Joint Position and Velocity Bounds in Discrete-Time Acceleration/Torque Control of Robot Manipulators

Andrea del Prete

Joint Position and Velocity Bounds in Discrete-Time Acceleration/Torque Control of Robot Manipulators

Andrea Del Prete, *Member, IEEE*

***Abstract*—This paper deals with the problem of controlling a robotic system whose joints have bounded position, velocity and acceleration/torque. Assuming a discrete-time acceleration control, we compute tight bounds on the current joint accelerations that ensure the existence of a feasible trajectory in the future. Despite the clear practical importance of this issue, no complete and exact solution has been proposed yet, and all existing control architectures rely on hand-tuned heuristics. We also extend this methodology to torque-controlled robots, for which joint accelerations are only indirectly bounded by the torque limits. Numerical simulations are presented to validate the proposed method, which is computationally efficient and hence suitable for high-frequency control.**

***Index Terms*—Viability, Constrained Systems, Optimal Control.**

I. Introduction

One of the main challenges in the control of robotic systems is the fact that they are highly constrained. In particular, this paper focuses on the joint position, velocity and acceleration bounds. First, let us introduce the notation that will be used throughout the paper:

- $t \in \mathbb{R}^+$ denotes time
- $i \in \mathbb{N}$ denotes discrete time steps
- δt is the time-step duration of the discrete-time controller
- $q(t), \dot{q}(t), \ddot{q}(t) \in \mathbb{R}$ are the joint position, velocity and acceleration at time t
- $q_i \triangleq q(i\delta t)$, $\dot{q}_i \triangleq \dot{q}(i\delta t)$, $\ddot{q}_i \triangleq \ddot{q}(i\delta t)$
- q^{min}, q^{max} are the joint position boundaries
- $\dot{q}^{max}, \ddot{q}^{max}$ are the maximum joint velocity and acceleration

Now we can define the above-mentioned bounds as:

$$q^{min} \leq q \leq q^{max}, \quad |\dot{q}| \leq \dot{q}^{max}, \quad |\ddot{q}| \leq \ddot{q}^{max}$$

Assuming that throughout each time step i we apply a constant joint acceleration, we can express the future joint position q_{i+1} and velocity $\dot{q}_{i+1}$ as functions of the current joint acceleration $\ddot{q}_i$:

$$q_{i+1} = q_i + \delta t\, \dot{q}_i + \frac{1}{2}\delta t^2 \ddot{q}_i$$
$$\dot{q}_{i+1} = \dot{q}_i + \delta t\, \ddot{q}_i$$

A naive approach to bound the acceleration is then to compute the maximum and minimum $\ddot{q}_i$ such that q_{i+1} and $\dot{q}_{i+1}$ are within their bounds:

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

People quickly realized that this approach does not work in practice [1] because the resulting acceleration bounds may be incompatible (i.e. upper bound less than lower bound). Since then, several improvements over this basic approach have been proposed, but none of them seems to have completely solved the problem.

A well-known trick [1], [2] is to use a larger value of δt in (1), which results most of the time in stricter bounds. In general this helps reducing the acceleration when getting close to a position/velocity limit, but it does not guarantee compatibility of the resulting constraints.

The authors are with the CNRS, LAAS, 7 avenue du colonel Roche, Univ de Toulouse, LAAS, F-31400 Toulouse, France. e-mail: adelpret@laas.fr.

Manuscript received ...

Another approach [3] is to bound the joint velocity with a (hand-tuned) linear function of the distance of the joint position to the limits. 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.

Other inverse-dynamics control frameworks [4], [5] simply do not model joint position-velocity bounds in the controller, 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 bounds.

To the best of our knowledge, Decre et al. [6] have been the first ones trying to provide formal guarantees of satisfaction of position, velocity and acceleration bounds. For the case of positive joint velocity, they write the future position trajectory for maximum deceleration and impose the satisfaction of the position upper 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 $(\dot{q}_{i+1} - \dot{q}_i)/\delta t$ (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 was later addressed by Rubrecht et al. [7], [8]. However, in doing so the authors introduced some arbitrary slackness in the solution. Even if they suggest that this slackness may be beneficial in case of measurement errors, we believe that the issue of measurement errors should be tackled explicitly if necessary, and not through some arbitrary slackness. Moreover, also this work deals with pseudo-acceleration bounds.

The two main contributions of the method proposed in this paper to ensure position, velocity and acceleration bounds are:

- the method is exact, meaning that it is not conservative (contrary to [7]) and it works in all situations (contrary to [6]);
- the method assumes constant acceleration between consecutive time steps, which allows bounding the real acceleration—rather than a pseudo-acceleration [6].

Section II formally states the problem and Section III presents an efficient and exact algorithm to solve it. Section IV discusses a way to extend this result to torque-controlled robots, whose joint accelerations are only indirectly bounded by the torque bounds. Section V validates the proposed approach through numerical simulations with a 7-degree-of-freedom manipulator. Finally Section VI draws the conclusions.

II. Problem Statement

Let us consider a robot whose joints have limited position, velocity and acceleration. We define the set of all feasible states $\mathcal{F}$ as:

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

Moreover, let us assume that our control inputs are the joint accelerations, which are also bounded: $|\ddot{q}| \leq \ddot{q}^{max}$. At each time step, we would like to know what are the maximum and minimum joint accelerations that we can apply, such that position and velocity limits can still be satisfied in the future. This can be formulated as

an infinite-horizon optimal control problem having as variables the sequence of accelerations from time step 0 to infinite:

$$\begin{aligned} \ddot{q}_0^{max} = \underset{\ddot{q}_0,\ddot{q}_1,\dots}{\text{maximize}} \quad & \ddot{q}_0 \\ \text{subject to} \quad & (q(t),\dot{q}(t)) \in \mathcal{F} \qquad \forall t > 0 \\ & |\ddot{q}_i| \leq \ddot{q}^{max} \qquad \forall i \geq 0 \\ & (q(0),\dot{q}(0)) \quad \text{fixed} \end{aligned} \tag{2}$$

Similarly, we could define $\ddot{q}_0^{min}$ as the solution of the analog problem in which we minimize $\ddot{q}_0$ rather than maximizing it. Since acceleration can change only at discrete time step, the position and velocity between two successive time steps are:

$$\begin{aligned} q(i\delta t + t) &= q_i + t\,\dot{q}_i + \frac{t^2}{2}\ddot{q}_i \qquad \forall i \geq 0, t \in [0, \delta t] \\ \dot{q}(i\delta t + t) &= \dot{q}_i + t\,\ddot{q}_i \qquad \forall i \geq 0, t \in [0, \delta t] \end{aligned}$$

The joint limits have to be satisfied not only at the discrete time steps, but in continuous time, so the problem has an (uncountable) infinite number of constraints. Because of the infinite number of variables and constraints we cannot solve this problem in this form. The main contribution of this paper is an efficient algorithm to compute the exact solution of this problem.

III. Problem Solution

A. Viability

We recall now the concept of viability, which we will exploit to reformulate problem (2) in a simpler form. A state is *viable* [9] if starting from that state there exists a sequence of control inputs that allows for the satisfaction of all the constraints in the future. Formally, we can define the viability kernel $\mathcal{V}$ gathering all viable states as:

$$\begin{aligned} (q(0),\dot{q}(0)) \in \mathcal{V} \quad \Leftrightarrow \quad \exists (\ddot{q}_i)_{i=0}^{\infty} : (q(t),\dot{q}(t)) \in \mathcal{F} \quad \forall t \geq 0, \\ |\ddot{q}_i| \leq \ddot{q}^{max} \quad \forall i \geq 0 \end{aligned}$$

In the field of obstacle avoidance, non-viable states are typically referred to as "inevitable collision states" [10]. The interest of introducing the viability kernel $\mathcal{V}$ 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 $\mathcal{V}$. However, this definition of $\mathcal{V}$ is basically useless in practice: 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 will derive an equivalent definition of $\mathcal{V}$ that allows us to check membership easily. Thanks to this, we can reformulate the (hard) problem of satisfying the position-velocity-acceleration limits as the (simple) problem of ensuring that the next state is viable.

1) Continuous-Time Control: Let us start by assuming that we can change $\ddot{q}$ at any instant. This results in a set of viable states $\mathcal{V}^c$ which is a superset of $\mathcal{V}$, that is $\mathcal{V} \subseteq \mathcal{V}^c$.

It is obvious that $\mathcal{V}^c$ is a subset of $\mathcal{F}$ because all viable states are also feasible, but clearly not all feasible states are viable. The reason is that if q is moving towards one of its boundaries with a large velocity, then the acceleration capabilities of the system may not be sufficient to avoid reaching the joint limit with nonzero velocity. For a given initial position q, we can find the maximum initial velocity $\dot{q}_M^{\mathcal{V}}$ that allows us to satisfy the position limits in the future:

$$\begin{aligned} \dot{q}_M^{\mathcal{V}} = \underset{\dot{q},\ddot{q}(t)}{\text{maximize}} \quad & \dot{q} \\ \text{subject to} \quad & q(0) = q, \quad \dot{q}(0) = \dot{q} \\ & (q(t),\dot{q}(t)) \in \mathcal{F} \qquad \forall t \geq 0 \\ & |\ddot{q}(t)| \leq \ddot{q}^{max} \qquad \forall t \geq 0 \end{aligned}$$

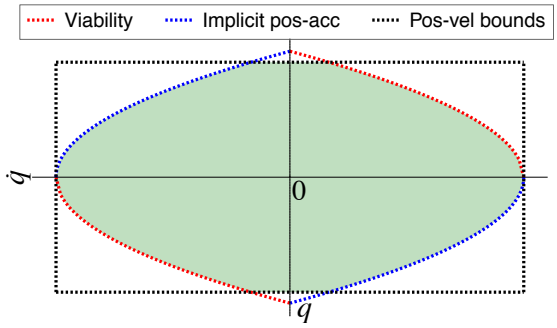


Fig. 1. State-space plot showing the viability constraints, the position and velocity bounds and the constraints implicitly imposed by the position and acceleration bounds for $q^{max} = -q^{min} = 1$ rad, $\dot{q}^{max} = 5$ rad/s, $\ddot{q}^{max} = 15$ rad/s^2.

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. Then to solve the problem we can:

1) write the position trajectory for constant acceleration $\ddot{q}(t) = -\ddot{q}^{max}$,
2) compute the time at which the velocity of this trajectory is zero $t^0 = \dot{q}/\ddot{q}^{max}$,
3) compute the initial velocity such that $q(t^0) = q^{max}$

Following these steps we find:

$$\dot{q}_M^{\mathcal{V}}(q) = \sqrt{2\ddot{q}^{max}(q^{max} - q)}$$

Similarly, the minimum initial velocity to ensure viability is such that, by constantly applying the maximum acceleration, we end up exactly at the lower position limit q^{min} with zero velocity, which leads us to:

$$\dot{q}_m^{\mathcal{V}}(q) = -\sqrt{2\ddot{q}^{max}(q - q^{min})}$$

We can conclude that the set of viable states (for continuous-time control) is:

$$\mathcal{V}^c = \{(q,\dot{q}) : (q,\dot{q}) \in \mathcal{F},\ \dot{q}_m^{\mathcal{V}}(q) \leq \dot{q} \leq \dot{q}_M^{\mathcal{V}}(q)\}$$

This definition of $\mathcal{V}^c$ allows us to check easily the viability of a state by just verifying three inequalities. Fig. 1 shows how $\mathcal{V}^c$ and $\mathcal{F}$ may look like in a state-space plot. Moreover, it shows also the state constraints implicitly imposed by the position-acceleration bounds, e.g. starting from q^{min} with zero velocity and applying maximum acceleration we find the maximum velocity we can reach for each state.

2) Discrete-Time Control: If the control can only change at discrete time steps, we have to pay attention to what happens after we reached the position boundary (e.g. q^{max}) with zero velocity. In case of continuous-time control we can switch immediately to $\ddot{q} = 0$ after reaching q^{max}, so we can be sure that no constraint violation will occur. In case of discrete-time control instead we need to maintain $\ddot{q} = -\ddot{q}^{max}$ until the end of the time step in which we reach q^{max}. In theory this may lead us to violating the lower position or velocity bound. However, in practice, this is extremely rare, as we argue in the following.

Let us assume the worst case, namely we reach the state $(q^{max}, 0)$ while applying maximum deceleration a moment after the beginning of the time step. In this case we need to maintain $\ddot{q} = -\ddot{q}^{max}$ for the remaining of the time step (which is approximately δt). At the end of the time step the state will then be $q = q^{max} - 0.5\delta t^2 \ddot{q}^{max}$, $\dot{q} = -\delta t\,\ddot{q}^{max} \triangleq \dot{q}^{discr}$. At that point we can bring the joint to a stop by applying $\ddot{q} = \ddot{q}^{max}$ for one time step, which would lead us to

$q = q^{max} - \delta t^2 \ddot{q}^{max} \triangleq q^{discr}$. In case q^{discr} and $\dot{q}^{discr}$ are within their bounds, then we have nothing to worry about and $\mathcal{V} = \mathcal{V}^c$. The conditions to verify are then:

$$q^{max} - q^{min} \geq \delta t^2 \ddot{q}^{max}, \qquad \dot{q}^{max} \geq \delta t\, \ddot{q}^{max} \tag{3}$$

Typically these conditions are verified because δt is much smaller than 1. For instance, even assuming a value of δt as large as 0.1 s, to violate (3) we would need either $\dot{q}^{max}$ to be more than 10 times smaller than $\ddot{q}^{max}$, or $(q^{max} - q^{min})$ to be at least 100 times smaller than $\ddot{q}^{max}$. Even if this is the case, we can safely set $\ddot{q}^{max}$ to be smaller than its real value so as to satisfy (3).

When (3) is not verified the definition of $\mathcal{V}$ is much more complex than that of $\mathcal{V}^c$. To avoid this (practically useless) mathematical complexity, in the remaining of the paper we will assume that (3) is verified, and so $\mathcal{V} = \mathcal{V}^c$.

B. Reformulating (2) in Terms of Viability

Now that we derived a simple definition of $\mathcal{V}$, we can exploit it to reformulate problem (2). Given the current state $(q, \dot{q}) \in \mathcal{V}$, we need to find the maximum (or minimum) value of $\ddot{q}$ such that: i) the next state $(q(\delta t), \dot{q}(\delta t))$ belongs to $\mathcal{V}$, and ii) the whole trajectory leading to the next state belongs to $\mathcal{F}$. Note that the first condition alone is not sufficient: the trajectory between two viable states may violate a constraint. Viability only ensures the existence of a feasible future trajectory, meaning that, even starting from a viable state, you can end up violating a constraint. In mathematical terms we reformulate (2) as:

$$\begin{aligned} \ddot{q}_0^{max} = \underset{\ddot{q}}{\text{maximize}} \quad & \ddot{q} \\ \text{subject to} \quad & q(0) = q, \quad \dot{q}(0) = \dot{q} \\ & (q(t), \dot{q}(t)) \in \mathcal{F} \qquad \forall t \in [0, \delta t] \\ & \dot{q}_m^{\mathcal{V}}(q(\delta t)) \leq \dot{q}(\delta t) \leq \dot{q}_M^{\mathcal{V}}(q(\delta t)) \\ & |\ddot{q}| \leq \ddot{q}^{max} \end{aligned} \tag{4}$$

This problem is much simpler than its original version because: i) it has a single variable (rather than an infinite sequence of variables), and ii) its constraints concern only the trajectory in $[0, \delta t]$ (rather than in $[0, \infty]$). However, even problem (4) is hard to solve because its constraints are i) infinitely many, and ii) nonlinear. In the next three subsections we reformulate the inequality constraints of problem (4); each one will give us a lower and an upper bound on $\ddot{q}$, which we will combine in Section III-F.

C. Position Inequalities

The position trajectory is a second-order polynomial, which we want to remain bounded within the position limits:

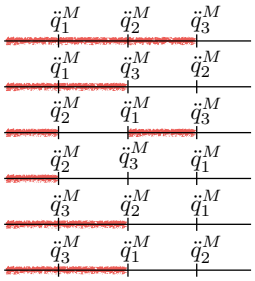
$$q^{min} \leq q + t\dot{q} + 0.5t^2\ddot{q} \leq q^{max} \qquad \forall t \in [0, \delta t] \tag{5}$$

The classical approach in the literature is to ensure that these inequalities are satisfied only for $t = \delta t$. If the velocity does not change sign in the current time step, this is actually enough to guarantee the satisfaction of these constraints over the whole interval $[0, \delta t]$. However, in case of a change in the sign of the velocity, the constraints may be satisfied for $t = \delta t$, but violated for some value of t between 0 and δt. First let us focus on the upper bound. To reduce the infinite number of constraints to a finite number we can rewrite the upper-bound constraint in (5) as:

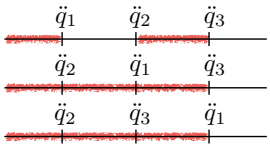
$$f(\ddot{q}) \leq q^{max}, \tag{6}$$

where:

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



(a) The six possible orderings of $\ddot{q}_1^M, \ddot{q}_2^M$ and $\ddot{q}_3^M$. The region in red represents the feasible acceleration region according to (8).



(b) The three possible orderings of $\ddot{q}_1, \ddot{q}_2$ and $\ddot{q}_3$. The region in red represents the feasible acceleration region according to (11).

To find the time at which the position trajectory reaches its extremum we set its derivative to zero:

$$\dot{q} + t\ddot{q} = 0 \quad \rightarrow \quad t^{ext} \triangleq -\frac{\dot{q}}{\ddot{q}}$$

If $\ddot{q} < 0$ then the extremum is a maximum, and if $t^{ext} \in [0, \delta t]$ then the extremum position is reached in the interior of the time step. These two conditions are equivalent to:

$$\dot{q} > 0, \qquad \ddot{q} \leq -\frac{\dot{q}}{\delta t} \triangleq \ddot{q}_1^M \tag{7}$$

If these conditions are satisfied the maximum position is:

$$f(\ddot{q}) = q + t^{max}\dot{q} + 0.5(t^{max})^2\ddot{q} = q - \frac{\dot{q}^2}{2\ddot{q}}$$

So the constraint (6) becomes:

$$\ddot{q} \leq \frac{-\dot{q}^2}{2(q^{max} - q)} \triangleq \ddot{q}_2^M$$

If instead the conditions (7) are not satisfied then the maximum position is reached at the boundaries of the time step, so we just need to ensure that the constraint is satisfied for $t = \delta t$, which gives us:

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

Now let us try to 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 (7) is satisfied then the upper bound is $\ddot{q}_2^M$, otherwise the upper bound is $\ddot{q}_3^M$. We may then rewrite (6) as:

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

or equivalently:

$$\ddot{q} \leq \min(\ddot{q}_1^M, \ddot{q}_2^M) \quad \vee \quad \ddot{q}_1^M < \ddot{q} \leq \ddot{q}_3^M \tag{8}$$

Depending on the specific values of $\ddot{q}_1^M, \ddot{q}_2^M$ and $\ddot{q}_3^M$, one among them is the *real* acceleration upper bound. Fig. 2a depicts the six possible scenarios. The third case is clearly impossible because it results in a discontinuous feasible region, so we can neglect it. When $\ddot{q}_3^M > \ddot{q}_1^M$ (i.e. the first two cases) then $\ddot{q}_3^M$ is the upper bound. In the other three cases the upper bound is the minimum between $\ddot{q}_1^M$ and $\ddot{q}_2^M$.

Repeating the same analysis for the lower bound and putting it all together we get the computation summarized by Algorithm 1.

D. Velocity Inequalities

Since the velocity trajectory is a line we just need to ensure that the velocity inequalities are satisfied for $t = \delta t$. This gives us:

$$\frac{1}{\delta t}(-\dot{q}^{max} - \dot{q}) \leq \ddot{q} \leq \frac{1}{\delta t}(\dot{q}^{max} - \dot{q}) \tag{9}$$

Algorithm 1 computeAccBoundsFromPosLimits

Require: $q, \dot{q}, q^{min}, q^{max}, \delta t$
$\ddot{q}_1^M \leftarrow -\dot{q}/\delta t$
$\ddot{q}_3^M \leftarrow 2(q^{max} - q - \delta t\,\dot{q})/(\delta t^2)$
$\ddot{q}_3^m \leftarrow 2(q^{min} - q - \delta t\,\dot{q})/(\delta t^2)$
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
    $\ddot{q}^{UB} \leftarrow \min(\ddot{q}_1^M, -\dot{q}^2/(2(q^{max} - q)))$
  end if
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
    $\ddot{q}^{LB} \leftarrow \max(\ddot{q}_1^M, \dot{q}^2/(2(q - q^{min})))$
  end if
end if
return $\{\ \ddot{q}^{LB}, \ddot{q}^{UB}\ \}$

Algorithm 2 computeAccBoundsFromViability

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

E. Viability Inequalities

Let us consider the upper bound of the viability inequality:

$$\begin{aligned} \dot{q}(\delta t) &\leq \sqrt{2\ddot{q}^{max}(q^{max} - q(\delta t))} \\ \dot{q} + \delta t\ddot{q} &\leq \sqrt{2\ddot{q}^{max}(q^{max} - q - \delta t\,\dot{q} - 0.5\delta t^2\ddot{q})} \end{aligned} \tag{10}$$

This constraint is clearly nonlinear in $\ddot{q}$, but we can derive a simple algorithm to reformulate it as a simple upper bound. Since the right-hand side (RHS) of (10) is always positive, if the left-hand side (LHS) is negative then (10) is satisfied:

$$\dot{q} + \delta t\,\ddot{q} \leq 0 \quad \Leftrightarrow \quad \ddot{q} \leq -\frac{\dot{q}}{\delta t} \triangleq \ddot{q}_1$$

If instead the LHS is positive we can take the square of both sides:

$$\begin{aligned} &(\dot{q} + \delta t\ddot{q})^2 \leq 2\ddot{q}^{max}(q^{max} - q - \delta t\,\dot{q} - 0.5\delta t^2\ddot{q})) \\ &a\ddot{q}^2 + b\ddot{q} + c \leq 0, \end{aligned} \tag{11}$$

where:

$$\begin{aligned} a &= \delta t^2 \\ b &= \delta t\,(2\dot{q} + \delta t\,\ddot{q}^{max}) \\ c &= \dot{q}^2 - 2\ddot{q}^{max}(q^{max} - q - \delta t\,\dot{q}) \end{aligned}$$

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 $\Delta = 0$):

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

Since $a > 0$ by definition, the inequality (11) 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 (11). Putting it all together we can rewrite (10) as:

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

Similarly to what we did for the position inequalities in Section III-C, we can analyse the different scenarios depending on the values of $\ddot{q}_1, \ddot{q}_2$ and $\ddot{q}_3$ (see Fig. 2b). Here we have only three possible scenarios because by definition $\ddot{q}_2 \leq \ddot{q}_3$. The first case is clearly impossible because it results in a discontinuous feasible region. In the other cases the bound is given by the maximum between $\ddot{q}_1$ and $\ddot{q}_3$. Repeating the same analysis for the lower bound, we get the computation summarized by Algorithm 2.

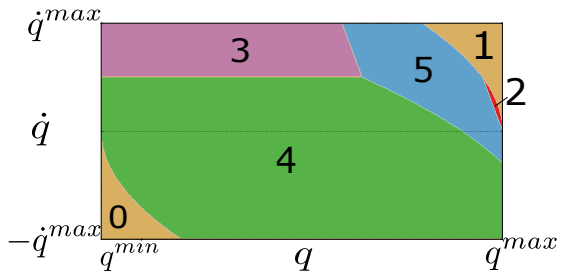


Fig. 2. Feasible state space for $q^{max} = -q^{min} = 0.5$ rad, $\dot{q}^{max} = 2$ rad/s, $\ddot{q}^{max} = 10$ rad/s^2, and $\delta t = 0.1$s. Region 0 represents the space that is not reachable, while region 1 represents the space that is not viable. In each of the other four regions there is a different acceleration upper bound that dominates the others. Region 2 is the one coming from the position inequality (5). Region 3 is the one coming from the velocity inequality (9). Region 4 is due to the acceleration upper bound $\ddot{q}^{max}$. Region 5 is the one coming from the viability inequality (10).

Algorithm 3 Compute Joint Acceleration Bounds

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

F. Final Algorithm to Solve (2)

Finally, to compute the solution of our original problem (2), we just need to compute all the upper and lower bounds of $\ddot{q}$, and then select the minimum among the upper bounds and the maximum among the lower bounds. This procedure is summarized by Algorithm 3.

Fig. 2 shows the feasible state space divided into six different regions. Apart from the non-viable region and the non-reachable region, in each of the other four regions there is a different active constraint in (4). Note that the size of these regions depends on the values of the time step and the position, velocity and acceleration bounds. In some cases certain regions may even disappear.

Algorithm 4 Estimate Acceleration Bounds From Torque Bounds

Require: $\mathbf{q}^{min}, \mathbf{q}^{max}, \dot{\mathbf{q}}^{max}, \tau^{max}, N$
  $\ddot{\mathbf{q}}^{UB} \leftarrow [\infty, \dots, \infty]$
  $\ddot{\mathbf{q}}^{LB} \leftarrow -[\infty, \dots, \infty]$
  for $i = 1 \rightarrow N$ **do**
    $\mathbf{q} \leftarrow \text{random}(\mathbf{q}^{min}, \mathbf{q}^{max})$
    $\dot{\mathbf{q}} \leftarrow \text{random}(-\dot{\mathbf{q}}^{max}, \dot{\mathbf{q}}^{max})$
    $M \leftarrow \text{computeMassMatrix}(\mathbf{q})$
    $\mathbf{h} \leftarrow \text{computeBiasForces}(\mathbf{q}, \dot{\mathbf{q}})$
    for $j = 1 \rightarrow \text{size}(\mathbf{q})$ **do**
      $\ddot{\mathbf{q}}^{max} \leftarrow (\tau^{max}[j] - \mathbf{h}[j])/M[j,j]$
      $\ddot{\mathbf{q}}^{min} \leftarrow (-\tau^{max}[j] - \mathbf{h}[j])/M[j,j]$
      $\ddot{\mathbf{q}}^{UB}[j] \leftarrow \min(\ddot{\mathbf{q}}^{UB}[j], \ddot{\mathbf{q}}^{max})$
      $\ddot{\mathbf{q}}^{LB}[j] \leftarrow \max(\ddot{\mathbf{q}}^{LB}[j], \ddot{\mathbf{q}}^{min})$
    end for
  end for
  return $\{\ddot{q}^{LB}, \ddot{q}^{UB}\}$

IV. Torque Control

During the last 15 years robots are switching from velocity/acceleration control to torque control [11], [12]. This section explains how to apply the methodology proposed in the previous section to a torque-controlled robot. In order to do so we have to face two issues.

First, so far we have assumed to have a constant acceleration during the time step, while now we have a constant torque. Considering an n-degree-of-freedom manipulator, the relationship between torques and accelerations is given by:

$$\tau = M(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}), \tag{12}$$

where $\tau \in \mathbb{R}^n$ are the joint torques, $\ddot{\mathbf{q}} \in \mathbb{R}^n$ are the joint accelerations, $M(\mathbf{q}) \in \mathbb{R}^{n \times n}$ is the inertia matrix, $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^n$ are the bias forces. According to (12) a constant torque does not imply a constant acceleration because $\mathbf{q}$ and $\dot{\mathbf{q}}$ change during the time step. However, since time steps are typically short (i.e. between 1 and 10 ms), $\mathbf{h}$ and M do not change much between the beginning and the end of the time step. Hence we can assume that accelerations are approximately constant throughout the time step.

The second issue is that joint accelerations are not directly bounded, but they are indirectly bounded by the torque limits τ^{max}. This means that the acceleration bounds depend on the system state. We suggest a conservative solution, that is to estimate offline *worst-case* acceleration bounds and use them in our algorithm. By worst case we mean the smallest upper bound and the largest lower bound. Algorithm 4 shows a heuristic procedure to estimate these bounds: i) we generate a large number N of random states, ii) for each state we compute the upper/lower acceleration bound at each joint, assuming that the other joint accelerations are zero, iii) for each joint we select the worst-case bounds. This procedure is not completely conservative because it computes the acceleration bounds for a certain joint assuming that the other joint accelerations are zero—while this is unlikely the case in reality. However, typically only one joint at a time requires maximum acceleration, so this should not be an issue in practice, as we show in our simulations. If needed, a more conservative procedure to estimate worst-case acceleration bounds from torque bounds can be used.

V. Simulations

In this section we validate the proposed method through numerical simulations with the 7-degree-of-freedom arm of the Baxter robot[1]. In both our tests we compare the naive approach to bound accelerations with the approach proposed in this paper. Our first test focuses on joint-space acceleration control, while in our second test the robot joints are torque controlled and the task is executed by the end-effector in Cartesian-space. All the kinematics and dynamics quantities have been computed with the Pinocchio library [14].

[1]http://www.rethinkrobotics.com/baxter/

A. Joint-Space Acceleration Control

In this test we use the following joint-space acceleration controller:

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

The desired joint accelerations are computed as:

$$\ddot{\mathbf{q}}^d = k_p(\mathbf{q}^r - \mathbf{q}) - k_d\dot{\mathbf{q}},$$

where $k_p, k_d \in \mathbb{R}^+$ are the proportional and derivative gains. The values used for the simulations are $k_p = 1000$, $k_d = 2\sqrt{k_p}$, $\delta t = 0.01$ s. We carried out several simulations using different methods to compute the acceleration bounds $\ddot{\mathbf{q}}^{lb}, \ddot{\mathbf{q}}^{ub}$ and verified which ones were successful in preventing violation of the joint limits. To incite the hitting of a joint bound we set the reference position $\mathbf{q}^r$ for the first joint outside its valid range (i.e. 0.1 radians above its upper bound). The results are summarized by Fig. 3.

1) Naive Acceleration Bounds: First we computed the acceleration bounds using the naive approach, that is using (1). Fig. 3a shows the resulting trajectory of the first joint, which violated the upper position bound. Fig. 3b shows that the common trick [1], [2] of using a larger value of δt for computing the acceleration bounds helps mitigating the issue, but it does not completely solve it. The impact velocity is reduced, but the constraint is still violated.

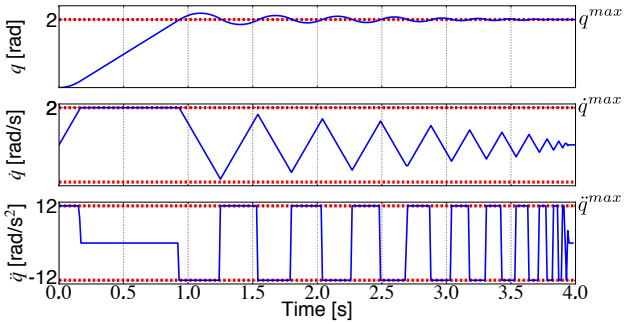
2) Viability-Based Acceleration Bounds: Fig. 3c shows the trajectory obtained using the algorithm proposed in this paper to compute the acceleration bounds. The position and velocity upper bounds are reached multiple times, but never violated. We can notice how the system never reaches the steady state $(q^{max}, 0)$—as it would if the controller were continuous time. The joint velocity oscillates around zero, while its acceleration is discontinuous. Even if unpleasant on a real system, this behavior is *sound*, because we did not limit the jerk of our trajectory. We can easily get rid of these discontinuities by using larger values of δt for computing the acceleration bounds, as shown in Fig. 3d.

B. Cartesian-Space Torque Control

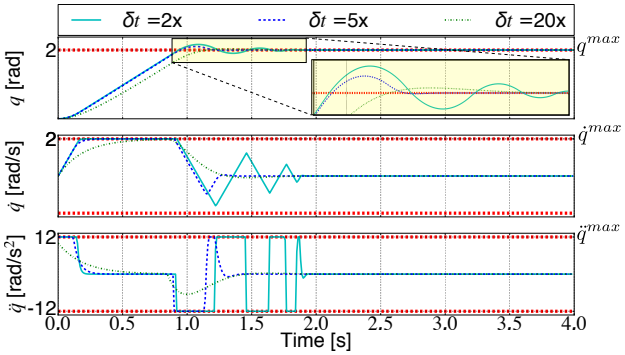
In this simulation we show the importance of correctly bounding the joint accelerations during task-space control. We used a task-space inverse dynamics controller [13] to reach a desired configuration with the end-effector of Baxter's arm (see Fig. 4). At the same time, a lower-priority postural task in joint space is used to resolve the manipulator redundancy. The controller also guarantees the satisfaction of the torque and joint acceleration limits. At each control loop we computed the desired joint torques by solving the following Quadratic Program:

$$\begin{aligned} \underset{\tau, \ddot{\mathbf{q}}}{\text{minimize}} \quad & ||\dot{\mathbf{v}}^d - J(\mathbf{q})\ddot{\mathbf{q}} - \dot{J}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}||^2 + w||\ddot{\mathbf{q}}^p - \ddot{\mathbf{q}}||^2 \\ \text{subject to} \quad & \tau = M(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \\ & \ddot{\mathbf{q}}^{lb} \leq \ddot{\mathbf{q}} \leq \ddot{\mathbf{q}}^{ub} \\ & |\tau| \leq \tau^{max}, \end{aligned}$$

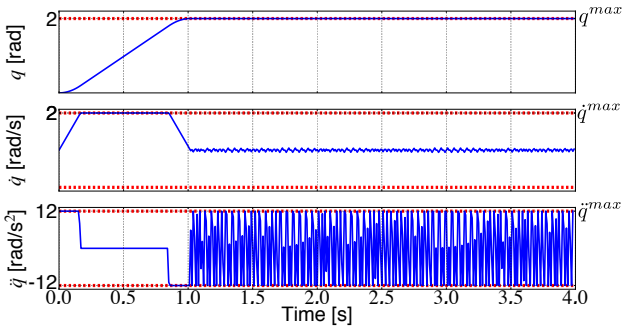
where $J(\mathbf{q}) \in \mathbb{R}^{6 \times n}$ is the end-effector Jacobian matrix, and $w \in \mathbb{R}$ is the weight used to create a non-strict priority of the end-effector



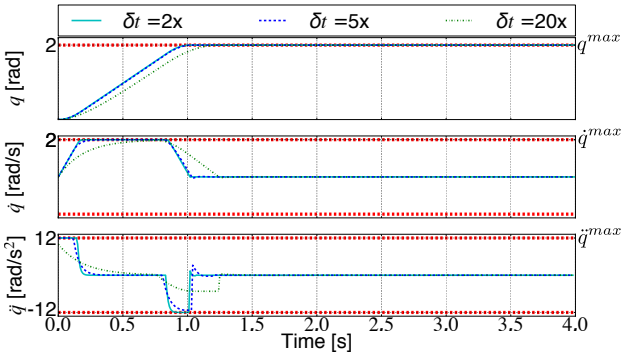
(a) Acceleration bounds computed with the naive approach (1) and the same δt of the controller.



(b) Acceleration bounds computed with the naive approach (1) and larger values of δt than the controller.

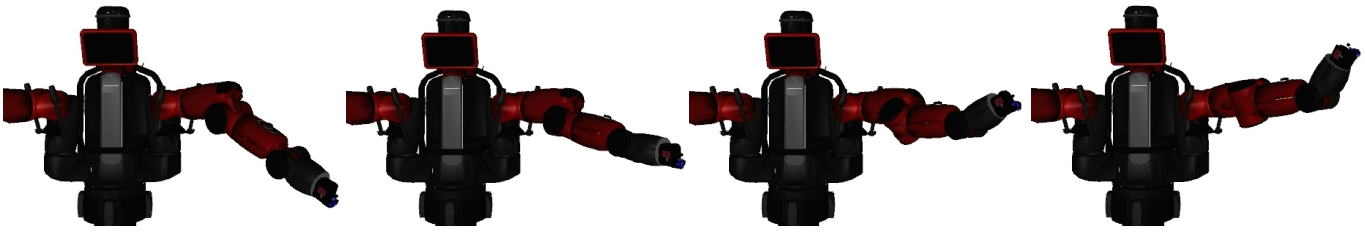


(c) Acceleration bounds computed with the approach proposed in this paper and the same δt of the controller. Even if these acceleration discontinuities may be undesirable on a real system, this behavior is *sound* because we did not limit the jerk of our trajectory. Using a larger value of δt in the acceleration bound computation makes them disappear.



(d) Acceleration bounds computed with the approach proposed in this paper and larger values of δt than the controller.

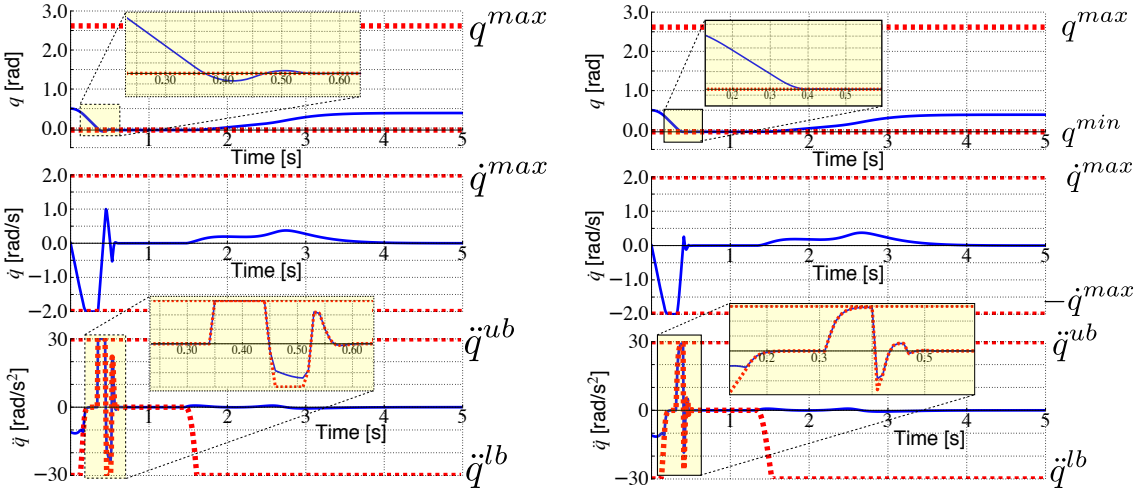
Fig. 3. Trajectories of the first joint of the Baxter arm obtained by using different techniques to compute the acceleration bounds. For this simulation we had $\delta t = 0.01$ s, $q^{max} = 2$ rad, $\dot{q}^{max} = 2$ rad/s, $\ddot{q}^{max} = 12$ rad/s^2.



(a) 0.0 s (b) 0.2 s (c) 0.4 s (d) 0.6 s

(e) 0.8 s (f) 1.0 s (g) 1.2 s (h) 5.0 s

Fig. 4. Screenshots of the simulation with the Baxter robot.



(a) Using the naive approach the joint reaches its lower position bound at 0.36 s with a velocity of -1.4 rad/s.

(b) Using the proposed method position, velocity and acceleration bounds are satisfied throughout the whole trajectory.

Fig. 5. Trajectory of the elbow joint using either the naive approach or our approach to compute the joint acceleration bounds. In both cases we used a larger δt ($2\times$) than the controller to avoid discontinuities.

task over the postural joint task. The desired end-effector acceleration and joint accelerations are computed as:

$$\dot{\mathbf{v}}^d = k_p \log(H(\mathbf{q})^{-1} H^d) - k_d J(\mathbf{q})\dot{\mathbf{q}}$$
$$\ddot{\mathbf{q}}^p = k_p(\mathbf{q}^r - \mathbf{q}) - k_d\dot{\mathbf{q}},$$

where $k_p, k_d \in \mathbb{R}^+$ are the proportional and derivative gains, respectively, and $H, H^d \in SE3$ are respectively the measured and desired rigid transformation from world to end-effector reference frame. The values used for this simulations are $k_p = 10$, $k_d = 2\sqrt{k_p}$, $\delta t = 0.01$ s, $w = 10^{-3}$.

Because of the initial and final pose of the end-effector (see Fig. 4) we can expect the elbow joint to move towards its lower bound. We carried out several simulations using different methods to compute the acceleration bounds $\ddot{\mathbf{q}}^{lb}, \ddot{\mathbf{q}}^{ub}$ and verified which methods were successful in preventing violation of the joint limits. In all cases we used the acceleration bounds that we estimated offline from the torque bounds using Algorithm 4.

1) Naive Acceleration Bounds: In our first simulation we computed $\ddot{\mathbf{q}}^{lb}$ and $\ddot{\mathbf{q}}^{ub}$ using the naive method (1). We computed the acceleration bounds with a value of δt twice as large as the controller to avoid discontinuities in the joint accelerations. The robot managed to reach the desired pose with its end-effector, but the elbow joint reached its lower position bound with a large velocity of 1.4 rad/s (as shown by Fig. 5a). On a real robot this behavior could have resulted in a serious mechanical damage. We also tried using a much larger value of δt ($20\times$). While this helped reducing the impact velocity of the elbow joint to 0.3 rad/s, it did not prevent the impact.

2) Viability-Based Acceleration Bounds: We then computed $\ddot{\mathbf{q}}^{lb}$ and $\ddot{\mathbf{q}}^{ub}$ using the method proposed in this paper (i.e. Algorithm 3). Similarly to the previous test, we used a larger δt ($2\times$) to compute the acceleration bounds. Also in this case the end-effector reached the desired pose, but no constraint was violated: Fig. 5b shows that the elbow joint safely reached its lower bound with zero velocity.

VI. Conclusions

This paper discussed the problem of controlling a robotic system whose joints have limited positions, velocities and accelerations. The difficulty of simultaneously satisfying position, velocity and acceleration bounds is well-known in robotics [1], [3], [6], [8]. However, up to now, no exact algorithm to solve this problem had been proposed. We focused on the computation of the maximum and minimum current joint accelerations that guarantee the existence of a feasible future trajectory. This problem can be formulated as an infinite-horizon optimal control problem, which is computationally intractable. We reformulated it exploiting the concept of viability and we derived a computationally-efficient algorithm to compute its exact solution. We also proposed a pragmatic way to adapt this methodology to torque-controlled robots, whose accelerations are only indirectly bounded by the torque limits. Our simulations on the 7-degree-of-freedom Baxter's arm clearly show the interest of using the proposed approach—rather than the common naive approach—to guarantee the satisfaction of the joint bounds.

References

[1] K. C. Park, P. H. Chang, and S. H. Kim, "The Enhanced Compact QP Mathod for Redundant Manipulators Using Practical Inequality Constraints," in *International Conference on Robotics and Automation*, 1998, pp. 107–114.

[2] L. Saab, O. E. Ramos, N. Mansard, P. Soueres, and J.-y. Fourquet, "Dynamic Whole-Body Motion Generation under Rigid Contacts and other Unilateral Constraints," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 346–362, 2013.

[3] F. Kanehiro, M. Morisawa, W. Suleiman, K. Kaneko, and E. Yoshida, "Integrating geometric constraints into reactive leg motion generation," *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pp. 4069–4076, 2010.

[4] P. M. Wensing and D. E. Orin, "Generation of Dynamic Humanoid Behaviors Through Task-Space Control with Conic Optimization," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, pp. 3088–3094.

[5] T. Koolen, J. Smith, G. Thomas, S. Bertrand, J. Carff, N. Mertins, D. Stephen, P. Abeles, J. Englsberger, S. Mccrory, J. V. Egmond, M. Griffioen, M. Floyd, S. Kobus, N. Manor, S. Alsheikh, D. Duran, L. Bunch, E. Morphis, L. Colasanto, K.-l. H. Hoang, B. Layton, P. Neuhaus, M. Johnson, and J. Pratt, "Summary of Team IHMC ' s Virtual Robotics Challenge Entry," in *13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013.

[6] W. Decré, R. Smits, H. Bruyninckx, and J. D. Schutter, "Extending iTaSC to support inequality constraints and non-instantaneous task specification," 2009, pp. 964–971.

[7] S. Rubrecht, V. Padois, P. Bidaud, and M. De Broissia, "Constraints Compliant Control : constraints compatibility and the displaced configuration approach," in *IEEE/RSJ Intenational Conference on Intelligent Robots and Systems (IROS)*, 2010.

[8] S. Rubrecht, V. Padois, P. Bidaud, M. Broissia, and M. DaSilvaSimoes, "Motion safety and constraints compatibility for multibody robots," *Autonomous Robots*, vol. 32, no. 3, pp. 333–349, jan 2012.
[9] J.-P. Aubin., *Viability Theory*. Birkhauser, 1990.
[10] T. Fraichard and H. Asama, "Inevitable collision states - a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.
[11] G. Hirzinger and A. Albu-Schäffer, "On a new generation of torque controlled light-weight robots," in *ICRA IEEE International Conference on Robotics and Automation*, 2001, pp. 3356–3363.
[12] A. Herzog, L. Righetti, and F. Grimminger, "Experiments with a hierarchical inverse dynamics controller on a torque-controlled humanoid," *arXiv preprint arXiv:1305.2042*, 2013.
[13] A. Del Prete, F. Nori, G. Metta, and L. Natale, "Prioritized Motion-Force Control of Constrained Fully-Actuated Robots: "Task Space Inverse Dynamics"," *Robotics and Autonomous Systems*, vol. 63, pp. 150–157, 2015.
[14] N. Mansard, "http://stack-of-tasks.github.io/pinocchio/index.html," 2016.