robotics

MDPI

Article

Viability and Feasibility of Constrained Kinematic Control of Manipulators

Marco Faroni [1], Manuel Beschi [2], Nicola Pedrocchi [2] and Antonio Visioli [1,*]

[1] Dipartimento di Ingegneria Meccanica e Industriale, University of Brescia, Via Branze 38, 25123 Brescia, Italy; m.faroni003@unibs.it

[2] Istituto di Sistemi e Tecnologie Industriali Intelligenti per il Manifatturiero Avanzato, National Research Council, Via Alfonso Corti 12, 20133 Milano, Italy; manuel.beschi@stiima.cnr.it (M.B.); nicola.pedrocchi@stiima.cnr.it (N.P.)

* Correspondence: antonio.visioli@unibs.it; Tel.: +39-030-3715426

Received: 13 June 2018; Accepted: 20 July 2018; Published: 23 July 2018

check for updates

Abstract: Recent advances in planning and control of robot manipulators make an increasing use of optimization-based techniques, such as model predictive control. In this framework, ensuring the feasibility of the online optimal control problem is a key issue. In the case of manipulators with bounded joint positions, velocities, and accelerations, feasibility can be guaranteed by limiting the set of admissible velocities and positions to a viable set. However, this results in the imposition of nonlinear optimization constraints. In this paper, we analyze the feasibility of the optimal control problem and we propose a method to construct a viable convex polyhedral that ensures feasibility of the optimal control problem by means of a given number of linear constraints. Experimental and numerical results on an industrial manipulator show the validity of the proposed approach.

Keywords: manipulators; trajectory planning; kinematic constraints; optimization; viability; inverse kinematics

1. Introduction

Robotic systems typically present kinematic and/or dynamic limitations. For example, joint positions are generally bounded within an available range of motion, while actuators implicitly present velocity and acceleration/torque limits. Further constraints can be due to the environment where the robot has to operate (e.g., partial occupation of the robot workspace) or to safety reasons, which may determine velocity and acceleration limitations. Including such constraints in the development of planning and control methods is of utter importance, as their violation might lead to unrealizable motions (with consequent significant errors in the execution of the tasks) or to safety issues (for example, in the case workspace limits are not respected).

Constrained methods are widely diffused, for example, in the case of offline trajectory planning. The general approach consists in the formulation of a constrained optimization problem that should optimize a given objective, such as minimum execution time [1] or minimum energy consumption [2]. This is also the case of global redundancy resolution methods [3], where the robot has to perform an assigned task, and the redundancy can be exploited to optimize a desired objective, such as maximum manipulability [4], dexterity [5], or maximum joint range availability [6]. However, all these methods are typically performed offline as they need the prior knowledge of the task to be performed, and they lead to heavy computational burdens that do not permit their online execution. For this reason, they are not able to handle online trajectory generation and re-planning techniques and this is a relevant limitation, considering recent robotic applications such as collaborative robotics and robots in unstructured environments. To handle the robot constraints in online planning and control methods, optimization-based techniques are becoming more and more widespread as the increasing computing power of modern processors allows their real-time implementation with small sampling periods.

Quadratic programming, for example, is intensively exploited in the resolution of the Inverse Kinematic (IK) problem of redundant manipulators [7–11]. In brief, a manipulator is termed kinematically redundant with respect to a task when it has a number of degrees of freedom greater than the one of the desired task. The IK problem does not present a unique solution, as the transformation between the joint and the task space results in an undetermined system. Such problem is usually addressed at differential level in order to exploit the linear relation between joint and task velocities given by the Jacobian of the robot. This can be formulated as a chain of Quadratic Programs (QP) with decreasing priorities: the tasks are written as cost functions and the robot limits are represented as linear constraints. The possibility of including robot constraints in the QP represents a great advantage with respect to other typical methods (such as the unconstrained projection of secondary tasks in the null space of the Jacobian [12,13]). Further developments in this field explicitly face the problem of possible saturations and task deformations by introducing a scaling variable into the QP to slow down the execution of the tasks if needed [14].

A further improvement of these methods is represented by the use of Model Predictive Control (MPC) techniques, which aim at overcoming issues related to the fact that the previous methods only consider the current state of the robot. In fact, MPC-based methods are able to take into account the future evolution of the system, the tasks, and the constraints, improving the behavior of the robot in terms of task satisfaction and smoothness [15,16]. MPC is also applied to constrained motion planning [17,18] or low-level control applications [19].

Disregarding the particular field of application, a fundamental issue to address when using optimization-based control techniques is the feasibility of the online optimization problem. In fact, feasibility must be ensured in any possible state of the system. Otherwise, a solution to the problem might not exist, and this could cause the algorithm to stop (unless specific but sub-optimal strategies are activated when infeasibility occurs).

Feasibility of the online optimization problem is strictly related to the concept of set viability [20]. A set is said to be viable if, given an initial state within such set, the state trajectory can be kept within the set by means of a proper and realizable input function. In other words, keeping the state within a viable set ensures feasibility for all future time instants. On the contrary, if the state exits the viable set, infeasibility of the control problem will surely occur at a certain time in the future. This is particularly relevant in the case of state-constrained systems, as infeasibility might occur if no viability conditions are added to the control problem but only state and input limits are taken into account.

For robot manipulator control, this issue is common to any control strategy that takes into account both position and acceleration/torque constraints. In this case, the simplistic imposition of box constraints on the position and the acceleration/torque of the robot does not ensure the existence of a feasible solution to the optimization problem. For example, if a robot joint approaches its position limit with high velocity, the position bound will be exceeded due to the bounded admissible deceleration. Few works addressed this issue by means of manually-tuned heuristic strategies that aimed at reducing the velocity of the robot when it approached its position limits [21–23]. Many other control strategies did not take into account position bounds, assuming that the reference trajectories were implicitly feasible [24,25]. Notice that the viability guarantee does not ensure the feasibility of the reference trajectory, which could be only verified offline when the whole trajectory is known a priori. Such approach does not apply to online methods, which should be able to handle online trajectory generation and re-planning. In these cases, we can only ensure the feasibility of the optimal control control, which typically aims at minimizing the deviation of the performed motion with respect to the reference trajectory. This means that, although the desired motion might be infeasible, the optimal control problem remains feasible. In this case, a deformation with respect to the nominal trajectory could arise.

A formal viability guarantee was given in [26], which proposed an invariance control scheme for constrained robot manipulators. In particular, viability conditions were derived in order not to exceed workspace limitations with bounded joint accelerations. In [14], a local redundancy resolution technique was proposed: to respect a given deceleration limit in the breaking phase, the position-velocity state region was limited by imposing that the robot was able to stop within the given joint limit (in fact, limiting the state region to a viable set). This approach was similarly adopted in [27] in the context of robust constrained motion planning. Recently, viability for joint-constrained manipulators was explicitly addressed in [28], with a particular focus on the discrete-time implementation of the viability constraints.

All the above-mentioned approaches were based on the derivation of an analytical viability condition for a double integral system with bounded input and output. The resulting condition is quadratic in the velocity state. However, as all the above-mentioned works refer to local/feedback methods, such condition could be easily linearized around the current system state, resulting in variable box constraints on the control actions. However, such linearization is not possible for predictive strategies. In this case, a linear approximation of the viability set should be obtained. For example, Faroni et al.[29] gives a sufficient condition to approximate the quadratic viable set with a linear constraint and maintaining the viability property. Although the approximation technique is easy to implement and does not remarkably increase the computational complexity of the problem, the resulting allowed state region could result to be significantly smaller than the original one.

In this paper, we propose a method to derive a viable convex polyhedron for a robotic system with bounded joint positions, velocities, and accelerations. In particular, a simple optimization problem is set up to determine the maximum polyhedron that approximates the original viable set. The original quadratic condition is approximated by a polyhedron with a given number of sides, by maximizing the area of the allowed velocity-position state-region. Moreover, viability of the resulting set is ensured by imposing that, for all points on the polyhedron boundary, there exists a realizable input action that keeps the next state within the polyhedron itself, which is also demonstrated to be convex and can be therefore rewritten as a linear inequality constraint in optimization-based algorithms (such as linear MPC techniques). The paper shows that, by increasing the number of sides, the polyhedron gives a better approximation of the original viable set, as expected. This means that a larger admissible state region can be exploited by the controller when the resulting constraints are included in the optimization problem. Consequently, the controller can obtain smaller tracking errors when the desired task requires the robot joint to get close to the maximal viable set boundary. In particular, numerical results show the enlargement of the admissible state region as the number of sides increases. Such improvement is more and more evident as the maximum acceleration values gets smaller. Finally, experimental results on a Universal Robots UR10 manipulator demonstrates the validity of the proposed approach. In particular, an MPC algorithm is applied to the tracking problem of a given joint-space trajectory and the results with different viability conditions are compared. Firstly, an experimental example shows that the use of an invariance condition is of vital importance to ensure both the feasibility of the online optimization problem and the satisfaction of the manipulator limits. Secondly, the results show that the viable sets obtained by means of the proposed method permits to obtain smaller (or null) tracking errors in the case the required motion is close to the viability boundary, increasing the performance of the MPC controller.

The paper is organized as follows. Section 2 introduces the concept of set viability applied to constrained kinematic control of robot manipulators. Section 3 illustrates the proposed method for the computation of the optimal viable polyhedron and gives viability and convexity proves. Numerical results demonstrate the effectiveness of the proposed approach in Section 4, while experimental results on a Universal Robot UR10 manipulator are shown in Section 5. Section 6 concludes the paper.

2. Feasibility of the Constrained Kinematic Control Problem

Consider a generic robot joint and denote with q, $\dot{q}$, $\ddot{q}$ its position, velocity, and acceleration, respectively. The limits on q, $\dot{q}$, and $\ddot{q}$ can be therefore expressed as:

$$q_{\min} \leq q \ \leq q_{\max}, \tag{1}$$
$$\dot{q}_{\min} \leq \dot{q} \ \leq \dot{q}_{\max}, \tag{2}$$
$$\ddot{q}_{\min} \leq \ddot{q} \ \leq \ddot{q}_{\max}, \tag{3}$$

where $q_{\min}$, $\dot{q}_{\min}$, $\ddot{q}_{\min}$, and $q_{\max}$, $\dot{q}_{\max}$, $\ddot{q}_{\max}$ are the minimum and maximum joint position, velocity, and acceleration, respectively.

Assume a discrete-time implementation with sampling period T in which the acceleration is considered as constant along each sampling period. At time k, kinematic limits at the next sampling time $k+1$ can be easily written as linear inequalities in the joint acceleration $\ddot{q}_k$ as:

$$q_{\min} \leq \ q_k + T\,\dot{q}_k + \frac{1}{2}\,T^2\,\ddot{q}_k \ \leq q_{\max}, \tag{4}$$
$$\dot{q}_{\min} \leq \ \dot{q}_k + T\,\ddot{q}_k \ \leq \dot{q}_{\max}, \tag{5}$$
$$\ddot{q}_{\min} \leq \ \ddot{q}_k \ \leq \ddot{q}_{\max}. \tag{6}$$

However, the simplistic imposition of Equations (4)–(6) may result in an empty admissible set, as no feasible solution might exist that satisfies all constraints at the same time [28].

Indeed, the existence of a solution is guaranteed if Equations (4)–(6) are feasible for all the admissible states of the system. This is strictly correlated to the concept of set viability, as mentioned in Section 1. As each joint is modeled as a double integrator, the viability analysis traces back to the viability of a double integrator system with bounded input and output. By imposing the feasibility of Equations (4)–(6), the maximal viability set for the double integrator can be derived analytically [26]. Intuitively, it can be calculated by imposing that, applying the maximum deceleration $\ddot{q}_{\min}$, the joint stops at $q = q_{\max}$ with null velocity ($\dot{q} = 0$). The resulting condition (valid for the upper bound) is given by:

$$\begin{cases} q - \frac{\dot{q}^2}{2\ddot{q}_{\min}} - q_{\max} \leq 0 & \text{if } \dot{q} > 0 \\ q - q_{\max} \leq 0 & \text{otherwise} \end{cases} \tag{7}$$

which expresses a quadratic condition in the system states. An analogous condition for the lower bound can be derived likewise.

Such condition can be easily linearized around the current velocity and position, in the case of local methods, as in [14,27]. The inclusion of Equation (7) in the QP ensures that the states of the system (i.e., the joint velocities and positions) remain within a viable set for which the problem is feasible. More details about the discrete implementation in robotic systems can be found in [28].

The necessity of deriving a linear approximation of the original quadratic equation (Equation (7)) comes from the advantages given by the linear formulation in optimization-based controllers (in particular, the significant decrement of computational time and the ease of implementation of linear MPC with respect to the nonlinear one). However, in the case of predictive strategies, the linearization adopted for local methods is not applicable.

To tackle this issue, Faroni et al. [29] proposed a linear viability condition based on a single constraint for each joint. Firstly, it showed that approximating the quadratic constraint in Equation (7) by means of a straight line passing through the extreme points of the maximal viable set (i.e., $(q_{\max}, 0)$ and the intersection between Equation (7) and $\dot{q} = \dot{q}_{\max}$ in Figure 1) leads to a non-viable set. Then, it derives a linear viability condition by imposing that the maximum deceleration along the linear constraint is exactly equal to the maximum admissible deceleration. In this way, the joint states can be always kept within the resulting set by means of a realizable acceleration. Such condition is given by:

$$-\frac{\ddot{q}_{\text{min}}}{\dot{q}_{\text{max}}}q+\dot{q}+\frac{\ddot{q}_{\text{min}}}{\dot{q}_{\text{max}}}q_{\text{max}}\leq 0 \tag{8}$$

for the upper bound, and likewise for the lower one.

Inequalities (Equations (7) and (8)) are graphically represented in Figure 1. Linearity of Equation (8) is a great advantage, as it allows the straightforward inclusion in an optimization problem as a linear constraints. However, it is clear that Equation (8) implies a conservative reduction of the available state-space region compared to Equation (7), and such reduction worsens as the acceleration limits become smaller.

From a practical perspective, the reduction of the admissible state region shows its drawbacks when the robot is required to perform a motion that would violate Equation (8). In fact, the states laying between the linear and the quadratic constraints in Equations (7) and (8) are potentially realizable by the robot, but they are automatically excluded by the controller, with consequent deformation of the desired trajectory. Similarly, for redundant manipulators, shrinking the admissible state region could results in a worse satisfaction of the secondary objectives.

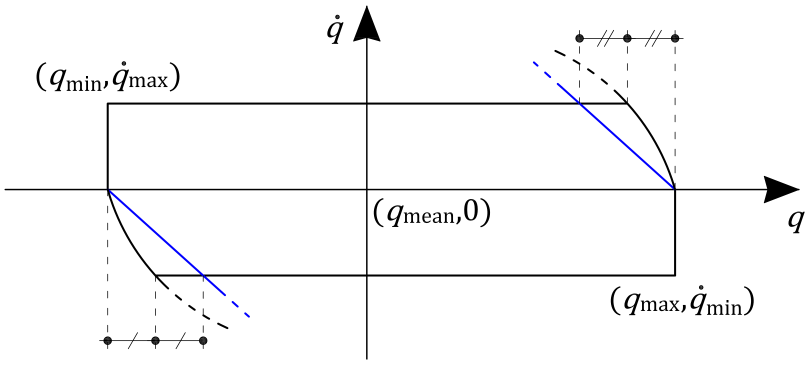

$\dot{q}$
$(q_{\text{min}},\dot{q}_{\text{max}})$
$(q_{\text{mean}},0)$
q
$(q_{\text{max}},\dot{q}_{\text{min}})$

Figure 1. Viable admissible set for the double integrator system. Black: Maximal region, given by the quadratic inequality in Equation (7). Blue: Conservative linear inequality in Equation (8), as proposed in [29].

3. Proposed Method

As mentioned in the previous section, viability conditions are fundamental to ensure the feasibility of the control problem. The derivation of linear viability conditions (e.g., Equation (8)) allows their straightforward implementation in an optimization-based control framework, such as MPC algorithms. A strategy to enlarge the admissible region without waving its linearity is therefore proposed. It consists in increasing the number of linear constraints that compose the admissible set. We can therefore state the following problem.

Problem 1. *Given the acceleration, velocity, and configuration limits for each joint, find the maximal viable convex polyhedron determined by a given number of linear inequalities.*

We address this problem by converting it into an optimization problem. In particular, for each joint, two optimization problems are set up (one for the upper position bound and one for the lower position bound). Without loss of generality, consider the upper configuration bound for a generic joint (i.e., the first quarter in Figure 1). The optimization variables of the problem are given by the coordinates of the extremes of the segment composing the polyhedron shown in Figure 2. Given a number of segments h, such variables are denoted with the vectors:

$$p_x = (p_{x,0}, \dots, p_{x,j}, \dots, p_{x,h}), \tag{9}$$

$$p_y = (p_{y,0}, \dots, p_{y,j}, \dots, p_{y,h}), \tag{10}$$

whereas a generic point on the polyhedron is denoted with $P_j(p_{x,j}, p_{y,j})$.

Now, impose that $(p_{x,0}, p_{y,0}) = (q_{\text{mean}}, \dot{q}_{\text{max}})$, where $q_{\text{mean}} = \frac{q_{\text{max}}+q_{\text{min}}}{2}$ and $(p_{x,h}, p_{y,h}) = (q_{\text{max}}, 0)$. To maximize the area covered by the polyhedron, a cost function equivalent to the opposite of such area is defined as follows:

$$\psi = \frac{1}{2}\sum_{j=1}^{h}(p_{x,j-1} - p_{x,j})(p_{y,j-1} + p_{y,j}). \tag{11}$$

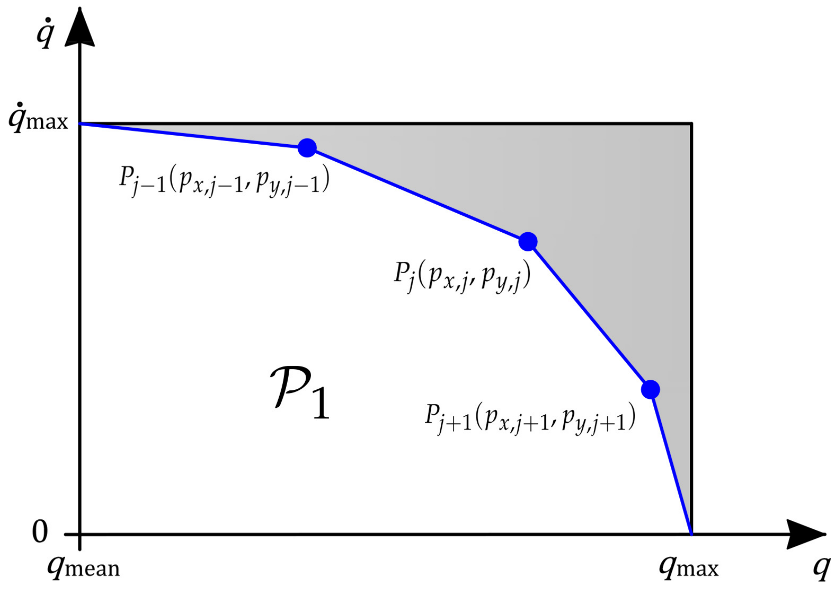


Figure 2. Construction of the polyhedron $\mathcal{P}_1$.

The extremes of the segments are required to lay in the first quarter and have to respect the maximum position and velocity bounds. This results in the following box constraints:

$$q_{\text{mean}} \le p_x \le q_{\text{max}}, \tag{12}$$

$$0 \le p_y \le \dot{q}_{\text{max}}. \tag{13}$$

Moreover, as a first necessary condition to the convexity of the polyhedron, we impose:

$$p_{x,j} - p_{x,j-1} \ge 0 \quad \forall j = 1, \dots, h, \tag{14}$$

$$p_{y,j-1} - p_{y,j} \ge 0 \quad \forall j = 1, \dots, h. \tag{15}$$

One last constraint needs to be imposed to ensure that the resulting polyhedron is viable and convex. To this purpose, we resort to a geometrical reasoning. Consider a generic segment $\overline{P_j\,P_{j+1}}$ whose extremes are the points $(p_{x,j}, p_{y,j})$ and $(p_{x,j+1}, p_{y,j+1})$ and denote with $\vec{n}_j$ the unitary vector normal to the segment and directed toward the inner region of the polyhedron. It results that:

$$\vec{n}_j = \left(\frac{p_{y,j+1} - p_{y,j}}{\|\overline{P_j\,P_{j+1}}\|}, \frac{p_{x,j} - p_{x,j+1}}{\|\overline{P_j\,P_{j+1}}\|}\right). \tag{16}$$

Consider then a generic point $P_\zeta(p_{x,\zeta}, p_{y,\zeta})$ on the segment. By applying the maximum deceleration $\ddot{q}_{\min}$ to the system, the states are forced to move on the curve given by:

$$\begin{cases} q = p_{x,\zeta} + p_{y,\zeta} t + \frac{1}{2}\ddot{q}_{\min} t^2 \\ \dot{q} = p_{y,\zeta} + \ddot{q}_{\min} t \end{cases} \tag{17}$$

The tangent vector to such curve in the $(q, \dot{q})$-plane is given by:

$$\vec{t} = \left(\frac{dq}{dt}, \frac{d\dot{q}}{dt}\right) = \left(p_{y,\zeta} + \ddot{q}_{\min} t,\ \ddot{q}_{\min}\right). \tag{18}$$

The vector $\vec{t}$ at $t = 0$ is therefore the tangent vector in the point P_ζ and results to be:

$$\vec{t}(p_{x,\zeta}, p_{y,\zeta}) = \left(p_{y,\zeta},\ \ddot{q}_{\min}\right). \tag{19}$$

Note that $\vec{t}(p_{x,\zeta}, p_{y,\zeta})$ represents the direction of the state movement when the maximum realizable deceleration is applied. The state will therefore be able to remain below the considered segment if the following condition holds:

$$\vec{n}_j \cdot \vec{t}(p_{x,\zeta}, p_{y,\zeta}) \geq 0 \qquad \forall\, (p_{x,\zeta}, p_{y,\zeta}) \in \overline{P_j\, P_{j+1}} \tag{20}$$

where $(\cdot)$ denotes the scalar product between two vectors. As

$$\min_{p_{x,\zeta}, p_{y,\zeta}} \left(\vec{n}_j \cdot \vec{t}(p_{x,\zeta}, p_{y,\zeta})\right) = \vec{n}_j \cdot \vec{t}(p_{x,j}, p_{y,j}), \tag{21}$$

condition in Equation (20) can be imposed on the sole extreme point P_j of each segment and becomes:

$$p_{y,j}(p_{y,j+1} - p_{y,j}) + \ddot{q}_{\min}(p_{x,j} - p_{x,j+1}) \geq 0. \tag{22}$$

Finally, the resulting optimization problem can be written as:

$$\begin{aligned} \underset{p_x,\, p_y}{\text{minimize}} \quad & \psi \\ \text{subject to} \quad & q_{\text{mean}} \leq p_x \leq q_{\max} \\ & 0 \leq p_y \leq \dot{q}_{\max} \\ & p_{x,j} - p_{x,j-1} \geq 0 \quad \forall j = 1, \ldots, h \\ & p_{y,j-1} - p_{y,j} \geq 0 \quad \forall j = 1, \ldots, h \\ & p_{y,j}(p_{y,j+1} - p_{y,j}) + \ddot{q}_{\min}(p_{x,j} - p_{x,j+1}) \geq 0 \quad \forall j = 1, \ldots, h-1 \\ & p_{x,0} = q_{\text{mean}} \\ & p_{y,0} = \dot{q}_{\max} \\ & p_{x,h} = q_{\max} \\ & p_{y,h} = 0 \end{aligned} \tag{23}$$

Following the same reasoning, an analogous optimization problem can be set up for the lower position bound. Note that, if the joint limits are symmetric, the solution for the lower bounds can be obtained by "mirroring" the solution of Equation (23) into the third quarter of the $(q, \dot{q})$-plane. Notice that, as Equation (23) is non-convex, global optimization such as genetic or multi-start algorithms should be adopted to solve the optimization problem (see Section 4).

Denoting with $\mathcal{P}_1$ the polyhedron determined by the segments obtained as solution of Equation (23), and with $\mathcal{P}_2$ the analogous polyhedron obtained for the lower bounds, the overall polyhedron for the single joint is given by:

$$\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \cup [q_{\min}, q_{\text{mean}}] \times [0, \dot{q}_{\max}] \cup [q_{\text{mean}}, q_{\max}] \times [\dot{q}_{\min}, 0]. \tag{24}$$

Proposition 1. *Polyhedron $\mathcal{P}$ is a viable set with respect to the given joint position, velocity and acceleration limits. Moreover, such polyhedron is convex.*

Proof (Viability). Viability of the set is implicitly ensured by the imposition of Equation (22). □

Proof (Convexity). To prove it by contradiction, denote the solution of Equation (23) with p_x^*, p_y^* and assume that the corresponding polyhedron $\mathcal{P}_1^*$ (i.e., the part of the polyhedron in the first quarter) is non-convex. Consider then a triple of consequent extremes $\{P_{j-1}, P_j, P_{j+1}\}$, which causes a non-convexity. Equations (20) and (21) imply:

$$\vec{n}_{j-1} \cdot \vec{t}(p_{x,j-1}^*, p_{y,j-1}^*) \geq 0, \tag{25}$$

$$\vec{n}_j \cdot \vec{t}(p_{x,j}^*, p_{y,j}^*) \geq 0. \tag{26}$$

Consider now the point P_c given by the projection of P_j onto $\overline{P_{j-1}\,P_{j+1}}$ and denote with $\vec{n}_c$ the vector normal to $\overline{P_{j-1}\,P_{j+1}}$. We want to prove that the polyhedron obtained by substituting P_c to P_j in $\mathcal{P}_1^*$ is a feasible and more efficient solution of Equation (23). First, constraints in Equations (12)–(15) are straightforwardly satisfied. Moreover,

$$\vec{n}_c \cdot \vec{t}(p_{x,j-1}^*, p_{y,j-1}^*) \geq \vec{n}_{j-1} \cdot \vec{t}(p_{x,j-1}^*, p_{y,j-1}^*) \geq 0 \tag{27}$$

as $\vec{n}_{j-1} \cdot \vec{i} \leq \vec{n}_c \cdot \vec{i} \leq 0$ and $\vec{n}_c \cdot \vec{j} \leq \vec{n}_{j-1} \cdot \vec{j} \leq 0$ by construction (where $\vec{i}$ and $\vec{j}$ denote the unitary vectors directed as the horizontal and the vertical axis, respectively).

Moreover, as $\ddot{q}_{\min}$ in Equation (19) is assumed to be negative, the following inequality holds:

$$\vec{n}_c \cdot \vec{t}(p_{x,c}, p_{y,c}) \geq \vec{n}_c \cdot \vec{t}(p_{x,j-1}^*, p_{y,j-1}^*) \geq 0. \tag{28}$$

Note that Equations (27) and (28) implies that the new candidate solution satisfies Equation (22) and is therefore a feasible solution to Equation (23). Moreover, the candidate solution is more efficient than the assumed one, as

$$\psi(P_0, \ldots, P_{j-1}, P_c, P_{j+1}, \ldots, P_h) \leq \psi(P_0, \ldots, P_{j-1}, P_j, P_{j+1}, \ldots, P_h), \tag{29}$$

which means that p_x^*, p_y^* are not the optimal solution of Equation (23), as a feasible and more efficient solution that eliminates such non-convexity exists. Applying this reasoning to any triples of extremes that cause a non-convexity implies that the optimal polyhedron $\mathcal{P}_1$ must be convex. The same demonstration can be applied to $\mathcal{P}_2$ leading to the same conclusion. □

4. Numerical Results

Consider the first joint of the robot manipulator Universal Robot UR10. The joint configuration and velocity limits from the datasheet are given by:

$$q_{\max} = -q_{\min} = 3.14 \text{ rad}, \qquad \dot{q}_{\max} = -\dot{q}_{\min} = 2.16 \text{ rad/s}. \tag{30}$$

Typically, acceleration limits are not explicitly given in the datasheet. Indeed, the actual dynamic limit of the joint is usually determined by the available joint torque. However, handling viability directly in terms of torque limits is typically avoided due to the complexity of the problem. The common approach consists in estimating a (usually conservative) corresponding acceleration limit for each joint separately. A method for the estimation of such acceleration limit is given, for instance, in [28]. As a first example, assume that the acceleration limits are given by $\ddot{q}_{\max} = -\ddot{q}_{\min} = 6 \text{ rad/s}^2$.

We solve the optimization problem (Equation (23)) for different values of h to evaluate the resulting polyhedrons as the number of edges grows and to compare the results to the quadratic and the linear viability constraints (Equations (7) and (8), respectively). Table 1 shows the values of the inner area of polyhedron $\mathcal{P}_1$. The values are normalized with respect to the maximal viable set obtained by means of Equation (7), whose area is therefore equal to one (last column of Table 1). As expected, the smaller area is given by Equation (8) (first column of Table 1), while the extension of the polyhedron obtained by solving Equation (23) grows with the number of edges. In other words, the larger the number of edges, the closer the optimal polyhedron is to the maximal extension (given by Equation (7)). This is clearly shown in Figure 3, which depicts the different viable sets.

An analogous example is performed by using $\ddot{q}_{\max} = -\ddot{q}_{\min} = 3 \text{ rad/s}^2$. The values of the normalized area of $\mathcal{P}_1$ are shown in Table 2 and the resulting viable sets are depicted in Figure 4. The results lead to conclusions similar to the ones given by the previous example. However, it is clear that the magnitude of the phenomenon grows as the acceleration limits get smaller.

Of course the use of a larger value of h also gives some drawbacks. First, the computational complexity of Equation (23) rapidly grows with h as the number of variables and constraints is linear in h. As an example, the time needed to solve Equation (23) for $h = 1$ and $h = 11$ was in the order of 1 second and 15 s, respectively (the computation was performed in Matlab using a multi-start gradient-descent method on a standard laptop mounting a 2.5 GHz Intel Core i5-2520M processor).

Furthermore, the value of h determines the number of linear inequalities describing the polyhedron. In optimization-based control algorithms (such as linear MPC), this affects the computational complexity of the online optimal control problem, which is typically a critical issue in online methods, as mentioned in Section 1.

Table 1. Inner area of $\mathcal{P}_1$ for different values of h in case $\ddot{q}_{\max} = -\ddot{q}_{\min} = 6 \text{ rad/s}^2$.

Equation (8)	***h* = 3**	***h* = 4**	***h* = 5**	***h* = 6**	***h* = 7**	***h* = 8**	***h* = 9**	***h* = 10**	***h* = 11**	**Equation (7)**
0.9139	0.9636	0.9773	0.9836	0.9871	0.9895	0.9911	0.9920	0.9923	0.9925	1

Table 2. Inner area of $\mathcal{P}_1$ for different values of h in case $\ddot{q}_{\max} = -\ddot{q}_{\min} = 3 \text{ rad/s}^2$.

Equation (8)	***h* = 3**	***h* = 4**	***h* = 5**	***h* = 6**	***h* = 7**	***h* = 8**	***h* = 9**	***h* = 10**	***h* = 11**	**Equation (7)**
0.8200	0.9239	0.9525	0.9656	0.9731	0.9731	0.9813	0.9838	0.9857	0.9857	1

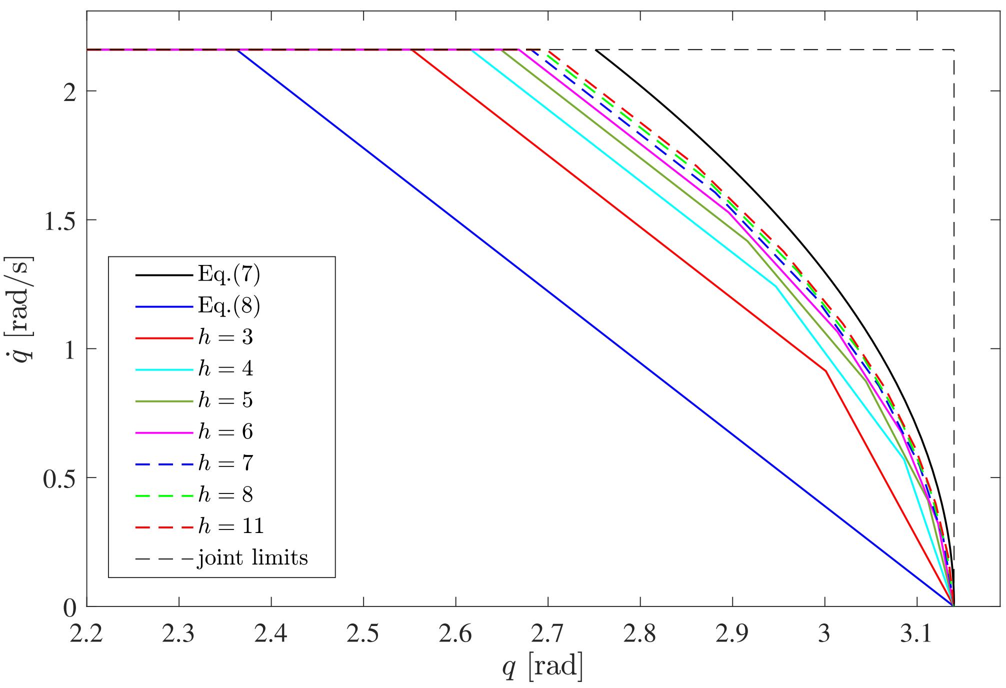


Figure 3. Comparison of the viable polyhedrons for different values of h (portion of interest of the first quarter) in the case $\ddot{q}_{\max} = -\ddot{q}_{\min} = 6$ rad/s^2.

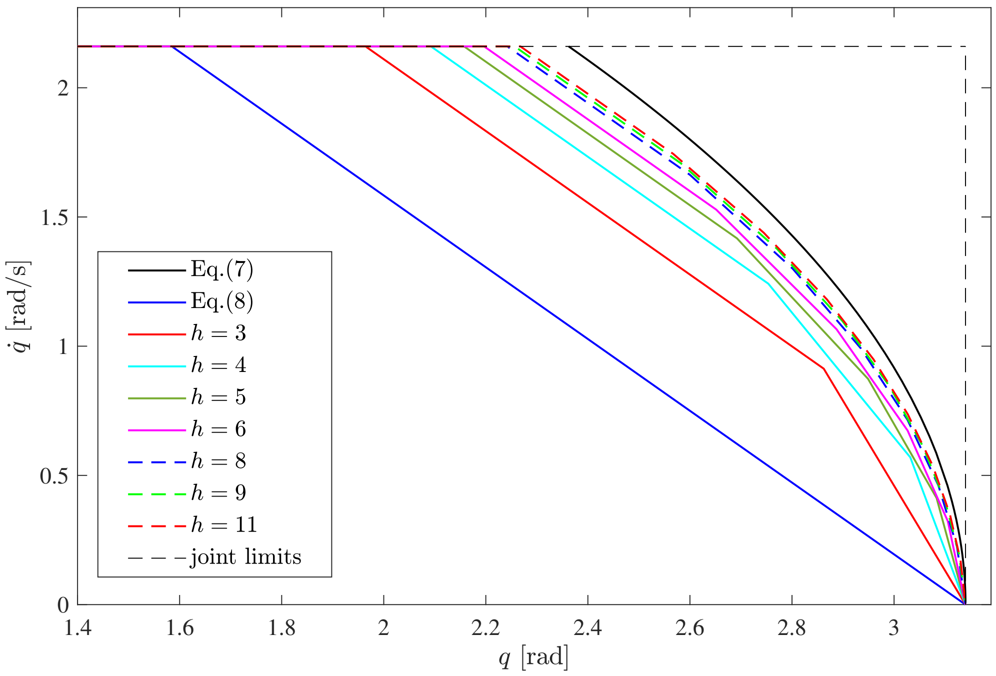


Figure 4. Comparison of the viable polyhedrons for different values of h (portion of interest of the first quarter) in the case $\ddot{q}_{\max} = -\ddot{q}_{\min} = 3$ rad/s^2.

5. Experimental Results

Experimental were also performed on a six-degrees-of-freedom Universal Robot UR10 manipulator to prove the validity of the proposed approach. For the purposes of the paper, the joint position and velocity limits have been set to slightly conservative values with respect to the ones given in the datasheet. In particular, they are set as:

$$Q_{\max} = -Q_{\min} = (\,3,\,3,\,3,\,3,\,3,\,3\,)\ \text{rad}, \tag{31}$$

$$\dot{Q}_{\max} = -\dot{Q}_{\min} = (\,2,\,2,\,3,\,3,\,3,\,3\,)\ \text{rad/s}. \tag{32}$$

The acceleration limits have been set as $\ddot{Q}_{\max} = -\ddot{Q}_{\min} = (\,3,\,3,\,3,\,3,\,3,\,3\,)\ \text{rad/s}^2$. The experimental platform is shown in Figure 5. The objective of these experiments was the evaluation of the performance of an optimization-based control algorithm when different viability constraints are implemented. To this purpose, a simple MPC scheme was applied to a trajectory following problem. The MPC control scheme is in charge of following a position reference signal in the joint space. The model implemented in the MPC consists of a double integrator for each joint, as typical of robot kinematic control [15,30]. The input action is therefore represented by the joint accelerations, which then feed the low-level controller of the robot. Joint position, velocity, and acceleration limits can be implemented as linear constraints in the MPC (see [15] for details) and the online optimal control problem results to be a QP which minimizes the weighted sum of the tracking error and the control effort, as typical of linear MPC [31]. Notice that the choice of such a simple control scheme is due to the will of highlighting the behavior of a linear MPC controller in the case of different viability conditions. However, the proposed method could be straightforwardly applied to more complex MPC techniques such as [15,16,29].

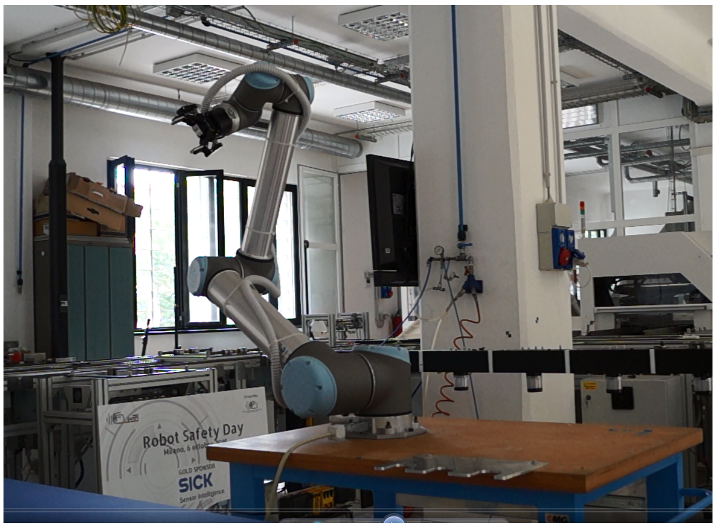


Figure 5. Universal Robot UR10 used as experimental platform.

The MPC algorithm was implemented using a sampling period $T = 8$ ms and by setting a predictive and a control horizon $N = 20$ sampling periods. The robot trajectory is controlled by means of a ROS-based control architecture. Namely, a position controller runs in a ROS Kinetic Ubuntu 16.04. The controller communicates with the robot by means of a TCP connection. The controller takes the MPC position output as reference and receives the actual joint position. The controller output is the sum of a proportional action, with gain equal to seven, and a feedforward term equal to the MPC velocity output.

We considered a simple trajectory given by a straight line in the joint space, parameterized with respect to the normalized longitudinal length along the path, denoted with $r \in [0, 1]$. The trajectory $Q_{\text{des}}(t)$ is therefore defined as:

$$Q_{\text{des}}(t) = Q_{\text{start}} + (Q_{\text{end}} - Q_{\text{start}})\, r(t), \tag{33}$$

where $Q_{\text{start}} = (\,0,\ -1,\ 0.5,\ 0,\ 0,\ 0.5\,)$ rad and $Q_{\text{end}} = (\,2.9,\ -0.5,\ -1.5,\ 1,\ -0.5,\ 2.9\,)$ rad are the initial and the final points of the trajectory, respectively, and $r : [0, t_{\text{end}}] \to [0, 1],\ t \mapsto r(t)$ is defined as a timing law with trapezoidal velocity profile, where t_{end} represents the total time of the trajectory, as depicted in Figure 6.

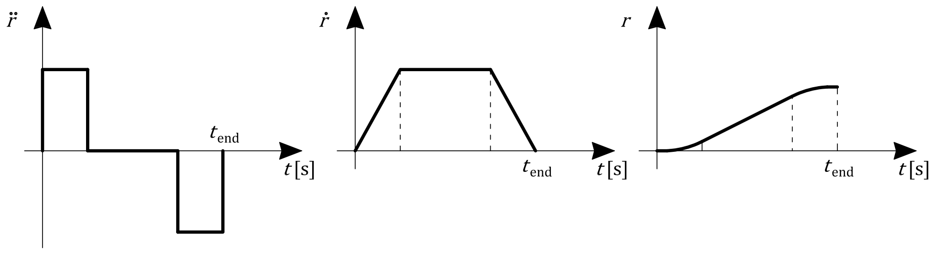


Figure 6. Timing law with trapezoidal velocity profile.

The control problem the MPC controller has to solve at each cycle k therefore results:

$$\begin{aligned} \underset{\ddot{Q}}{\text{minimize}} \quad & \sum_{i=1}^{N} \left\| Q_{\text{des}}(k+j) - Q(k+j) \right\|^2 + \lambda \sum_{i=0}^{N-1} \left\| \ddot{Q}(k+j) \right\|^2 \\ \text{subject to} \quad & \dot{Q}_{\min} \le \dot{Q}(k+j) \le \dot{Q}_{\max} \qquad \forall j = 1, \ldots, N \\ & \ddot{Q}_{\min} \le \ddot{Q}(k+j) \le \ddot{Q}_{\max} \qquad \forall j = 0, \ldots, N-1 \\ & Q(k+1) = Q(k) + T\dot{Q}(k) + 0.5T^2\ddot{Q}(k) \\ & \dot{Q}(k+1) = \dot{Q}(k) + T\ddot{Q}(k) \end{aligned} \tag{34}$$

where Q, $\dot{Q}$, and $\ddot{Q} \in \mathbb{R}^6$ are the joint position, velocity, and acceleration vectors, respectively, and the controller effort weighting factor $\lambda = 1 \times 10^{-6}$ was tuned empirically. Different implementations of the position limits will then be added to the problem in order to evaluate the different behaviors of the system.

As a first example, the total time of the trajectory was chosen as $t_{\text{end}} = 2.07$ s and the behavior of the MPC controller was evaluated in three different scenarios:

- No position bounds are implemented.
- Box position constraints are implemented, which means that the following constraints are added to Equation (34):

$$Q_{\min} \le Q(k+j) \le Q_{\max} \qquad \forall j = 1, \ldots, N \tag{35}$$

- The linear viability condition in Equation (8) is applied, that is, Equation (35), and the following constraint are added to Equation (34):

$$-\frac{\ddot{Q}_{i,\min}}{\dot{Q}_{i,\max}}\, Q_i(k+j) + \dot{Q}_i(k+j) + \frac{\ddot{Q}_{i,\min}}{\dot{Q}_{i,\max}}\, Q_{i,\max} \le 0 \qquad \forall j = 1, \ldots, N, \quad \forall i = 1, \ldots, 6; \tag{36}$$

where the subscript i denotes the ith element of the vector.

The viability constraints are imposed throughout the whole control horizon for the sake of clarity and because the small increment in the computational time (due to the larger number of linear constraints) does not represent a significant issue in the presented experimental tests. However, in the case of linear MPC, imposing such constraints only on the last time instant of the horizon would be enough to ensure feasibility of the control problem and would not give significant differences in terms of control performance. The imposition of the constraint throughout the whole horizon permits to obtain a more reliable prediction of the future trajectory. This can be helpful especially in the case the prediction is utilized in linearized methods such as [15], as a non-reliable prediction might worsen the performance of the method.

Notice that the trajectory is devised in such a way that the desired motions of the first and the sixth joints exceed the maximal viable set, as shown in Figure 7a,b. The figures also show the phase plot of the position/velocity variables computed by the MPC in the three above-mentioned cases. In the case no position bounds are implemented (dashed green line), the joint position obviously does not satisfy the position limit. However, when box position constraints are implemented in the MPC (dash-dotted purple line), infeasibility of the control problem occurs and, since that time, the joint is forced to decelerate with the maximum admissible deceleration, but the position limit cannot be satisfied anyway. This highlights the importance of the viability property. In fact, the implementation of the linear constraints in Equation (8) does ensure the feasibility of the control problem and permits to satisfy the position bound (with a deformation of the original trajectory) (dashed red line). These behaviors are clarified also in Figure 8a,b, where the ideal and the measured joint positions for Joint 1 and Joint 6 are shown.

A second experiment is performed by choosing a total trajectory time $t_{\text{end}} = 2.65$ s. In this case, the trajectory drives the first and the third joints in the region between Equations (7) and (8). The behavior of the MPC controller is evaluated in cases where three different viability constraints are implemented:

- The linear viability condition in Equation (8), that is, Equation (36) is added to Equation (34).
- The viable polyhedron obtained by solving Equation (23) with $h = 3$, that is, the following constraints are added to Equation (34):

$$\big(Q_i(k+j), \dot{Q}_i(k+j)\big) \in \mathcal{P}^i_{h=3} \qquad \forall j = 1, \ldots, N, \quad \forall i = 1, \ldots, 6; \tag{37}$$

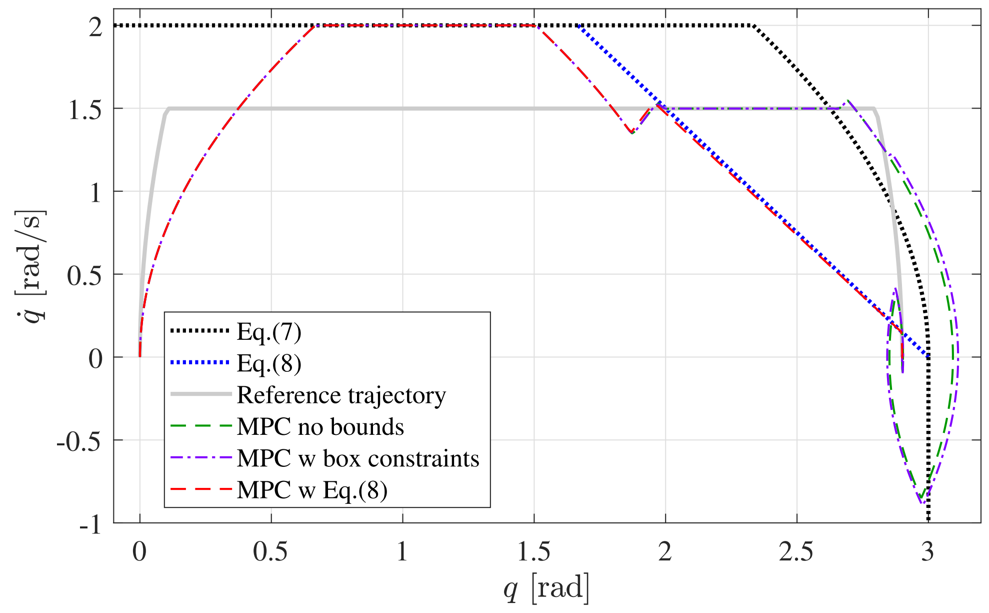
  where the superscript i denotes the viable polyhedral of the ith joint.
- The viable polyhedron obtained by solving Equation (23) with $h = 4$, that is, the following constraints are added to Equation (34):

$$\big(Q_i(k+j), \dot{Q}_i(k+j)\big) \in \mathcal{P}^i_{h=4} \qquad \forall j = 1, \ldots, N, \quad \forall i = 1, \ldots, 6. \tag{38}$$

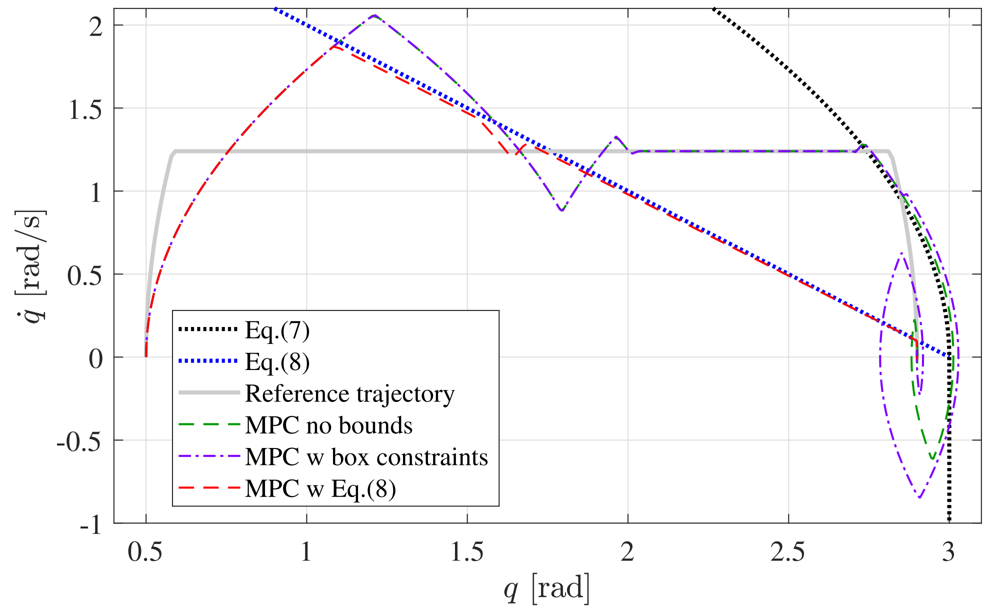
Figure 9a,b shows the state trajectory for the first joint in the three different cases. Notice that the position limit is respected and the control problem remains feasible for all cases. However, the reduction of the admissible region determined by Equation (8) gives rise to a significant modification of the original trajectory, although the desired states are always potentially realizable by the robot (as the trajectory does not exceed Equation (7)). An improvement is obtained when the polyhedron with $h = 3$ is used.

Notice that the small velocity bumps visible in the figures are due to predictive nature of the controller. In fact, as the control scheme is based on the tracking of a position reference along the predictive horizon, the controller slightly increases the velocity when it realizes the viability constraint will be activated and a deformation of the task will arise (as the state will be forced to follow the constraint). By giving a small acceleration before the activation of the constraint, the controller minimizes the future deviation with respect to the given position reference.

Finally, the use of $h = 4$ permits to obtain a viable polyhedron that is large enough to enclose the whole desired trajectory. These behaviors are clarified also in Figure 10a,b, where the ideal and the measured joint positions for Joint 1 and Joint 6 are shown.

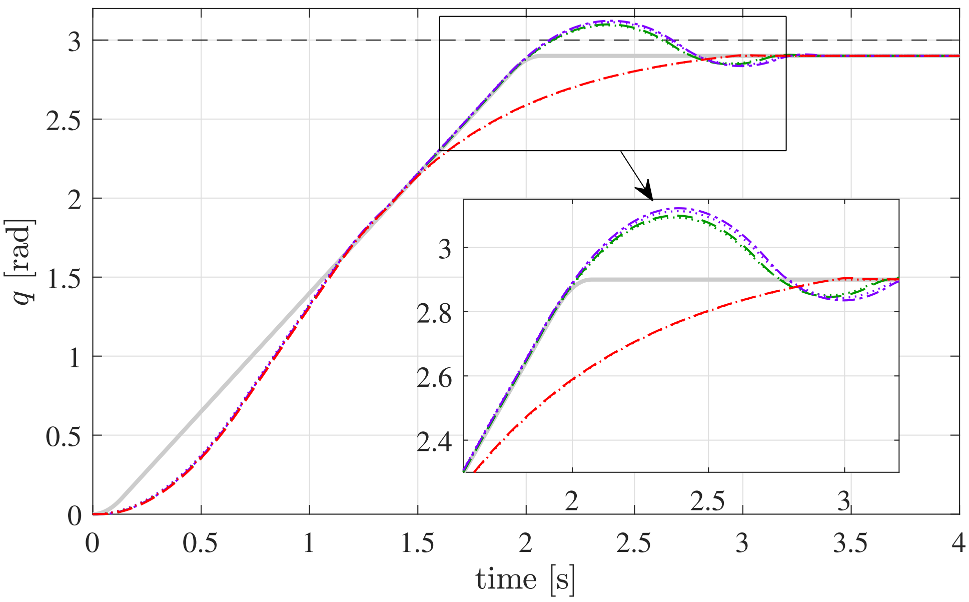


(**a**)

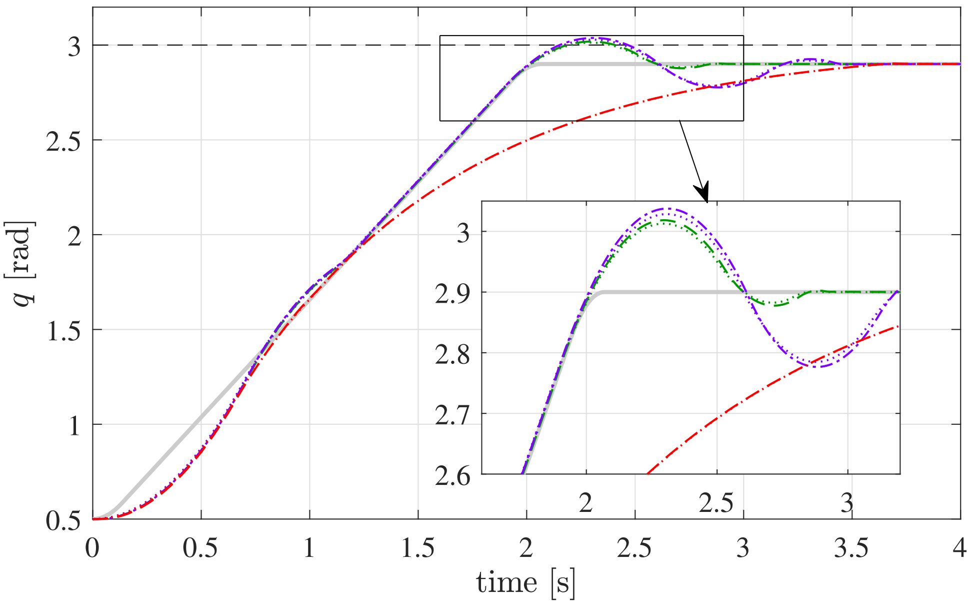


(**b**)

Figure 7. State trajectory in the $(q, \dot{q})$-plane for (**a**) Joint 1 and (**b**) Joint 3 for $T_{tot} = 2.07$ s: Reference trajectory (gray solid line); MPC without position bounds (green dashed line); MPC with box position bounds (purple dash-dot line); MPC with linear viability inequality (Equation (8)) (red dashed line); maximal viable set (black dotted line); and linear viability inequality ((Equation (8)) (blue dotted line).

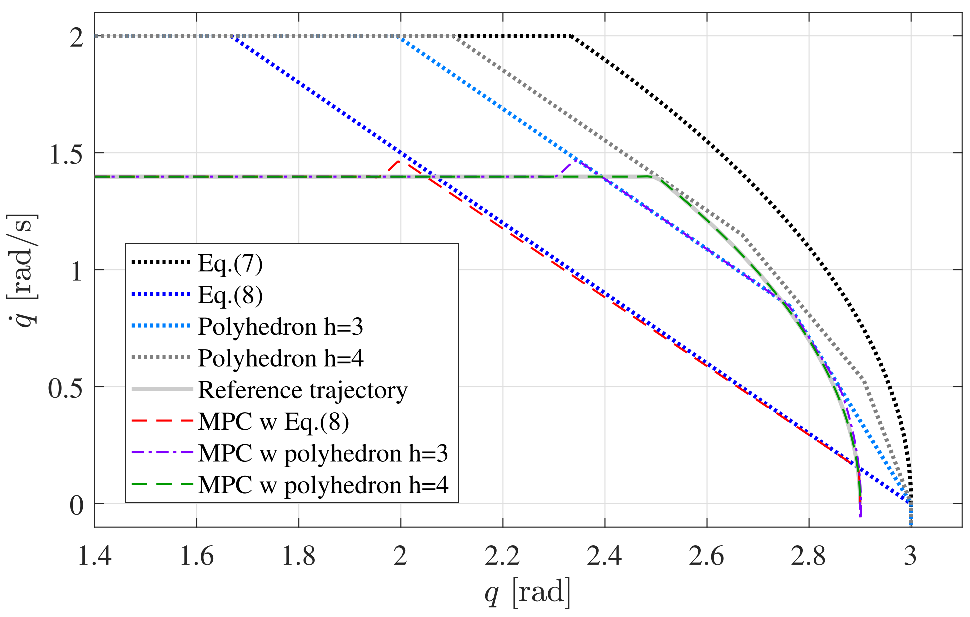


(**a**)

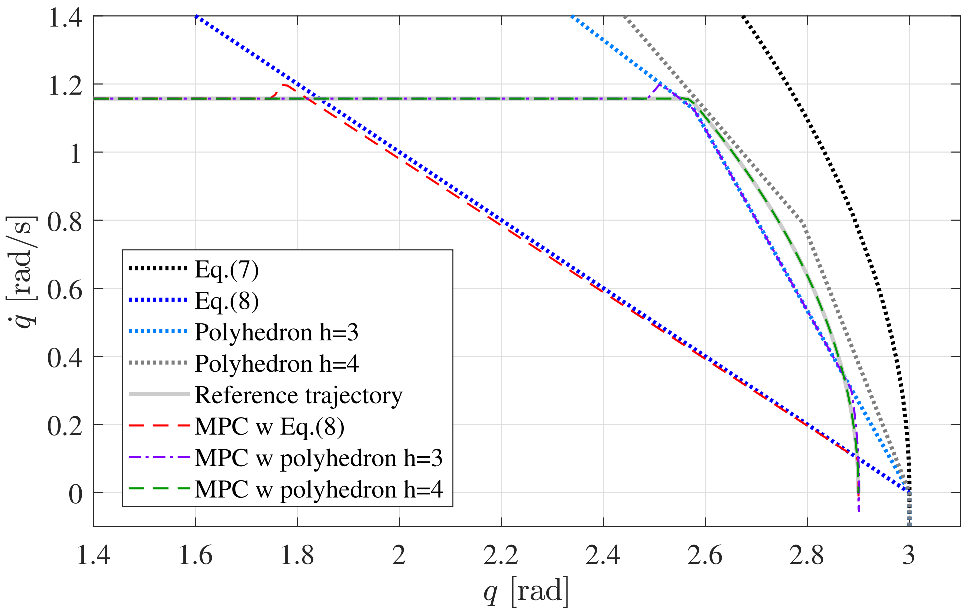


(**b**)

Figure 8. Position of (**a**) Joint 1; and (**b**) Joint 3 for $T_{\text{tot}} = 2.07$ s: Reference trajectory (gray solid line). Output of MPC given as reference signal to the low-level controller: MPC without position bounds (green dotted line); MPC with box position bounds (purple dotted line); and MPC with linear viability inequality (Equation (8)) (red dotted line). Measured position: MPC without position bounds (green dashed line); MPC with box position bounds (purple dash-dot line); and MPC with linear viability inequality (Equation (8)) (red dashed line).

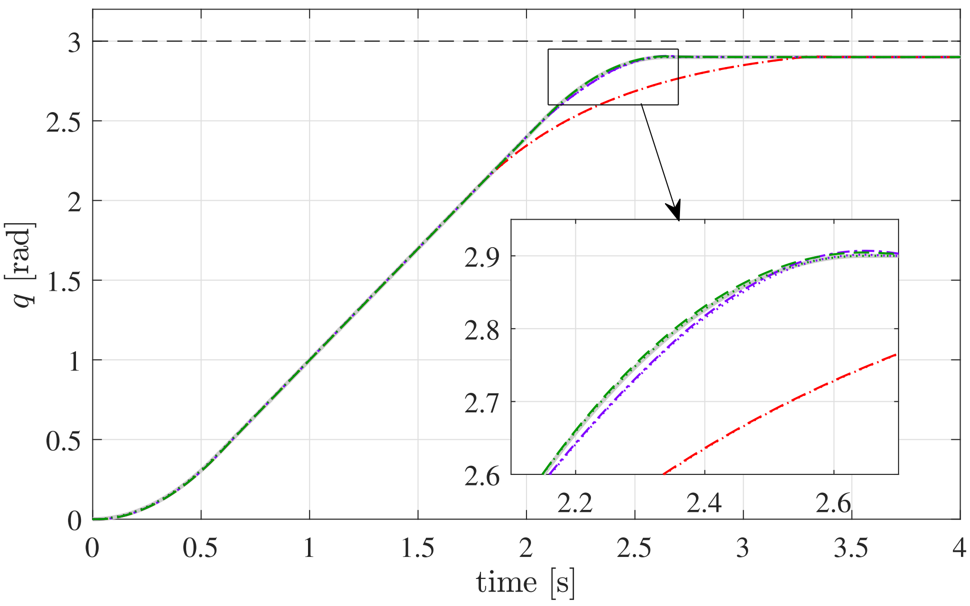


(**a**)

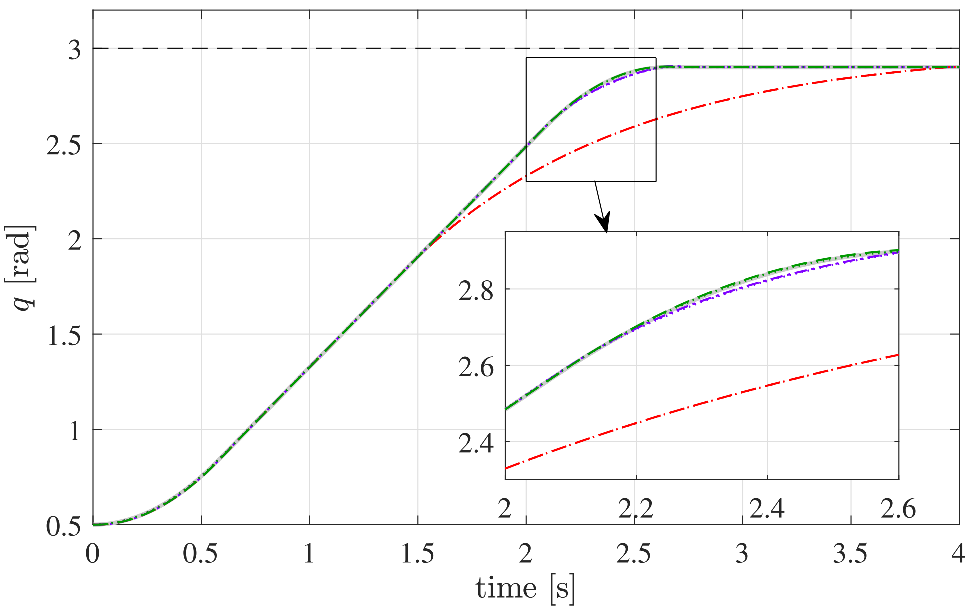


(**b**)

Figure 9. State trajectory in the $(q, \dot{q})$-plane for (**a**) Joint 1 and (**b**) Joint 3 for $T_{tot} = 2.65$ s: Reference trajectory (gray solid line); MPC with linear viability inequality (Equation (8)) (red dashed line); MPC implementing the viable polyhedron with $h = 3$ (purple dash-dot line); and MPC implementing the viable polyhedron with $h = 4$ (dashed green). Viable sets: Maximal (black dotted line); Linear viability inequality (Equation (8)) (blue dotted line); Polyhedron with $h = 3$ (light blue dotted line); and Polyhedron with $h = 4$ (gray dotted line).



(**a**)



(**b**)

Figure 10. Position of (**a**) Joint 1 and (**b**) Joint 3 for $T_{\text{tot}} = 2.65$ s. Reference trajectory (gray solid line). Output of MPC given as reference signal to the low-level controller: MPC with linear viability inequality (Equation (8)) (red dotted line); MPC implementing the viable polyhedron with $h = 3$ (purple dotted line); and MPC implementing the viable polyhedron with $h = 4$ (green dotted line). Measured position: MPC with linear viability inequality (Equation (8)) (red dashed line); MPC implementing the viable polyhedron with $h = 3$ (purple dash-dot line); and MPC implementing the viable polyhedron with $h = 4$ (green dashed line).

6. Conclusions

In this paper, we have proposed a method to compute the maximal viable polyhedron for a robot manipulator with bounded joint positions, velocities, and accelerations. In the proposed approach, each joint is considered separately and two optimization problems are devised: one for the upper bounds and one for the lower ones. Given its number of sides, the resulting polyhedron maximizes the area of the admissible position/velocity region. Moreover, the set is proven to be viable and convex. In this way, it can be easily implemented as linear constraints in optimization-based control methods, such as linear MPC. Numerical results demonstrate that the percentage of area covered by the polyhedrons increases as the number of sides grows. This allows the controller to exploit a larger admissible state region and, thus, gives it a broader margin of maneuver in the case the desired motion drives the robot in proximity of the viability boundary. In this case, a better task following can be achieved, as demonstrated by means of experimental results on a six-degrees-of-freedom manipulator.

Author Contributions: M.F. devised the method and wrote the paper; M.B. and N.P. conceived and performed the experiments; and A.V. supervised the whole research work.

Funding: The research leading to these results was partially funded by the European Union H2020-CS2-738039-Eureca: Enhanced Human Robot cooperation in Cabin Assembly tasks.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Verscheure, D.; Demeulenaere, B.; Swevers, J.; Schutter, J.D.; Diehl, M. Time-Optimal Path Tracking for Robots: A Convex Optimization Approach. *IEEE Trans. Autom. Control* **2009**, *54*, 2318–2327. [CrossRef]
2. Riazi, S.; Wigström, O.; Bengtsson, K.; Lennartson, B. Energy and Peak Power Optimization of Time-Bounded Robot Trajectories. *IEEE Trans. Autom. Sci. Eng.* **2017**, *14*, 646–657. [CrossRef]
3. Kazerounian, K.; Wang, Z. Global versus local optimization in redundancy resolution of robotic manipulators. *Int. J. Robot. Res.* **1988**, *7*, 3–12. [CrossRef]
4. Faroni, M.; Beschi, M.; Visioli, A.; Molinari Tosatti, L. A global approach to manipulability optimisation for a dual-arm manipulator. In Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation, Berlin, Germany, 6–9 September 2016; pp. 1–6.
5. Lee, J.; Chang, P.H. Redundancy resolution for dual-arm robots inspired by human asymmetric bimanual action: Formulation and experiments. In Proceedings of the IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 26–30 May 2015; pp. 6058–6065.
6. Liegeois, A. Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms. *IEEE Trans. Syst. Man Cybern.* **1977**, *7*, 868–871.
7. Cheng, F.T.; Chen, T.H.; Sun, Y.Y. Resolving manipulator redundancy under inequality constraints. *IEEE Trans. Robot. Autom.* **1994**, *10*, 65–71. [CrossRef]
8. Kanoun, O.; Lamiraux, F.; Wieber, P.B. Kinematic control of redundant manipulators: Generalizing the task priority framework to inequality tasks. *IEEE Trans. Robot.* **2011**, *27*, 785–792. [CrossRef]
9. Escande, A.; Mansard, N.; Wieber, P.B. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *Int. J. Robot. Res.* **2014**, *33*, 1006–1028. [CrossRef]
10. Liu, M.; Tan, Y.; Padois, V. Generalized hierarchical control. *Auton. Robots* **2016**, *40*, 17–31. [CrossRef]
11. Rocchi, A.; Hoffman, E.M.; Caldwell, D.G.; Tsagarakis, N.G. Opensot: A whole-body control library for the compliant humanoid robot Coman. In Proceedings of the IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 26–30 May 2015; pp. 6248–6253.
12. Siciliano, B.; Slotine, J.J.E. A general framework for managing multiple tasks in highly redundant robotic systems. In Proceedings of the International Conference on Advanced Robotics, Pisa, Italy, 19–22 June 1991; pp. 1211–1216.
13. Flacco, F.; De Luca, A. Discrete-time redundancy resolution at the velocity level with acceleration/torque optimization properties. *Robot. Auton. Syst.* **2015**, *70*, 191–201. [CrossRef]
14. Flacco, F.; De Luca, A.; Khatib, O. Control of Redundant Robots Under Hard Joint Constraints: Saturation in the Null Space. *IEEE Trans. Robot.* **2015**, *31*, 637–654. [CrossRef]

15. Faroni, M.; Beschi, M.; Visioli, A.; Molinari Tosatti, L. A Predictive Approach to Redundancy Resolution for Robot Manipulators. In Proceedings of the IFAC World Congress, Toulouse, France, 9–14 July 2017; pp. 8975–8980.
16. Faroni, M.; Beschi, M.; Visioli, A. Predictive Inverse Kinematics for Redundant Manipulators: Evaluation in Re-Planning Scenarios. In Proceedings of the IFAC Symposium on Robot Control, Budapest, Hungary, 27–30 August 2018.
17. Buizza Avanzini, G.; Zanchettin, A.M.; Rocco, P. Reactive Constrained Model Predictive Control for Redundant Mobile Manipulators. In *Intelligent Autonomous Systems 13, Advances in Intelligent Systems and Computing*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 1301–1314.
18. Ghazaei Ardakani, M.M.; Olofsson, N.; Robertsson, A.; Johansson, R. Real-Time Trajectory Generation using Model Predictive Control. In Proceedings of the IEEE International Conference on Automation Science and Engineering, Gothenburg, Sweden, 24–28 August 2015; pp. 942–948.
19. Ferrara, A.; Incremona, G.P.; Magni, L. A robust MPC/ISM hierarchical multi-loop control scheme for robot manipulators. In Proceedings of the Annual Conference on Decision and Control, Florence, Italy, 10–13 December 2013; pp. 3560–3565.
20. Blanchini, F. Set invariance in control. *Automatica* **1999**, *35*, 1747–1767. [CrossRef]
21. Saab, L.; Ramos, O.E.; Keith, F.; Mansard, N.; Soueres, P.; Fourquet, J.Y. Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. *IEEE Trans. Robot.* **2013**, *29*, 346–362. [CrossRef]
22. Park, K.C.; Chang, P.H.; Kim, S.H. The enhanced compact QP method for redundant manipulators using practical inequality constraints. In Proceedings of the IEEE International Conference on Robotics and Automation, Leuven, Belgium, 20 May 1998; pp. 107–114.
23. Kanehiro, F.; Morisawa, M.; Suleiman, W.; Kaneko, K.; Yoshida, E. Integrating geometric constraints into reactive leg motion generation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 4069–4076.
24. Wensing, P.M.; Orin, D.E. Generation of dynamic humanoid behaviors through task-space control with conic optimization. In Proceedings of the IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 3103–3109.
25. Guarino Lo Bianco, C.; Gerelli, O. Online trajectory scaling for manipulators subject to high-order kinematic and dynamic constraints. *IEEE Trans. Robot.* **2011**, *27*, 1144–1152. [CrossRef]
26. Scheint, M.; Wolff, J.; Buss, M. Invariance control in robotic applications: Trajectory supervision and haptic rendering. In Proceedings of the IEEE American Control Conference, Seattle, WA, USA, 11–13 June 2008; pp. 1436–1442.
27. Zanchettin, A.M.; Rocco, P. Motion planning for robotic manipulators using robust constrained control. *Control Eng. Pract.* **2017**, *59*, 127–136. [CrossRef]
28. Del Prete, A. Joint Position and Velocity Bounds in Discrete-Time Acceleration/Torque Control of Robot Manipulators. *IEEE Robot. Autom. Lett.* **2018**, *3*, 281–288. [CrossRef]
29. Faroni, M.; Beschi, M.; Pedrocchi, N.; Visioli, A. Predictive Inverse Kinematics for Redundant Manipulators with Task Scaling and Kinematic Constraints. *IEEE Trans. Robot*, **2018**, submitted.
30. Gerelli, O.; Guarino Lo Bianco, C. Nonlinear variable structure filter for the online trajectory scaling. *IEEE Trans. Ind. Electron.* **2009**, *56*, 3921–3930. [CrossRef]
31. Wang, L. *Model Predictive Control System Design and Implementation Using MATLAB®*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2009.