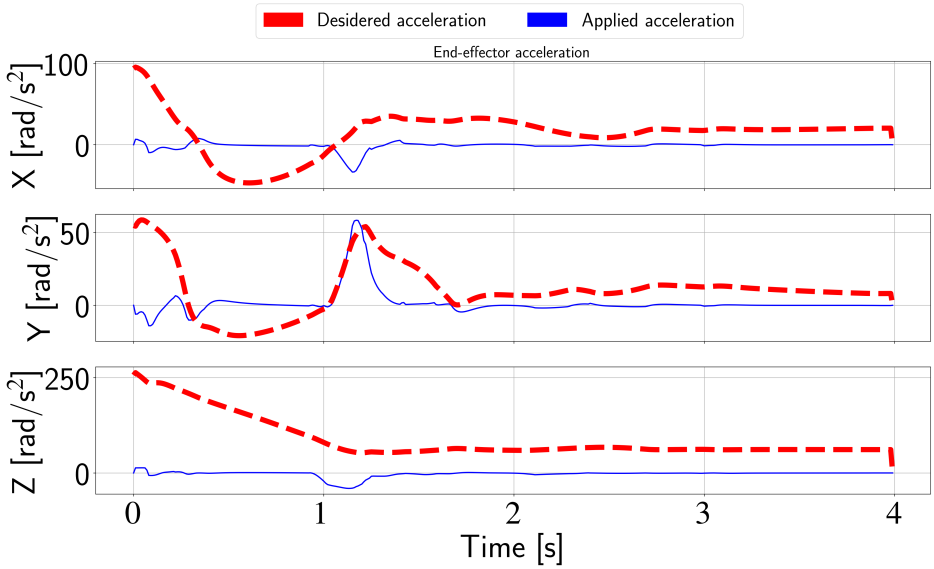


Figure 2.31: Cartesian space acceleration of end-effector with $w_t = \pm\bar{w}$ based on velocity sign for robust method.

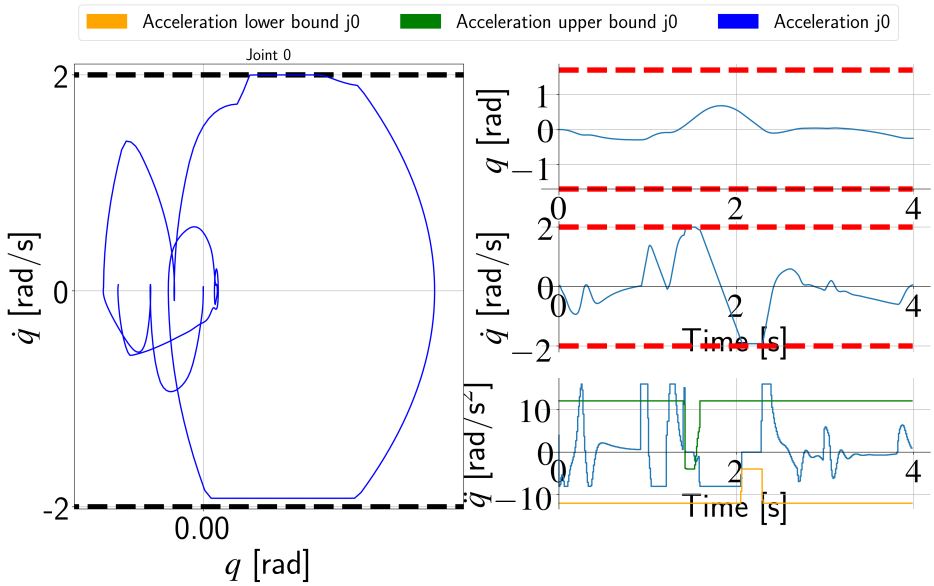


Figure 2.32: Joint state-space with $w_t = \pm\bar{w}$ based on velocity sign for robust method.

Chapter 3

Viability with joint torque limits

3.1 Problem statement

The problem considered in the previous chapter has the drawback of considering that a robotic manipulator can deliver the same amount of acceleration at every state. However, in practice, this is not the case. The motors that actuate the joints have a nominal maximum torque value that cannot be exceeded since it can lead to thermal failure of the motor components due to high current. In order to account for this fact, in this chapter we are going to focus on a more realistic (and complex) model of a robot manipulator, whose dynamic equations can be written as:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q), \quad \tau^{min} \leq \tau \leq \tau^{max} \tag{3.1}$$

where:

- $M(q) \in \mathbb{R}^{n \times n}$ is the mass matrix, which depends on the mass distribution of the system and on the robot configuration q;
- $C(q,\dot{q})\dot{q} \in \mathbb{R}^n$ accounts for Coriolis and centrifugal effects;
- $g(q) \in \mathbb{R}^n$ contains the gravity forces;
- $\tau \in \mathbb{R}^n$ represents the torques given by the motors, bounded between the two nominal working values tau^{min} and τ^{max}

The sum of $C(q,\dot{q})\dot{q}$ and $g(q,\dot{q})$ can be also referred to as "bias forces", denoted as: $h(q,\dot{q})$. Since the mass matrix $M(q)$ is always invertible, the joint accelerations can be computed as:

$$\ddot{q} = M(q)^{-1}(\tau - (C(q,\dot{q})\dot{q} + g(q)), \tag{3.2}$$

From (3.2) it is clear that the acceleration that a joint can reach depends (nonlinearly) on the positions and velocities of the system.

The problem that we would like to tackle consists in computing a state-space region such that, if the initial state is in that region, it is possible to keep it in that region. Such a region is called a *control invariant set* and its computation has been thoroughly investigated in the literature [16, 15, 6, 21, 10, 41, 9, 5, 7, 8, 19, 31]. Even though this is a long-standing and well-known problem, to the best of our knowledge, no approach is capable of computing such sets for arbitrary high-dimensional (i.e. more than 4 states) nonlinear systems.

In this chapter we are going to analyze the problem, showing different approaches as explained in the literature and trying to use them to deal with the specific nonlinear system under study (i.e., a robot manipulator). First, we show how to solve the problem for linear systems (as formulated in the previous chapter) by means of a well-known iterative method [16], and we highlight how this approach is not suitable for our problem. Second, we discuss a method able to deal with nonlinear systems [6] and try to apply this method to our case. Even in this case, the method is not directly applicable to our problem, so we try to extend it exploiting the specific features of the system under study.

3.2 State-of-the-art approaches

3.2.1 Linear Dynamics — Maximal output set

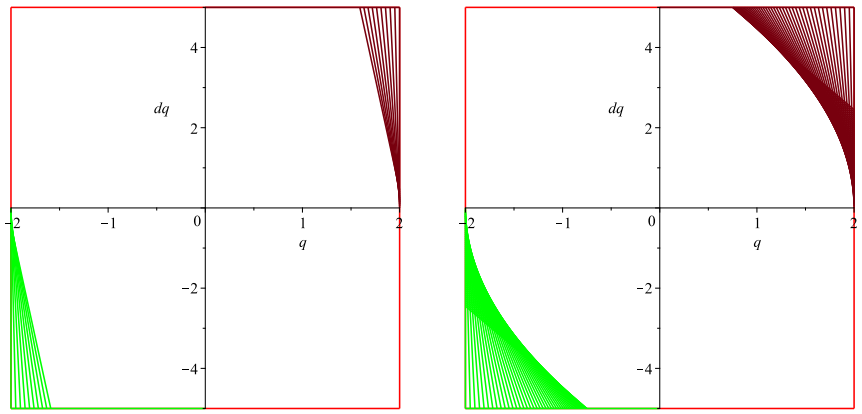
The robotic system with acceleration limits and without torque limits considered so far is a linear system subject to linear control and state constraints. [16] developed a method to compute a maximal output admissible set for such systems, with dynamics:

$$x^+ = Ax$$

The maximal output admissible set, referred to in the paper as O_∞, is the maximal set such that the output $y = Cx$ remains inside a given set Y forever. the set Y is represented by a set of nonlinear functions: $Y = \{y : f_i(y) \leq 0, \quad i = 1, \ldots, s\}$. The maximal output admissible set O_∞ is a control invariant set, defined as:

$$O_\infty(A, C, Y) = \{x \in \mathbb{R}^n : \quad CA^t x \in Y \quad \forall t \in \mathbb{N}^+\}, \tag{3.3}$$

The conceptual algorithm proposed is the following: starting from $t = 0$, the algorithm computes O_{t+1} for increasing values of t until $O_{t+1} = O_t$, then it stops and it declares $O_\infty = O_t$. This operation is implemented by solving a set of mathematical



(a) 10 steps (b) 50 steps

Figure 3.1: Convergence of the algorithms at various steps.

programming problems.

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & J_i(x) = f_i(CA^{t+1}x) \\ \text{subject to} \quad & f_j(CA^k x) \leq 0, \quad j = 1,\ldots,s, \quad k = 0,\ldots,t \end{aligned} \tag{3.4}$$

At every cycle the algorithm performs a series of minimization in the form of (3.4). By mean of $J_i(x)$ it is possible to compute the contour of O_i, i.e., the contour is represented by a series of the $f_i(CA^t x)$ active constraints of the problem. Let J_i^* be the maximum value of $J_i(x)$. If $J_i^* \leq 0$ for $i = 1,\ldots,s$, then the algorithm stops, it sets $t^* = t$ and it defines O_∞ as

$$O_\infty = \{x \in \mathbb{R}^n : f_i(CA^t x) \leq 0, \quad i = 1,\ldots,s, \quad t = 0,\ldots,t^*\}$$

The algorithm is able to stop only if O_∞ is finitely determined.

A brief numerical example

Considering the simple joint case, as in 2.2, without the disturbances it is possible to define

$$A = \begin{bmatrix} 1 & \delta t & \frac{\delta t^2}{2} \\ 0 & 1 & \delta t \\ 0 & 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad x = \begin{bmatrix} q \\ \dot{q} \\ \ddot{q} \end{bmatrix} \quad X_{limit} = \begin{bmatrix} q_{limit} \\ \dot{q}_{limit} \\ \ddot{q}_{limit} \end{bmatrix} \tag{3.5}$$

where the state has been extended to include the acceleration. This allows us to model the acceleration limits as state bounds. With this model, we can initialize

The vector of position, velocity, and acceleration bounds X_{limit} is arbitrary chosen with the values of $[2,5,10]$ for this example. Fig. 3.1 shows the convergence of the algorithm at two different iterations. At convergence, the algorithm gives, by mean of the constrains, the same viability contour found for the simple joint case with the viability method. The algorithm has a slow convergence, and it requires more computation time with respect to the one proposed in the beginning of this manuscript. This section has the only intent to show possible alternatives to compute a control invariant set for a linear system that were investigated.

Gilbert-Tan with torque limits

The Gilbert-Tan algorithm is also able to compute a maximal output admissible set for a linear system with nonlinear constrains. We can write the torque limits as nonlinear constraints of the system state:

$$\tau^{min} \leq M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \leq \tau^{max} \tag{3.6}$$

One of the limitations of the Gilbert-Tan algorithm is that, if the constraints $f_i(CA^t x)$ are not convex, then the algorithm requires solving nonconvex optimization problems, which is not possible in general. Since is not possible to ensure that the dynamic of a generic robotic system is a convex function, we cannot rely on this method for tackling the viability problem with torque limits.

3.2.2 Nonlinear dynamics — One-step reachable set

A method to compute 1-step reachable sets for constrained nonlinear systems was developed in [6]. This algorithm can in turns be used to compute the viability kernel (even though it is not the most efficient way to do it). The method consists in two algorithms. Before briefly explaining the two algorithms it is necessary to introduce a key tool used by this method: intervals arithmetic.

Interval arithmetic

Interval arithmetic is an arithmetic defined on sets of intervals, rather than sets of real numbers. It is useful to guarantee a certain degree of robustness in dealing with rounding errors or measurement errors in mathematical computation. The main goal of this arithmetic is to provide a simple method to calculate the upper and lower bounds of a function on a given interval of its input variables. The interval operators are recalled hereafter

- **Addition:** $[x_1,x_2] + [y_1,y_2] = [x_1 + y_1, x_2 + y_2]$
- **Subtraction:** $[x_1,x_2] - [y_1,y_2] = [x_1 - y_1, x_2 - y_2]$

- **Multiplication:** $[x_1,x_2] \times [y_1,y_2] =$

(3.7)

- **Division:** $\frac{[x_1,x_2]}{[y_1,y_2]} = [x_1,x_2] \times \frac{1}{[y_1,y_2]}$ where

$$\frac{1}{[y_1,y_2]} = [\frac{1}{y_2},\frac{1}{y_1}] \quad 0 \notin [y_1,y_2]$$

$$\frac{1}{[y_1,0]} = [-\infty,\frac{1}{y_1}]$$
$$\frac{1}{[0,y_2]} = [\frac{1}{y_2},\infty]$$
$$\frac{1}{[y_1,y_2]} = [-\infty,\frac{1}{y_1}] \cap [\frac{1}{y_2},\infty] \quad 0 \in (y_1,y_2)$$

Any function can be extended to be applied on intervals, and it is then called an *interval extension function*. In case of monotonic functions the process is straightforward; due the monotonicity, the extreme of the output interval are simply given by evaluating the function at the extremes of the input interval. In general the process, even if not in a easy way, can be extended to any function f resulting in what is called an interval extension of f. In the following we will refer to the interval extension of a function f as $f\square$.

Divide and conquer algorithm (Bravo-2005)

This section explains the method by [6] starting from its assumptions and its definitions. It has to be considered a system described by a nonlinear discrete model.

$$x_{k+1} = f(x_k,u_k), \tag{3.8}$$

where $x_k \in \mathbb{R}^n$ is the system state and $u_k \in \mathbb{R}^m$ is the control signal at sample time k. The system state and control can be constrained:

$$x_k \in X \quad u_k \in U, \tag{3.9}$$

where X,U represent the feasible state and control regions, respectively. Those regions must be compact sets that contain the origin. $f(\cdot,\cdot)$ is assumed to be continuous in the $X \times U$ space.

Let us now define *control invariant sets*. A set $\Omega \subset X$ is a control invariant set for system (3.8) subject to the constraints $u \in U, x \in X$, if for all $x \in \Omega$ there exists an admissible input u such that $f(x,u) \in \Omega$.

To compute a control invariant set we can use a one-step operator such in [6]. The one-step set, given a target set Ω, is the set of states which can be steered in one timestep to the target set by an admissible control action, i.e. $Q(\Omega) = \{x \in \mathbb{R}^n : \exists u \in U \quad | \quad f(x,u) \in \Omega\}$. Given an admissible invariant set Ω the i-step stabilizable set can be obtained by the recursion $V_i = Q(V_{i-1}) \cap X$, where $V_0 = \Omega$. every set V_i is contained in the final set V_∞, the maximal stabilizable set. In order to find this $Q(\Omega)$, [6] developed an algorithm, called *one-step set approximation algorithm*.

This is an interval algorithm, basically a branch and bound algorithm where the ranges of functions are bounded by using interval arithmetic. It computes an inner approximation of the one-step set $Q(\Omega)$. Here we describe the algorithm trying to give some insights into its working principles rather than aiming at an exhaustive mathematical description. This algorithm iteratively process a list of regions L, which initially contains just the region (X,U). Until this list is not empty, it takes from L the region $Z_i = (X_i,U_i)$, where $X_i \subset X$ and $U_i \subset U$, as the box with the maximum diameter in L, and computes a control action u_i. To obtain a feasible u_i it is possible to formulate a linear program such that $f(X_i,u_i) \cap \Omega \neq \emptyset$. Once the u_i control is obtained, three action are possible on Z_i.

- If the interval extension $\square f(X_i,U_i) \cap \Omega = \emptyset$ then the box is rejected.
- If the interval extension $\square f(X_i,U_i) \in \Omega$ then the box X_i is saved in a $B(\Omega)$ list and all boxes $Z_j = (X_j,U_j) \subseteq L$ such that $X_j = X_i$ are removed from the list.
- If the interval extension $\square f(X_i,U_i) \cap \Omega \neq \emptyset$ the width of Z_i is checked and if it is under a certain predefined ε, Z_i is discarded. If its width is bigger than ε then the box is split in two boxes Z_1, Z_2 such that $Z_i = Z_1 \cup Z_2$, and Z_1, Z_2 are added to the list L. The bisection is performed on the component with maximum value.

The region $B(\Omega)$ obtained by the algorithm is a union of boxes, therefore it is a non-convex set in general. In order to use this region, the paper proposes then a second algorithm that builds an inner convex polytope contained inside such area.

This algorithm computes in a iterative way an approximation of a one-step set. To do so, it needs the interval evaluation of the discrete-time dynamics function $f(\cdot,\cdot)$, which gives an outer approximation of the dynamics evaluated on the set:

$$f(X,u) \subseteq f\square(X,U) \tag{3.10}$$

Limitations

This algorithm has two major drawbacks:

- *Flat* Ω set: The main goal is to calculate the one-step control invariant set. In our problem the set that we want to reach is the set $\Omega(q,\dot{q}) = \{q,\dot{q} : q^{min} \leq q \leq q^{max}, \quad \dot{q} = 0\}$. Dealing with a set Ω with a dimension of size zero the *Bravo-2005* algorithm would fail to compute one-step set. This is because the outer approximation of the dynamics would never be included in such Ω.

- One-step set: the algorithm computes the one-step set, but we aim to find the infinite-step set Ω_∞. It is possible to compute the infinite-step set by using the one-step algorithm iteratively, but this is not efficient.

However, the algorithm has one great advantage:

- Scalability: the method used for investigate the X_i state produce a conservative set, but potentially also it makes easy to manipulate system with many constraints.

3.3 Proposed solution

3.3.1 Notation

Before to start to discuss the proposed method it is necessary to recap the notation that we are going to use. The algorithm requires the characteristic of our robotic system such as:

- n : number of *DOF* of the system

- q : vector of the joints position of the robotic system. In general $q \in [q^{min}, q^{max}], q^{min}, q^{max} \in \mathbb{R}^n$

- $\dot{q}$: vector of the joints velocity of the robotic system. In general $\dot{q} \in [\dot{q}^{min}, \dot{q}^{max}], \dot{q}^{min}, \dot{q}^{max} \in \mathbb{R}^n$

- X : the feasible state space of the robotic system with $X \subset \mathbb{R}^n \times \mathbb{R}^n$.

$$X \triangleq \{(q,\dot{q}) : q^{min} \leq q \leq q^{max}, \dot{q}^{min} \leq \dot{q} \leq \dot{q}^{max}\}$$

  A sub-region of this space is referred to as X_i

- x : it denotes a pair $(q,\dot{q}) \in X$

- τ : vector of the joints torque of the robotic system. In general $\tau \in [\tau^{min}, \tau^{max}], \tau^{min}, \tau^{max} \in \mathbb{R}^n$

- U : the input range for our problem.

$$U \triangleq \{\tau : \tau^{min} \leq \tau \leq \tau^{max}\}$$

  A more general and widespread notation for a subset of such input is u_i, as used in the description of 3.2.2. Considering that we assume to deal specifically with robotic systems, we prefer the use of τ.

- $f(x,\tau)$: the dynamics of our robotic system, depending on both x, τ. For our case

$$f(x,\tau) \triangleq \begin{bmatrix} q + \delta t\dot{q} + \frac{\delta t^2}{2} M^{-1}(\tau - h) \\ \dot{q} + \delta t M^{-1}(\tau - h) \end{bmatrix}$$

  As stated in 3.2.2, its interval extension is referred as $\Box f(X_i, \tau)$

- A : interval of feasible joint accelerations. The values are obtained by the means of $ABA - INNER(X,\tau)$. A subset of this element is defined as A_i.

- $ABA - INNER(X,\tau)$: The *Articulated Body Algorithm* (*ABA*) can be used to compute $\ddot{q}$ from $q, \dot{q}, \tau$.

$$\ddot{q} = M(q)^{-1}(\tau - h(q,\dot{q})) = ABA(q,\dot{q},\tau)$$

  Applying the interval extension of the *ABA* on a certain X_i permits to compute an interval of acceleration for that region. The extremes of this interval are the *inner* and *outer* approximations of the acceleration. The *inner* (i.e. feasible at every point of X_i and conservative) acceleration and deceleration are stored in A.

$$A_i \leftarrow ABA - INNER(X_i,\tau)$$

  More detailed explanation follows in 3.3.2.

- $\mathscr{V}$: viability kernel of the robotic system. Note that $\mathscr{V} \subset X$. The viability kernel of a X_i subset is defined as $\mathscr{V}_{X_i}$. The viability kernel is the control invariant set that we are looking for. A subset of this complete viability kernel is defined as V_i.

- $\mathscr{V}^{min}$: conservative approximation of the viable area of the robotic system. Note that $\mathscr{V}^{min} \subset \mathscr{V}$. A subset of this area is referred as $\mathscr{V}_i^{min}$. This viable area is computed by means of $\ddot{q}^m$ from the $\Box\, ABA(X,\tau)$.

Let us define some user-defined parameters:

- trh_{pos}, trh_{vel} thresholds on the minimum sizes of the box. If one of the dimension is under such thresholds the box is discarded

The algorithm makes use of lists and we use this general rules to define and access elements in those list

- *Access of a position in the list:* using a notation inspired by the list management in Python, given a list L, we denote with $L[i]$ the i-th element of the L list.
- *Access of an attribute of an object* using a notation inspired by the object oriented programming paradigm, we use the . to denote the access to an attribute of an object. For example, $L[i].q_{min}$ represents the q_{min} attribute of the object stored in the list L at position i.

The algorithm makes use of the following lists:

- L : list of the initial state-space areas to be processed.
- R : list of the viable subsets coming from the analyzed X_i areas. This list results in an inner approximation of the viable region of X, so $R \subset \mathscr{V}$
- $L1$: list of the areas generated by the algorithm where

$$\begin{aligned} L1[i].q_{min} &= R[i].q_{min}, \\ L1[i].q_{max} &= R[i].q_{max}, \\ L1[i].\dot{q}_{min} &= R[i].q_{max}, \\ L1[i].\dot{q}_{max} &= X.\dot{q}_{max} \end{aligned}$$

  (e.g. see Fig. 3.2 to see the first area that populate L1, *Area 1*). The elements in the list are of type X_i

- $L2$: list of the areas generated by the algorithm where

$$\begin{aligned} L2[i].q_{min} &= R[i].q_{max}, \\ L2[i].q_{max} &= X.q_{max}, \\ L2[i].\dot{q}_{min} &= R[i].q_{min}, \\ L2[i].\dot{q}_{max} &= X.\dot{q}_{max} \end{aligned}$$

  (e.g. see Fig. 3.2 to see the first area that populate L2, *Area 2*). The elements in the list are of type X_i

The Bravo-2005 algorithm can be extended to work with a flat target set. Requiring that $f\Box(X,\tau) \subseteq \Omega$ for τ is sufficient but not necessary to conclude that Ω is reachable from X. Therefore it is possible to replace this condition with a less restrictive one. Given a state region X_i, it is possible to compute a set of accelerations $A_i = [\ddot{q}_i^m, \ddot{q}_i^M]$ that are feasible at all states in X_i. We can then find a subset of X_i from which each joint can come to a stop (i.e. reach Ω) without leaving X_i and using accelerations within the computed bounds.

3.3.2 Inner approximation of forward dynamics with interval arithmetic

In this section we explain how to compute the above-mentioned acceleration bounds for a given state region X.

$$\tau^{min} \leq M(q)\ddot{q} + C(q,\dot{q}) + g(q) \leq \tau^{max} \tag{3.11}$$

The joint accelerations $\ddot{q}$ can then be computed as:

$$\ddot{q} = M(q)^{-1}(\tau - h(q,\dot{q})) = ABA(q,\dot{q},\tau)$$

The *Articulated Body Algorithm* (*ABA*) can be used to compute directly $\ddot{q}$ from $q,\dot{q},\tau$ without explicitly computing $M(q)$ and $h(q,\dot{q})$. Given a symmetric range for $\ddot{q} \in [-\alpha a, \alpha a]$, where $\alpha \in \mathbb{R}$ and $a \in \mathbb{R}^n$, it is possible to verify whether the torque limits would be satisfied for all value of $q,\dot{q},\ddot{q}$ in the above mentioned ranges by checking these inequalities

$$\begin{aligned} \bar{M}(q)a\alpha + \bar{C}(q,\dot{q}) + \bar{g}(q) &\leq \tau^{max} \\ -\bar{M}(q)a\alpha + \underline{C}(q,\dot{q}) + \underline{g}(q) &\geq \tau^{min} \end{aligned} \tag{3.12}$$

where

- $\bar{M}$ contains the maximum absolute value of each element of M for $q \in [q^{min}, q^{max}]$
- $\bar{C}$ contains the maximum value of each element of $C(q,\dot{q})$
- $\underline{C}$ contains the minimum value of each element of $C(q,\dot{q})$
- $\bar{g}$ contains the maximum value of each element of $g(q,\dot{q})$
- $\underline{g}$ contains the maximum value of each element of $g(q,\dot{q})$

These values can be computed by mean of intervals arithmetic. Once the terms $\bar{M},\bar{C},\bar{g},\underline{C},\underline{g}$ have been computed, and a has been defined by the user, it is easy to compute the maximum α that satisfies these inequalities. This method can in general deal with *multi-DOF* system, even though in the following we will test it only for a 1-*DOF* system.

3.3.3 Case of study: 1-*DOF* simple pendulum

The system that we are going to deal with in the test section is a simple 1-*DOF* pendulum. The simple pendulum dynamic equation does not depend on the angular velocity of the pendulum, i.e. the pendulum dynamics only depends on its angular position. From (3.11) it is possible to obtain:

$$\tau^{min} \leq M(q)\ddot{q} + g(q) \leq \tau^{max} \tag{3.13}$$

Considering

- m the mass of the pendulum
- l the length of the pendulum

$$\tau^{min} \leq ml^2\ddot{q} + mlg\sin(q) \leq \tau^{max} \tag{3.14}$$

From (3.14) we can see that $M(q) = ml^2$, and only $g(q) = mlg\sin(q)$ depends on position. Applying the interval arithmetic on (3.14) it is possible to notice that the only term where this operator is effective is the $\sin()$ function.

3.3.4 Intuitive idea of the algorithm

For the simple pendulum, the computed acceleration set A consists in the two extreme values of the available acceleration available at every point in the feasible state space X. Given this acceleration bound it is possible to apply the method developed in the first part of the manuscript and check, for each joint, whether it is possible to come to a stop with such acceleration without leaving X. Considering the maximum deceleration available at every location of X at a certain joint as $-\ddot{q}^{min}$ then the corresponding trajectory is

$$q(t) = q_0 + t\dot{q}_0 - \frac{t^2}{2}\ddot{q}^{min} \quad \dot{q}(t) = \dot{q}_0 - t\ddot{q}^{min}$$

and in a same way as in 1.2.1 defining t_0 as the time where the system reaches $\dot{q} = 0$:

$$q(t_0) = q_0 + \frac{\dot{q}_0^2}{\ddot{q}^{min}} \leq \bar{q}$$

These computations allow us to determine a subset of X from which the system can stop, while remaining in X. Starting at q^{min} and integrating backwards in time with $-\ddot{q}^m$ until one of the bounds of X is reached, it is possible to define a viable area inside X. From the previous section it is known that starting from such area guarantees to possibility to remain in X.

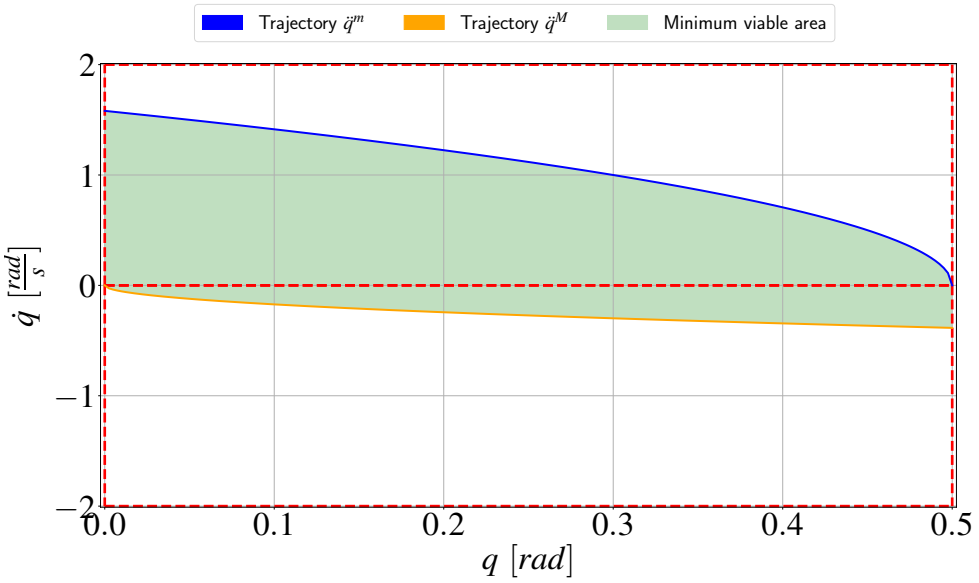


Figure 3.2: The figure shows the space of the allowable trajectories generated by the extreme accelerations obtained from the intervals arithmetic for the simple pendulum system. The numerical data are $q = [0.0, 0.5] rad$, $\dot{q} = [-2.0, 2.0] rad/s$, $m = 1.5 Kg$, $l = 1.5 m$, $\tau = [-15, 15] Nm$. The outer trajectories (blue and orange) represent the trajectories with the smaller absolute value for both acceleration and deceleration (i.e. that the system can always provide in every position of the state space and so more conservative). The area enclosed by the two outer trajectories is the *conservative* viable set $\mathscr{V}^{min}$. The complete $\mathscr{V}$ includes this area.

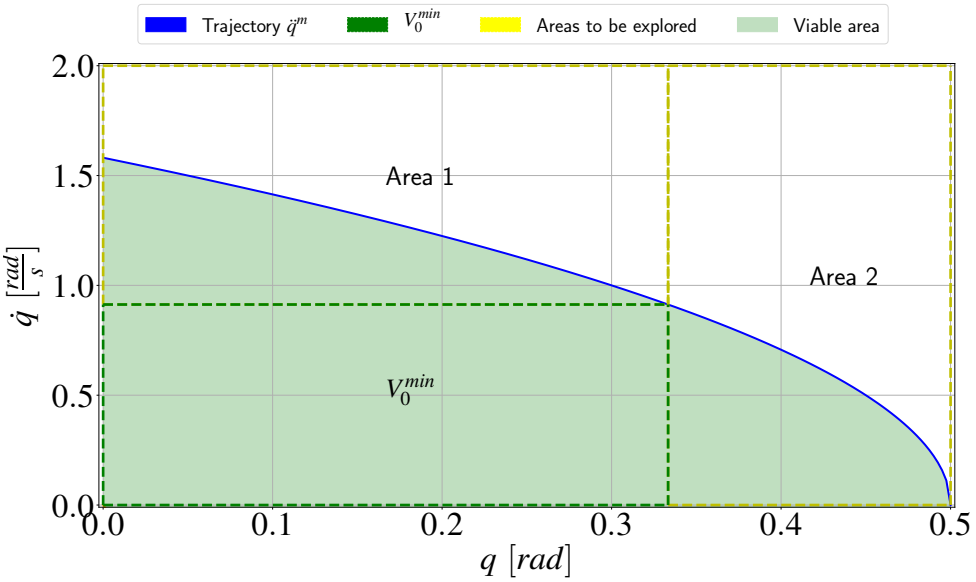


Figure 3.3: The V_0 box represents the first box generated by the method. This box is a polytopic subset $V_0 \in \mathscr{V}^{min}$ and the wider polytope inscribed in the minimum viable area. The two yellow areas represent the area to be explored in the next steps by the algorithm.

However, this viable area is *nonlinear*, which makes it hard to manipulate. Therefore, we compute the largest box that is a subset of this nonlinear area (e.g., see Fig. 3.3), and we store this as a viable region.

Once this viable box is computed, the remaining space can be split into 2 boxes (for the 1-DOF system), which can be analyzed iteratively with the same technique. The generated two boxes have a main difference: while for *Area 2* the *target* velocity to reach is $\dot{q} = 0$ before reaching the upper position bound, for the *Area 1* instead the *target* velocity correspond to the upper velocity bound of the previous found area. The concept of using as a target a previous found area helps to overcome the *one-set step* problem described in 3.2.2 This process can be repeated until the regions are smaller than a given user-defined threshold, resulting in a list of viable regions, in the form of boxes.

The algorithm reorder the areas to be explored in order to analyze in first place the areas with a greater upper position bound. When a new region is analyzed the algorithm always check if there is a already computed viable region that has as lower position bound, the upper position bound of the current analyzed region and a upper velocity bound greater than the lower velocity bound of the current analyzed region.

Before starting to describe more in depth the code implementation of algo-

rithm, some properties and assumption must be introduced and discussed:

- In general retrieving $A_i \leftarrow \square\, ABA(X_i, \tau)$ of robotic system can lead to a void range, for example in the case that the robotic system can not compensate for its own weight for some positions and velocities of its feasible space. Trying to keep the problem simple in this first approach we assume that the robot can provide at least a torque able to compensate for its own weight on all the feasible X region.
- a subset $V_i \subset \mathcal{V}_X$ of the viable area $\mathcal{V}_X \in X$ is not control invariant in general, i.e. it may not be possible to remain in V_i, but only to remain in X. This is of course problematic for applications of the computed set as terminal set for Model Predictive Control, but since this is a preliminary study we postpone the investigation of this problem to future work.
- a subsets $V_i \subset \mathcal{V}_X$ can be used as a target set for a generic X_j under testing. If from a generic region under testing we can reach an already-known as viable subset V_i the region analyzed is a viable subset itself. Using as a target subsets V_i allows to *inflate* the viable region in a very efficient way and helps to overcome the *one-set step* problem described.

3.3.5 Algorithm description

The algorithm is divided in two main parts. The first part computes the acceleration bounds for a given area in $L, L1, L2$ and adds the resulting viable subarea to R. The other part has the purpose to populate the lists $L, L1, L2$ with the areas that have to be analyzed until reaching an exit condition when the size of the areas are below a user-defined threshold.

Algorithm 5 MaximumAccelerations

Require: $X_i, \square\, ABA - INNER(X_i, \tau), \tau$
$(a_{min}, a_{max}) \leftarrow \square\, ABA - INNER(X_i, \tau)$
return $\{\, a_{min}, a_{max} \,\}$

Notice that $ABA - INNER(X_i, \tau)$ return two accelerations (a_{min}, a_{max}). For sake of clarity, we use only the a_{min} (i.e. deceleration) term to show how the algorithm works.

Notice also that Alg. 5, Alg. 6, and Alg. 7 are defined to work with X_i areas. $L, L1, L2$ are lists that contain X_i-type areas. In the definition of the inputs of those algorithms we prefer to use such notation to reinforce the concept of which type of inputs those functions expect, and in addition because those functions do not modify *directly* the $L, L1, L2$ lists.

Algorithm 6 MaximizeArea

Require: X_i, a_{min}
$area \leftarrow (q - X_i.q_{min}) \times (\sqrt{2a_{min}(X_i.q_{max} - q) + X_i.\dot{q}_{min}})$
$q \leftarrow maximize(area).q$
$\dot{q} \leftarrow \sqrt{2a_{min}(X_i.q_{max} - q) + X_i.\dot{q}_{min}}$
return $\{ q, \dot{q} \}$

Algorithm 7 CreateNewAreas

Require: X_i, q, dq
$area1 \leftarrow [X_i.q^{min}, q, \dot{q}, X_i.\dot{q}^{max}]$
$area2 \leftarrow [q, X_i.q^{max}, X_i.\dot{q}^{min}, X_i.\dot{q}^{max}]$
return $\{ area1, area2 \}$

Algorithm 8 CheckTreshold

Require: $L1, L2, R, trh_{pos}, trh_{vel}$
if $L1[-1].q_{max} - L1[-1].q_{min} \leq trh_{pos}$ *or* $L1[-1].\dot{q}_{max} - L1[-1].\dot{q}_{min} \leq trh_{vel}$ **then**
$L1[-1].remove()$
$L2[-1].remove()$
$R[-1].remove()$
if $L2[-1].q_{max} - L2[-1].q_{min} \leq trh_{pos}$ *or* $L2[-1].\dot{q}_{max} - L2[-1].\dot{q}_{min} \leq trh_{vel}$ **then**
$L1[-1].remove()$
$L2[-1].remove()$
$R[-1].remove()$
return

Algorithm 9 ModifyL1Area

Require: $L1, R, X$
$L1[-1] \leftarrow [L1[0].q_{min}, R[-1].q_{max}, R[-1].\dot{q}_{max}, X.\dot{q}_{max}]$
return

Algorithm 10 Extended Bravo's algorithm

$L \leftarrow X$
$L1 \leftarrow []$
$L2 \leftarrow []$
Require: L, τ
$(a_{min}, a_{max}) \leftarrow MaximumAccelerations(L[0], ABA - INNER(X_i, \tau), \tau)$
$(q, \dot{q}) \leftarrow MaximizeArea(L[0], a_{min})$
$(newarea1, newarea2) \leftarrow CreateNewAreas(L[0], q, \dot{q})$
$L1.append(newarea1)$
$L2.append(newarea2)$
$R.append([L[0].q_{min}, q, L[0].\dot{q}_{min}, \dot{q}])$
while $L1 \neq []$ **do**
  while $L2 \neq []$ **do**
    $(a_{min}, a_{max}) \leftarrow MaximumAccelerations(L2[0], ABA - INNER(X_i, \tau), \tau)$
    $(q, \dot{q}) \leftarrow MinimizeArea(L2[0], a_{min})$
    $(newarea1, newarea2) \leftarrow CreateNewAreas(L1[0], \tau)$
    $L1.append(newarea1)$
    $L2.append(newarea2)$
    $R.append([q, L2[0].q_{min}, L2[0].\dot{q}_{min}, \dot{q}])$
    $CheckTreshold(L1, L2)$
    $L2.remove(L2[0])$
  if $R[-1].\dot{q}_{max} \leq L1[0].\dot{q}_{min}$ **then**
    $ModifyL1Area(L1, R, X)$
  $(a_{min}, a_{max}) \leftarrow MaximumAccelerations(L1[0], ABA - INNER(X_i, \tau), \tau)$
  $(q, \dot{q}) \leftarrow MinimizeArea(L1[0], a_{min})$
  $(newarea1, newarea2) \leftarrow CreateNewAreas(L1[0], q, \dot{q})$
  $L1.append(newarea1)$
  $L2.append(newarea2)$
  $R.append([L1[0].q_{min}, q, L1[0].\dot{q}_{min}, dq])$
  $CheckTreshold(L1, L2)$
  $L1.remove(L1[0])$

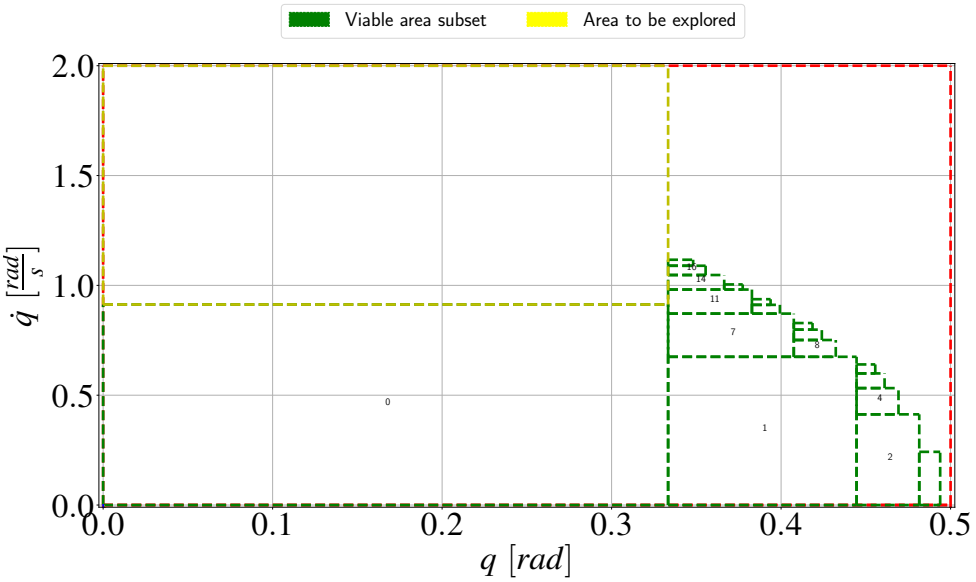


Figure 3.4: This figure shows the condition in which the area to be explored has a lower velocity bound respect to already approved contiguous areas.

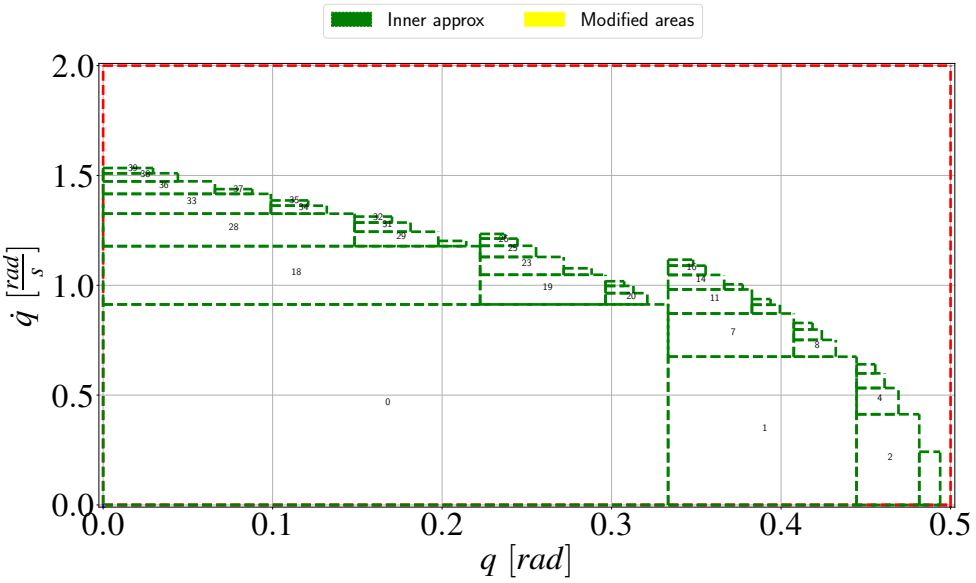


Figure 3.5: Discontinuities when the Alg. 9 is not in use.

Instead Alg. 8 and Alg. 9 are related to the list management and so they work directly on the list. Thus, we prefer to define its inputs as the lists used in the main algorithm (i.e. Alg. 10).

The operation performed by Alg. 9 helps the algorithm to obtain a *smoother*

contour. Once the algorithm have discarded all the $L2$ area it is possible that a R_i has a greater upper velocity bound respect to the next area to be analyzed in $L1$ (e.g. See Fig. 3.4). If we continue to analyze the remaining area without considering this fact we will end up with a discontinuity in that location (e.g. See Fig. 3.5). This modification means to not analyze with the algorithm specific areas (See Fig. 3.5). It is necessary to check that for those areas the acceleration available would lead the system in a contiguous viable area. we can use 5 to check such condition. Graphically the area checked and approved with this method are labeled with an *_ext* suffix. Build a new $L1$-type area to be analyzed in such manner also helps to overcome excessive conservative behavior. 8 discards both X_i, X_j area created by a $L1, L2$ area if one of them is under the thresholds. 9 analyzing again the upper velocity bound of those $L1, L2$ area with different condition can restore a acceptable X_i, X_j previously discarded.

3.4 Tests

The system under test is a single joint pendulum system, as discussed in 3.3.3. The data for such system are

- $m = 1.5[Kg]$
- $l = 1.5[m]$
- $\tau = [-15.0, 15.0][Nm]$
- $[q^{min}, q^{max}] = [0.0, 0.5][rad]$
- $[\dot{q}^{min}, \dot{q}^{max}] = [0.0, 2.0][rad/s]$

The tests are performed considering

- **Torque limits:** the values chosen for the torque limit guarantee that the system is able to compensate for the gravity effect at every feasible position.
- **Velocity bounds:** the upper and lower $\dot{q}$ limit is here intentionally chosen with a non-constraining behaviour for the system in order to have a more representative picture of how the algorithm deals with the viability region as generated only by the dynamic limits.
- **Thresholds:** the thresholds values are chosen as a fraction of the corresponding state-space dimension range.

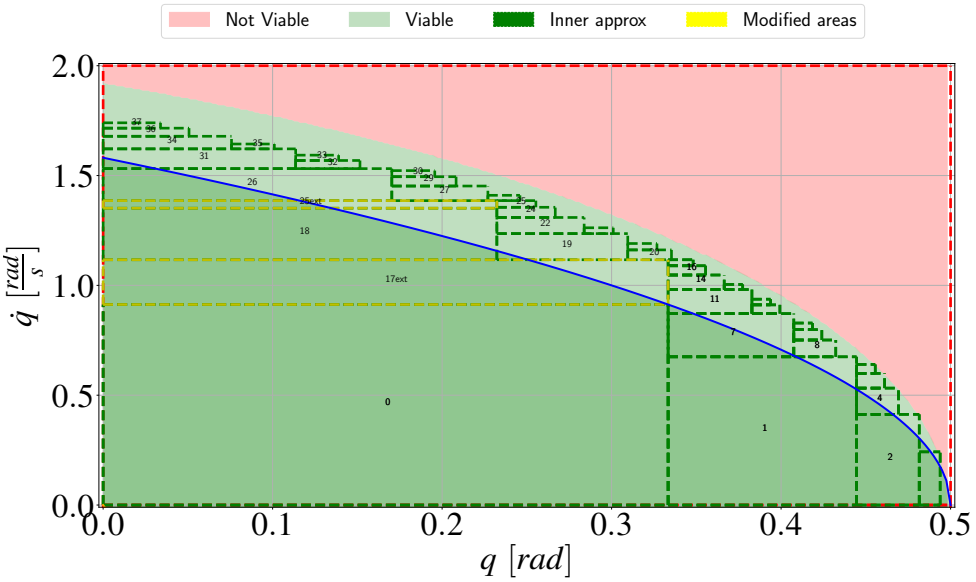


Figure 3.6: The result obtained from the algorithm for a $X : 0 \leq q \leq 0.5 \wedge 0 \leq \dot{q} \leq 2$. The region underlying to the blue trajectory represents the most conservative viable region $\mathscr{V}_{q_{min}} \in \mathscr{V}$. The area labelled as 0 is the first region generated by the algorithm (i.e. the first area that populates R). It is the polytopic subset $\mathscr{V}_i^{min}$ with the maximum area such that $\mathscr{V}_i^{min} \in \mathscr{V}_{q_{min}}$. The light green and red zones correspond to, respectively, the points from which, applying the maximum negative torque, the system is able to stop before the q^{max} constrain. Those regions are obtained by a brute force time demanding calculation on a fine mesh representing the X region. Their goal is to show a good visual approximation of the viable region for the system. The area enclosed by all the green and yellow boxes is the computed inner approximation of the viable region. The labels inside those areas show in which order the areas were generated by the algorithm.

- **Ground-truth:** in order to check if the resulting area is correct is necessary to compute the *ground-truth* of the viable area in X. To do so we perform a brute force computation with a very fine mesh that represent X verifying which points are viable (e.g., see Fig. 3.6).

From those test is possible to notice

- **Area order:** in Fig. 3.4, Fig. 3.7, Fig. 3.8, Fig. 3.9 it possible to notice in which order the algorithm create the new areas of R, moving at first towards the upper position limit and then proceeding backwards near the $\mathscr{V}$ region calculated with the brute force computation.

- **Threshold dependencies:** Fig. 3.7, Fig. 3.8, Fig. 3.9 show how the algo-

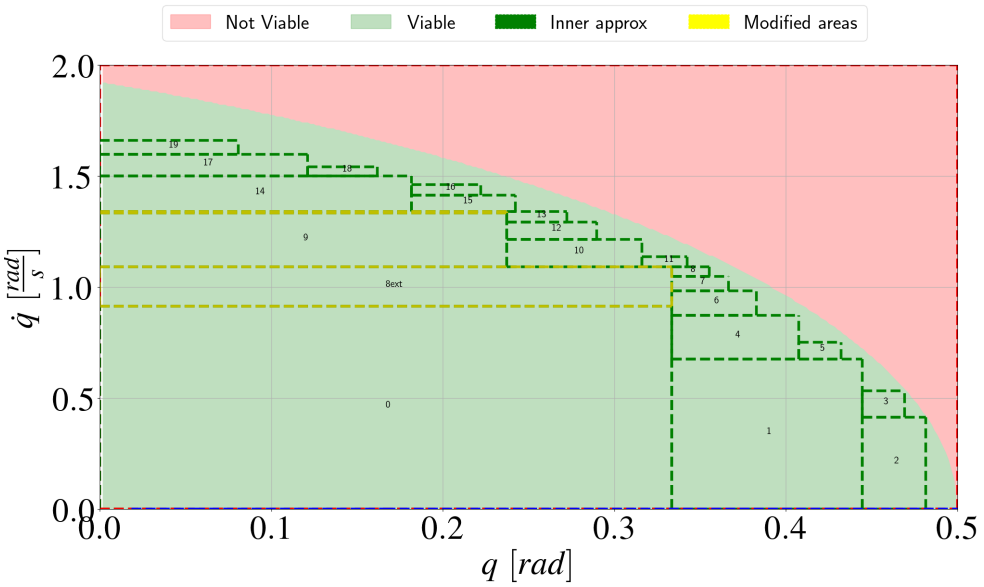


Figure 3.7: Viability region approximated with a threshold of $\frac{1}{50}$ and generating 22 areas.

rithm improves the computed area with various value of thresholds. It is possible to see a clear improvement diminishing the threshold value. This threshold influence *accuracy drift* that we are going to discuss later.

- $\mathcal{V}^{min}$ **area:** Fig. 3.6 and Fig. 3.12 underline also the difference between $\mathcal{V}^{min}$ and $\mathcal{V}$.

3.5 Discussion

The algorithm has some drawbacks.

- **Accuracy drift:** It is possible to notice a drift in the accuracy of the algorithm. The motivation relies on how the algorithm works. In first place it explores areas in the neighborhoods of the upper position limit and lower velocity limit. The areas generated here as an approximation of the viable region are conservative, so an inner approximation of the effective real region. As mentioned in 3.3.4, these areas are then used as a target for the computation of the remaining regions. The intrinsic conservative behaviour of the method makes the error difference grow while processing the areas, leading to a larger error in the last areas explored by the algorithm i.e. the

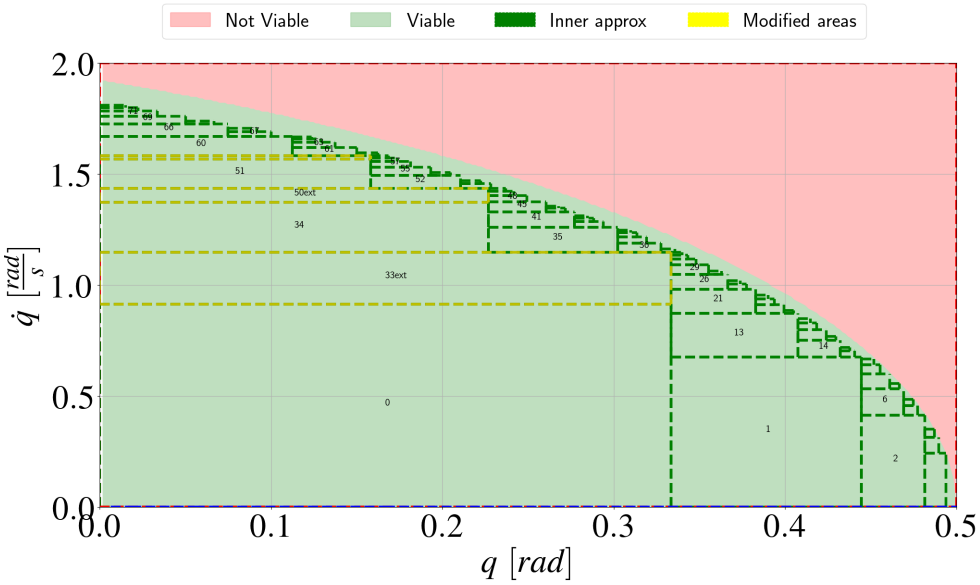


Figure 3.8: Viability region approximated with a threshold of $\frac{1}{200}$ and generating 77 areas.

regions in the neighborhood of the upper velocity limit. In Fig. 3.7, Fig. 3.8, Fig. 3.9 it is possible to notice this phenomenon. Imposing a more restrictive exit condition for the area, so a lower threshold, make possible to obtain a better approximation, but the problem still exists. With a threshold of $\frac{1}{200}$ of the position range ($q^{max} - q^{min}$) can obtain in the worst point a value of 1.81 rad/s instead of the ground truth value of 1.92 rad/s computed with a brute force approach. This means a 5.7% underestimation error in the worst point of the feasible state (e.g., see Fig. 3.11). With a threshold of $\frac{1}{2000}$ of the position range ($q^{max} - q^{min}$) can obtain in the worst point a value of 1.88 rad/s instead of the ground truth value of 1.92 rad/s computed with a brute force approach. This means a 2.1% underestimation error in the worst point of the feasible state (e.g., see Fig. 3.11).

- **Non-convex set:** the list of boxes R obtained so far is a non-convex set. Consequently, while it is guaranteed that the system remains always in X, it is not possible to ensure that the system will remain in R starting at every point of R. To ensure such condition it is necessary to build a polytopic inner approximation of the R area. It leads to a even more conservative region.

- **Scalability:** the algorithm can easily deals with the computation of avail-

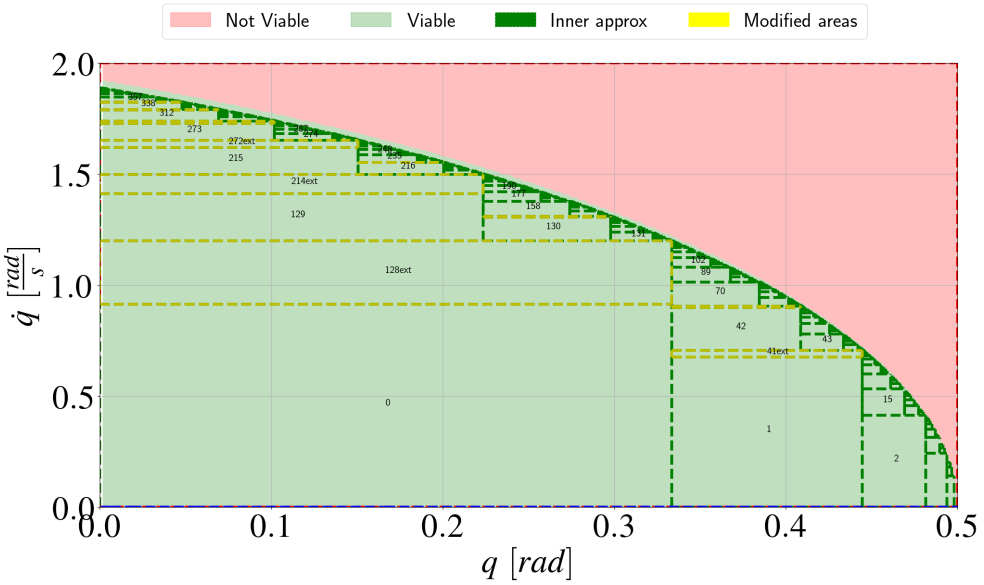


Figure 3.9: Viability region approximated with a threshold of $\frac{1}{2000}$ and generating 402 areas.

able acceleration for a generic $X \in \mathbb{R}^n$ space. The exploration of the X state for the utilized *single-DOF* case results in the exploration of a $X \in \mathbb{R}^2$ space. Even if the complexity increase linearly, which it is fair enough, it means to deal with a $X \in \mathbb{R}^4$ hyper-space for a *2-DOF* case such as the double pendulum. The choice of how to explore such hyper-space has to be pondered, especially considering the *accuracy drift* phenomenon that affect the last analyzed areas.

- **Control invariant set:** the main problem of the algorithm is that the list of boxes R obtained so far is an inner approximation of the viability kernel for such nonlinear system. The viability kernel is a control invariant set, but its inner approximation R it is not. Our initial goal was to find a control invariant set because it allows us to use the previous result discussed in the first section of the manuscript. As for the *non-convex set* discussion we can guarantee that, starting from R the system would stop in X, but not while remaining in R. This is of course problematic for applications of the computed set as terminal set for Model Predictive Control.

It is important to notice that, even if *non-convex set* and *control invariant set* seems similar and related, the concept is different. A control invariant set can be a non-

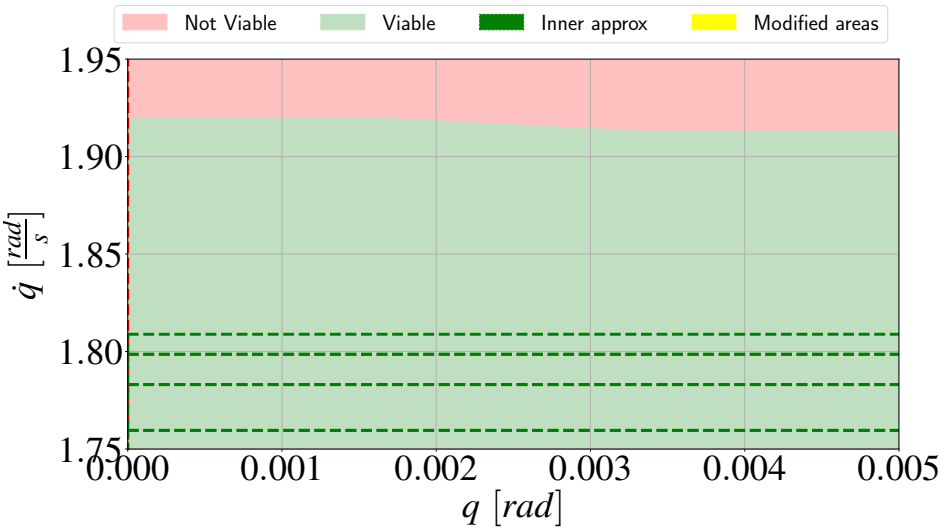


Figure 3.10: Underestimation error in $q = 0$ with a threshold= 200. The error between the real and computed value is $= 5.7\%$.

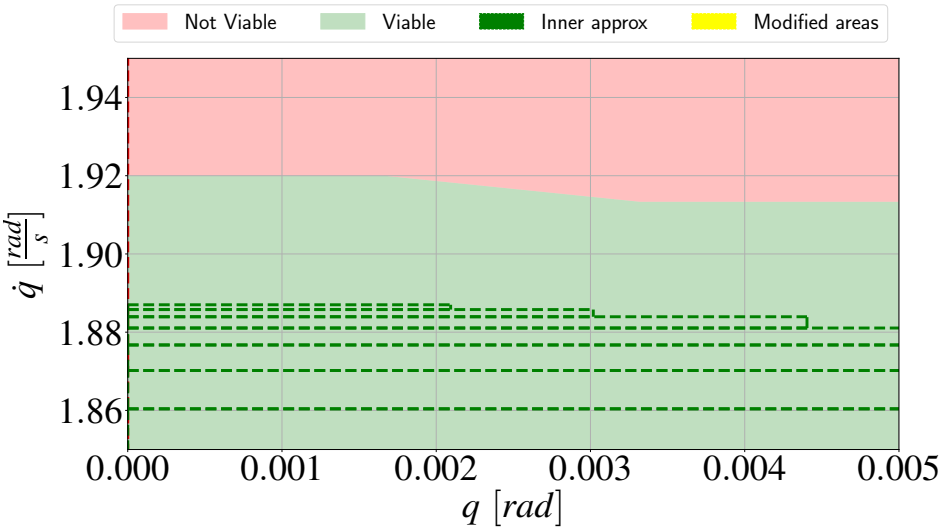


Figure 3.11: Underestimation error in $q = 0$ with a threshold= 2000. The error between the real and computed value is $= 2.1\%$.

convex set. The only condition that it must ensure is that starting from a point in such set there would be a feasible trajectory that allows the system to remain in such set. The non-convex set problem, even if is not a trivial problem, can be solved with a method similar to the one developed in [6], but the resulting convex inner polytope is still *not* a control invariant set.

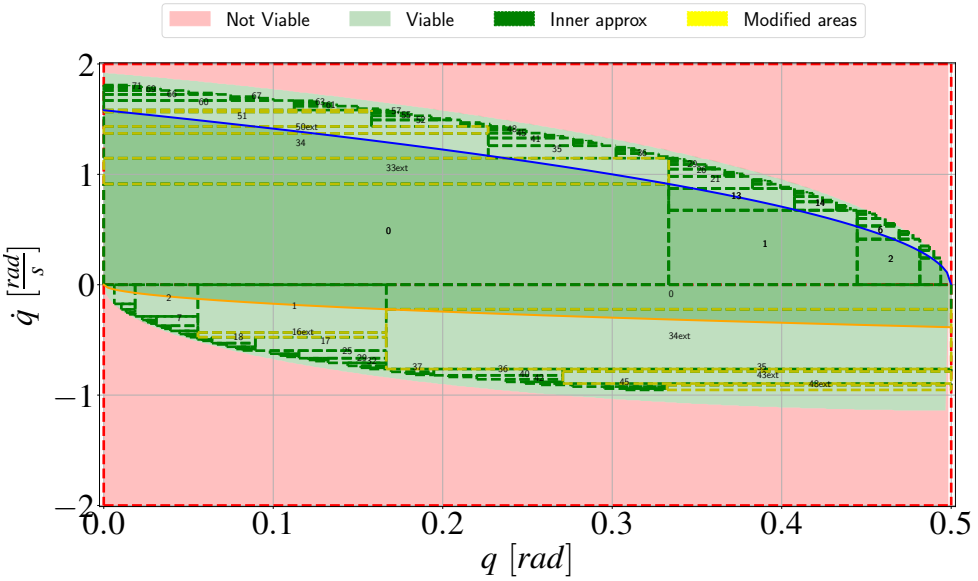


Figure 3.12: Test with an $X : 0 \leq q \leq 0.5 \wedge -2 \leq \dot{q} \leq 2$. It is possible to notice the *accuracy drift* phenomenon in both the positive and negative limit of the viability area computed.

Chapter 4

Conclusions and Future Work

This thesis has focused on the control of robot manipulators subject to different kinds of joint constraints. In particular, in the first part the focus has been on joint position, velocity and acceleration bounds. In the second part instead we have moved our attention from the joint acceleration bounds to the joint torque bounds, which are a better approximation of the real physical limitations of the system.

The problem with joint position, velocity and acceleration bounds had already been tackled in the literature [12], but without considering the presence of disturbances. This was a severe limitation because all physical systems are subject to some degree of uncertainty coming from, for instance, modeling errors, sensor noise and communication delays. Therefore, we have developed a new approach that can guarantee the satisfaction of the constraints despite the presence of bounded additive disturbances on the joint accelerations. The results obtained in simulation on a single joint and on the 6-degree-of-freedom Baxter robot arm show better performance with respect to other state-of-the-art methods, with our method being the only one capable of ensuring constraint satisfaction. Moreover, the presented approach has similar computational complexity with respect to [12], making it easily usable for real-world applications.

The second part of the work tackled a much more challenging problem because the introduction of joint torque bounds makes the problem nonlinear. In this preliminary investigation we have developed a method to compute an inner approximation of the viability kernel for such systems. The method is an extension of [6] that relies on an iterative algorithm to discretize the state space in small regions, and then uses interval arithmetic to conservatively assess viability in each region. We have shown that, for a single-joint system, our approach is able to compute an arbitrarily accurate inner approximation of the viability kernel.

4.1 Future work

The work presented in this thesis has concretely contributed to the problem of controlling constrained robot manipulators. Nonetheless, this challenging problem is still far from being completely solved.

Regarding the problem with joint acceleration bounds, even though the developed approach is exact (i.e. tight) for the case of bounded additive disturbances, one could prefer to model disturbances using random variables [21]. The *stochastic* approach to uncertainty modeling could indeed result in a less conservative behavior of the system, and thus improve performance. This is because stochastic approaches account for the fact that small errors are in general more likely than large errors, which is completely neglected by the robust approach. Therefore, solving the presented problem for the case of stochastic uncertainties is an interesting direction for future work.

Regarding the problem with joint torque limits, much work is still necessary before the presented approach can be used in real-world applications. In its current form, the main weakness of our approach is that the computed inner approximation of the viability kernel is *not* control invariant. In other words, we can guarantee that the system could avoid violating the constraints starting from such set, but it might need to leave the set to do so. This makes it hard to use this set in a control algorithm (e.g., Model Predictive Control), which requires the property of control invariance in order to guarantee recursive feasibility [33].

Another concern with the presented approach is its scalability, which is a common limitation of algorithms computing viability kernel approximations for nonlinear systems [6, 28]. While some parts of the algorithm could easily scale to higher dimensions (e.g., the computation of the acceleration bounds using interval arithmetic), other parts seem to present more challenges, such as the order in which different state-space regions should be analyzed. These aspects are definitely worth further investigation.

4.2 Implementation

The Python implementation of the algorithms presented in this thesis can be found at:
https://github.com/ErikZan/Robust-Joints-bounds-guarantee

Bibliography

[1] Eugene Asarin, Thao Dang, and Antoine Girard. “Hybridization methods for the analysis of nonlinear systems”. In: *Acta Informatica* 43.7 (2007), pp. 451–476.

[2] Eugene Asarin, Thao Dang, and Antoine Girard. “Reachability Analysis of Nonlinear Systems Using Conservative Approximation”. In: *Hybrid Systems: Computation and Control*. Ed. by Oded Maler and Amir Pnueli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 20–35. ISBN: 978-3-540-36580-8.

[3] Eugene Asarin et al. “Approximate Reachability Analysis of Piecewise-Linear Dynamical Systems”. In: *Hybrid Systems: Computation and Control*. Ed. by Nancy Lynch and Bruce H. Krogh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 20–31. ISBN: 978-3-540-46430-3.

[4] JP Aubin. “Viability Theory Birkhäuser”. In: *Boston, Basel, Berlin* (1991).

[5] F. Blanchini. “Set invariance in control”. In: *Automatica* 35.11 (1999), pp. 1747–1767. ISSN: 0005-1098. DOI: https://doi.org/10.1016/S0005-1098(99)00113-2. URL: https://www.sciencedirect.com/science/article/pii/S0005109899001132.

[6] J. M. Bravo et al. “On the computation of invariant sets for constrained nonlinear systems: An interval arithmetic approach”. In: *2003 European Control Conference (ECC)*. 2003, pp. 288–293. DOI: 10.23919/ECC.2003.7084969.

[7] M. Cannon, V. Deshmukh, and B. Kouvaritakis. “Nonlinear model predictive control with polytopic invariant sets”. In: *Automatica* 39.8 (2003), pp. 1487–1494. ISSN: 0005-1098. DOI: https://doi.org/10.1016/S0005-1098(03)00128-6. URL: https://www.sciencedirect.com/science/article/pii/S0005109803001286.

[8] Mark Cannon, Basil Kouvaritakis, and Venkatesh Deshmukh. "Enlargement of polytopic terminal region in NMPC by interpolation and partial invariance". In: *Automatica* 40.2 (2004), pp. 311–317. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2003.10.004. URL: https://www.sciencedirect.com/science/article/pii/S0005109803003285.

[9] Y. Chen et al. "Data-Driven Computation of Minimal Robust Control Invariant Set". In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018, pp. 4052–4058. DOI: 10.1109/CDC.2018.8619312.

[10] Benjamin Decardi-Nelson and Jinfeng Liu. *Computing robust control invariant sets of constrained nonlinear systems: A graph algorithm approach*. 2020. arXiv: 2009.13581 [eess.SY].

[11] Wilm Decré et al. "Extending iTaSC to support inequality constraints and non-instantaneous task specification". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2009.

[12] A. Del Prete. "Joint Position and Velocity Bounds in Discrete-Time Acceleration/Torque Control of Robot Manipulators". In: *IEEE Robotics and Automation Letters* 3.1 (Jan. 2018), pp. 281–288. ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2738321.

[13] Andrea Del Prete et al. "Prioritized Motion-Force Control of Constrained Fully-Actuated Robots: "Task Space Inverse Dynamics"". In: *Robotics and Autonomous Systems* 63 (2015), pp. 150–157. arXiv: 1410.3863.

[14] M. Fiacchini, T. Alamo, and E.F. Camacho. "On the computation of convex robust control invariant sets for nonlinear systems". In: *Automatica* 46.8 (2010), pp. 1334–1338. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2010.05.007. URL: https://www.sciencedirect.com/science/article/pii/S000510981000213X.

[15] Mirko Fiacchini and Mazen Alamir. *Computing control invariant sets is easy*. 2017. arXiv: 1708.04797 [cs.SY].

[16] E. G. Gilbert and K. T. Tan. "Linear systems with state and control constraints: the theory and application of maximal output admissible sets". In: *IEEE Transactions on Automatic Control* 36.9 (1991), pp. 1008–1020. DOI: 10.1109/9.83532.

[17] Fumio Kanehiro et al. "Integrating geometric constraints into reactive leg motion generation". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2010.

[18] Shahab Kaynama et al. “Computing the viability kernel using maximal reachable sets”. In: *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*. 2012, pp. 55–64.

[19] E. C. Kerrigan and J. M. Maciejowski. “Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control”. In: *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*. Vol. 5. 2000, 4951–4956 vol.5. DOI: 10.1109/CDC.2001.914717.

[20] Ki Cheol Park, Pyung Hun Chang, and Seung Ho Kim. “The enhanced compact QP method for redundant manipulators using practical inequality constraints”. In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*. Vol. 1. 1998, 107–114 vol.1. DOI: 10.1109/ROBOT.1998.676327.

[21] E. Kofman, J. A. De Doná, and M. M. Seron. “Probabilistic ultimate bounds and invariant sets for LTI systems with Gaussian disturbances”. In: *2011 Australian Control Conference*. 2011, pp. 537–542.

[22] Twan Koolen et al. “Summary of Team IHMC ’ s Virtual Robotics Challenge Entry”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2013.

[23] Trosten Kröger and Friedrich M Wahl. “On-Line trajectory generation in robotic systems. Basic concepts for instananeous reactions to unforseen (sensor) events”. In: *IEEE Transaction on Robotics* 26.1 (2010), pp. 94–111.

[24] Scott Kuindersma et al. “Optimization-based Locomotion Planning , Estimation , and Control Design for the Atlas Humanoid Robot”. In: *Autonomous Robots (accepted pending minor revision)* (2015).

[25] John N. Maidens et al. “Lagrangian methods for approximating the viability kernel in high-dimensional systems”. In: *Automatica* 49.7 (2013), pp. 2017–2029. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2013.03.020. URL: https://www.sciencedirect.com/science/article/pii/S0005109813001891.

[26] Oded Maler. “Computing reachable sets: An introduction”. In: ().

[27] D.Q. Mayne. “Control of Constrained Dynamic Systems”. In: *European Journal of Control* 7.2 (2001), pp. 87–99. ISSN: 0947-3580. DOI: https://doi.org/10.3166/ejc.7.87-99. URL: https://www.sciencedirect.com/science/article/pii/S0947358001711417.

[28] Ian M. Mitchell and Claire J. Tomlin. “Overapproximating Reachable Sets by Hamilton-Jacobi Projections”. In: *Journal of Scientific Computing* 19.1-3 (2003), pp. 323–346. ISSN: 08857474. DOI: 10.1023/A:1025364227563.

[29] Quan Nguyen and Koushil Sreenath. “Exponential Control Barrier Functions for enforcing high relative-degree safety-critical constraints”. In: *American Control Conference*. 2016.

[30] Ki Cheol Park, P. H. Chang, and Seung Ho Kim. “The Enhanced Compact QP Mathod for Redundant Manipulators Using Practical Inequality Constraints”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 1998.

[31] S. V. Rakovic et al. “Simple Robust Control Invariant Tubes for Some Classes of Nonlinear Discrete Time Systems”. In: *Proceedings of the 45th IEEE Conference on Decision and Control*. 2006, pp. 6397–6402. DOI: 10.1109/CDC.2006.377551.

[32] Manuel Rauscher, Melanie Kimmel, and Sandra Hirche. “Constrained Robot Control Using Control Barrier Functions”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016.

[33] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*. Vol. 2. Nob Hill Publishing Madison, WI, 2017.

[34] Sébastien Rubrecht et al. “Constraints Compliant Control : constraints compatibility and the displaced configuration approach”. In: *IEEE/RSJ Intenational Conference on Intelligent Robots and Systems (IROS)*. 2010.

[35] Sébastien Rubrecht et al. “Motion safety and constraints compatibility for multibody robots”. In: *Autonomous Robots* 32.3 (2012), pp. 333–349.

[36] M. Rungger and P. Tabuada. “Computing Robust Controlled Invariant Sets of Linear Systems”. In: *IEEE Transactions on Automatic Control* 62.7 (2017), pp. 3665–3670. DOI: 10.1109/TAC.2017.2672859.

[37] Layale Saab et al. “Dynamic Whole-Body Motion Generation under Rigid Contacts and other Unilateral Constraints”. In: *IEEE Transactions on Robotics* 29.2 (2013), pp. 346–362.

[38] Patrick Saint-Pierre. “Approximation of the viability kernel”. In: *Applied Mathematics and Optimization* 29.2 (1994), pp. 187–209.

[39] Joris Vaillant and Abderrahmane Kheddar. “Vertical Ladder Climbing by HRP-2 Humanoid Robot”. In: *2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2014.

[40] Patrick M. Wensing and David E. Orin. "Generation of Dynamic Humanoid Behaviors Through Task-Space Control with Conic Optimization". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013.

[41] Shuyou Yu et al. "Tube MPC scheme based on robust control invariant set with application to Lipschitz nonlinear systems". In: *Systems Control Letters* 62.2 (2013), pp. 194–200. ISSN: 0167-6911. DOI: https://doi.org/10.1016/j.sysconle.2012.11.004. URL: http://www.sciencedirect.com/science/article/pii/S0167691112002289.