Garelli, F.; Gracia Calandin, LI.; Sala, A.; Albertos Pérez, P. (2011). Sliding mode speed auto-regulation technique for robotic tracking. Robotics and Autonomous Systems. 59(7-8):519-529. doi:10.1016/j.robot.2011.03.007.

Sliding mode speed auto-regulation technique for robotic tracking

Fabricio Garelli[a], Luis Gracia[b], Antonio Sala[b], Pedro Albertos[b]

[a] *CONICET and Universidad Nacional de La Plata, C.C.91 (1900), La Plata, Argentina (e-mail: fabricio@ing.unlp.edu.ar).*
[b] *Dept. of Systems Engineering and Control, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain (e-mails: luigraca@isa.upv.es, asala@isa.upv.es, pedro@aii.upv.es).*

Abstract

In advanced industry manufacturing involving robotic operations, the required tasks can be frequently formulated in terms of a path or trajectory tracking. In this paper, an approach based on sliding mode conditioning of a path parametrization is proposed to achieve the greatest tracking speed which is compatible with the robot input constraints (joint speeds). Some distinctive features of the proposal are that: (1) it is completely independent of the robot parameters, and it does not require a-priori knowledge of the desired path either, (2) it avoids on-line computations necessary for conventional analytical methodologies, and (3) it can be easily added as a supervisory block to pre-existing path tracking schemes. A sufficient condition (lower bound on desired tracking speed) for the sliding mode regulation to be activated is derived, while a chattering amplitude estimation is obtained in terms of the sampling period and a tunable first-order filter bandwidth. The algorithm is evaluated on the freely accesible 6R robot model PUMA-560, for which a path passing through a wrist singularity is considered to show the effectiveness of the proposal under hard tracking conditions.

Keywords: Robotic tracking, sliding mode, input saturation, multivariable systems

1. Introduction

A major issue in robotics is the tracking of *reference trajectories*. In most practical applications that use industrial and/or mobile robots [1, 2]

(e.g., machining, arc-welding, adhesive application, spray painting, assembling, inspection, object transportation in warehouses, surveillance in known environments, etc.), the robot task is based on tracking a given *path* with *negligible* error and with the highest possible velocity, so that the cycle time of the robot task is minimized. In this manner, both quality and productivity indexes can be enlarged. However, both the accuracy and the speed with which this tracking can be performed is strongly related with the joint actuators physical limitations, which are seldom considered in commercial robots to regulate the robot forward motion. Instead, the tracking speed usually has to be computed a priori by the robot operator in order to avoid an error message.

The reference path, i.e. the path to be followed, can usually be expressed as one-dimensional curve in the cartesian space, i.e., a time-dependent vector which can be parameterized in terms of a scalar *motion parameter* whose first-order time derivative is related to the path *tracking speed* by well-known expressions. In this sense, the idea of a parameterized path has been successfully used in several research works to adjust the tracking speed so that the robot is able to track different types of references, even those ones crossing robot kinematics singularities [3, 4, 5, 6]. Among the more significant works in this research line, a self-paced fuzzy controller was designed in [7] to adjust the tracking speed of two-dimensional paths in accordance with contour conditions such as curvature. Similarly, a path parametrization satisfying input and state constraints was obtained in [8] using look-ahead optimization and a prediction of the evolution of the robot, for which a priori knowledge of the desired path and a robot model are required. More recently, a time warp is considered in [9] to slow down the task-space trajectory when joint limits are encountered. In [10], instead, the power limits of the electrical motors driving the robot are considered to measure the maximum possible velocity and force that can be physically generated by the robot to perform the required task. Finally, path tracking is rigourously divided into a geometric (desired error) and a dynamic (desired speed) task in [11], where speed profiles are assigned for nonlinear systems to track non-smooth paths.

This paper proposes a simple method which allows regulating the robotic tracking speed in order to avoid path deviations because of joint actuators constraints. In order to achieve this goal, a sliding mode auxiliary loop is added to conventional path tracking schemes, which is inspired on recent reference conditioning algorithms developed to deal with constraints in multivariable control systems [12, 13]. It acts as a supervisory block, since it is

only activated when the desired speed would lead the joint actuators to reach their limit values. Interestingly, a practical consequence is the fact that, if a sufficiently high speed reference (motion parameter) is set, the method computes the maximal tracking speed which is compatible with the joint actuator limits. As an advantage over most of the above cited proposals, the proposed technique is independent of the main path tracking control algorithm and it does not require a priori knowledge of the desired path. Since the method was thought to be used with commercial industrial robots, speed joint constraints are assumed (see Section 2). A well-known six-revolute (6R) robotic arm is taken as case study, for which a path passing through a robot singularity is considered. The method implementation can be even carried out by means of analog electronics since the switching device is confined to the low-power side of the system.

The article is organized as follows. In the next section the classical kinematic control scheme for robotic path tracking is recalled, and some common alternatives to deal with actuator constraints are introduced. Section 3 presents some basic concepts of variable structure theory and develops the sliding mode auto-regulation technique for tracking speed in order to avoid path errors due to actuator nonlinearities. In Section 4 simulation results are presented using the free-access 6R robot model PUMA 560, for which the main distinctive features of the method are illustrated. Finally, some conclusions are given.

2. Classical control scheme for robotic path tracking

Due to the computational complexity of advanced control algorithms developed in current robotics research [14, 15], classical control techniques are still widely used in industrial robot applications. In most practical robot systems, the controller consists of three nested control loops: an analog actuator current controller, an analog velocity controller, and a typically digital position controller. The great majority of industrial robot manufacturers implement the inner control loops (i.e., the current loop and the velocity loop) internally in the so-called *joint controllers* and do not allow the robot operator to modify these loops. Conversely, the outer-loop position controller is usually open for the user and can be manipulated.

Let us now discuss some common setups for robot path tracking in the above described framework.

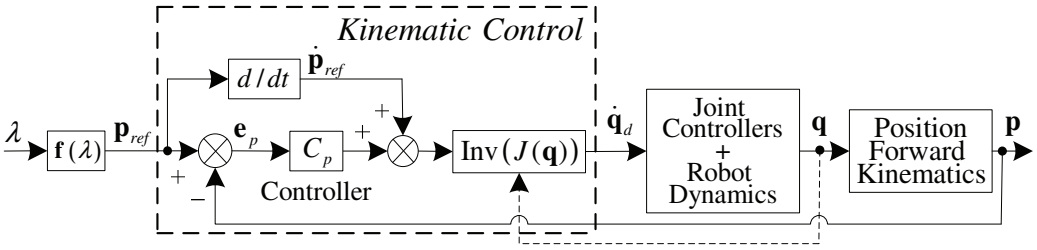

Figure 1: Robotic path tracking control scheme.

2.1. Kinematic control setups

Workspace-coordinates kinematic control. Let us denote as $\mathbf{p}_{ref}(t)$ the position reference defining the desired path in some user-chosen workspace coordinates (for instance, Cartesian position and Euler-angle orientation of the end effector). As mentioned in the introduction, the trajectory $\mathbf{p}_{ref}(t)$ can be usually expressed in terms of a desired path function $\mathbf{f}(\lambda)$ whose argument is the so-called motion parameter $\lambda(t)$ as

$$\mathbf{p}_{ref} = \mathbf{f}(\lambda), \tag{1}$$

and, therefore, the desired speed comes from:

$$\dot{\mathbf{p}}_{ref} = \frac{\partial \mathbf{f}}{\partial \lambda}\dot{\lambda}. \tag{2}$$

A kinematic control block in a robot closes a loop using position information in both joint coordinates, to be denoted as $\mathbf{q}$, and workspace coordinates $\mathbf{p}$, as well as desired position and speed information from the target trajectory.

The relationship between the $\mathbf{q}$ configuration and the end-effector position/orientation $\mathbf{p}$ is highly nonlinear, generically expressed as:

$$\mathbf{p} = \mathbf{l}(\mathbf{q}), \tag{3}$$

where the function $\mathbf{l}$ is called the kinematic function of the robot model. The first order kinematics results in:

$$\dot{\mathbf{p}} = \frac{\partial \mathbf{l}}{\partial \mathbf{q}}\dot{\mathbf{q}} = J(\mathbf{q})\dot{\mathbf{q}}, \tag{4}$$

where $J(\mathbf{q})$ is denoted as the Jacobian matrix or simply *Jacobian* of the kinematic function.

Fig. 1 shows a common setup for the kinematic control block in robot path tracking, consisting of a two-degree of freedom (2-DOF) control structure which incorporates a correction based on the position error $\mathbf{e}_p = \mathbf{p}_{ref} - \mathbf{p}$ by means of the position loop controller C_p plus a feedforward term depending on the first-order time derivative of the position reference, i.e. $\dot{\mathbf{p}}_{ref}$.

Note that, in this scheme the error correction is performed in the Cartesian space and then the inverse of the robot jacobian $J(q)$ is used to obtain the joint velocity vector $\dot{\mathbf{q}}$. Indeed, for a non-redundant manipulator (square Jacobian), the joint velocities $\dot{\mathbf{q}}$ producing a particular end-effector motion $\dot{\mathbf{p}}_0$ can be written as:

$$\dot{\mathbf{q}} = J^{-1}(\mathbf{q})\dot{\mathbf{p}}_{\mathbf{0}}, \tag{5}$$

and the kinematic control loop is in charge of determining the desired value for $\dot{\mathbf{p}}_0$ as a function of current position ($\mathbf{p}$) and current target trajectory point ($\mathbf{p_{ref}}$) and speed $\dot{\mathbf{p}}_{ref}$. Once $\dot{\mathbf{p}}_0$ is computed, (5) is applied and sent to the actuators.

The Jacobian of a generic robotic arm can be easily obtained with the vectorial approach described in [16]. It is well-known that there are certain workspace limits and internal positions where J is singular. This matter is later discussed in Section 4.

Joint-coordinates kinematic control. Another conventional approach for kinematic control consists of performing the error correction directly in the joint space [17]. This second approach requires to compute the position inverse kinematics, i.e., $\mathbf{q} = \mathbf{l^{-1}}(\mathbf{p})$. In any case, the proposed technique also applies for that or any other kinematic control.

2.2. Dealing with actuator constraints

In order to account for input constraints, joint speed saturation is from now on considered between the desired joint speeds $\dot{\mathbf{q}}_d$ of Fig. 1 and the achievable ones, denoted as $\dot{\mathbf{q}}_{ds}$.

Naturally, the maximum values of the *robot control signals*, which are given by the power constraints of the actuators, limit the path tracking speed. Basically, the following three approaches can be found in practical applications in order to face with robot actuators constraints:

a) To use a (conservative) low tracking speed, so that the robot control signals never exceed their maximum values.

b) To also use a fixed tracking speed, but higher than the previous one, in such a way that the robot control signals saturate at least once during the tracking.

c) To compute for each point on the path the maximum tracking speed allowed by the limits of the control signals and to use that value for the motion parameter speed.

The first approach is extremely conservative and thus a not advisable solution. In effect, it gives rise to an excessively slow path tracking, which indeed wastes the tracking capabilities of the robotic system. The second approach, which is the classical one, has as its main drawback that when the control signals are saturated the robot losses the reference and even leaves the desired path, which makes it inappropriate for high-accuracy applications. The third option is the best choice among the three listed practical approaches; however, it depends on the desired path and on the robot Jacobian and, hence, it is more involved computationally and, furthermore, modeling errors might give a speed over the desired limits.

In practical implementations of the second or third options, the actuator limitations are typically faced in two different ways:

1. as a direct and *independent saturation* element for each joint, for which:

$$\dot{q}_{ds,i} = \begin{cases} \dot{q}_{max,i} & \text{if } \dot{q}_{d,i} > \dot{q}_{max,i} \\ \dot{q}_{d,i} & \text{if } \dot{q}_{min,i} \leq \dot{q}_{d,i} \leq \dot{q}_{max,i},\ i = 1, ..., n \\ \dot{q}_{min,i} & \text{if } \dot{q}_{d,i} < \dot{q}_{min,i} \end{cases} \tag{6}$$

   with $\dot{q}_{max,i}$ and $\dot{q}_{min,i}$ denoting the maximum and minimum actuator (speed-servo) output of the corresponding joint, and n the the robot's degrees-of-freedom. For the sake of simplicity, it is assumed in the following that speed limits are symmetric, i.e. $\dot{q}_{min,i} = -\dot{q}_{max,i}$, although the methodologies to be presented can be trivially modified if that were not the case.

2. as a *directionality preserving saturation*, in which the joint speeds vector ($\dot{\mathbf{q}}_{ds}$) direction remains constant, hence the direction of $\dot{p}$ does so as well, but its modulus is scaled as:

$$\dot{\mathbf{q}}_{ds} = f_{dir}\ \dot{\mathbf{q}}_{d} \tag{7}$$

where

$$f_{dir} = \begin{cases} \dfrac{1}{\|F_N\ \dot{\mathbf{q}}_d\|_\infty} & \text{if } \|F_N\ \dot{\mathbf{q}}_d\|_\infty > 1, \\ 1 & \text{otherwise,} \end{cases} \tag{8}$$

notation $\|\cdot\|_\infty$ denotes element-wise maximum (infinity norm) and F_N is a diagonal matrix with the ith diagonal element equal to $1/\dot{q}_{max,i}$.

The latter is the most frequent approach in robotic tracking schemes to address joint speed limitations, and thus it will be the case considered in the SM auto-regulation algorithm proposed in the next section.

In the following, a simple methodology to regulate the tracking speed in presence of actuator constraints is presented, which can be added as a supervisory block to any tracking scheme independently of the robot model and the desired path.

3. Tracking speed auto-regulation technique

3.1. Background on sliding modes

The kinematic-loop control to be proposed in this work incorporates some sliding-mode elements. Sliding-mode control is a well-developed discipline and the reader is referred to references such as [18, 19, 20] for ample detail. Among other attractive features, sliding-mode controllers are easy to implement (see (10) below), reduce the order of the system dynamics, and provide robustness to matched uncertainties and external disturbances. Because of its interesting properties, a large number of papers presenting practical applications of SM control have been reported. For instance, in the recent contributions [21, 22, 23, 24] the application of SM to robotic systems is discussed. The basic ideas to be used in later sections are recalled here.

A variable structure system comprises a set of two continuous subsystems with an associated switching function that determines a manifold on the state space, the so-called sliding surface. According to the sign of the switching function, the control signal takes one of different possible values, leading to a discontinuous control law.

In particular, consider the following dynamical system:

$$\dot{\mathbf{x}} = h(\mathbf{x}) + g(\mathbf{x})u \tag{9}$$

where $\mathbf{x} \in \Re^{\mathbf{n}}$ is the system state, u is the discontinuous control signal, and $h(\mathbf{x})$ and $g(\mathbf{x})$ are vector fields in $\Re^n$. The variable structure control law is

defined as

$$u = \begin{cases} u^- & \text{if} \quad \sigma(\mathbf{x}) < \mathbf{0} \\ u^+ & \text{if} \quad \sigma(\mathbf{x}) > \mathbf{0} \end{cases} \tag{10}$$

according to the sign of the auxiliary output $\sigma(\mathbf{x})$. The sliding surface $\mathcal{S}$ is defined as the manifold where the auxiliary output, also called switching function, vanishes. That is,

$$\mathcal{S} = \{\mathbf{x} \in \Re^{\mathbf{n}} | \, \sigma(\mathbf{x}) = \mathbf{0}\}. \tag{11}$$

The two basic ideas of variable-structure (sliding-mode) control are:

- Find a suitable auxiliary output σ such that some prescribed properties of the closed-loop dynamics are fulfilled (stability, time constant, etc.)
- Enforce the state to reach the prescribed sliding surface and, henceforth, to remain ("slide") on it by applying the control u_{eq} enforcing $\dot{\sigma} = 0$, not explicitly but by means of a very fast switching action arising from the discontinuity in (10). This mode of operation is denoted as sliding mode (SM) or sliding regime.

Once this sliding mode is established, the prescribed manifold imposes the new system dynamics.

If, as a result of the switching policy (10), the reaching condition

$$\begin{cases} \dot{\sigma}(\mathbf{x}) < \mathbf{0} & \text{if} \quad \sigma(\mathbf{x}) > \mathbf{0} \\ \dot{\sigma}(\mathbf{x}) > \mathbf{0} & \text{if} \quad \sigma(\mathbf{x}) < \mathbf{0}, \end{cases} \tag{12}$$

holds in a neighborhood of $\mathcal{S}$, or, equivalently

$$\lim_{\sigma \to 0} \sigma \dot{\sigma} < 0, \tag{13}$$

locally holds at both sides of the surface, a switching sequence at very high frequency (ideally infinite) occurs, constraining the system state trajectory to slide on $\mathcal{S}$. The inherent low-pass characteristics of realizable physical systems averages that switching signal so that its behaviour is equal to the so-called equivalent control u_{eq} fulfilling:

$$\dot{\sigma} = \frac{\partial \sigma}{\partial \mathbf{x}} \left(h(\mathbf{x}) + g(\mathbf{x}) u_{eq} \right) = 0 \tag{14}$$

so the sliding-mode regime will be maintained as long as

$$min\{u^-, u^+\} < u_{eq} < max\{u^-, u^+\} \tag{15}$$

From (9), (10) and (14) it can be easily observed that for a sliding motion to exist on $\mathcal{S}$ (in other words, to satisfy condition (13)), the auxiliary output $\sigma(\mathbf{x})$ must have unitary relative degree with respect to the discontinuous signal, i.e. its first derivative must explicitly depend on u [18]:

$$\frac{\partial \sigma}{\partial \mathbf{x}} g(\mathbf{x}) \neq \mathbf{0}. \tag{16}$$

The inequality (16) is known as *transversality condition*.

3.2. Proposed sliding-mode conditioning of path tracking speed

Differing from conventional SM control, where the discontinuous signal is commonly used as the main control action (this is the case of the robotic applications reported in [21, 22, 23, 24]), the approach to be presented in this section exploits sliding regime as a simple and robust way of determining the maximum rate of change of the motion parameter λ which is compatible with the robot actuators physical limits. Its purpose is, once the robot is sufficiently close to the desired path, to automatically regulate the speed with which the path is followed in such a way that the control signal generated by the path reference through the kinematic control does never exceed the prescribed limits, thus preventing the robot from path deviations due to actuator constraints.

The sliding mode technique proposed to regulate the path following speed is presented in Fig. 2, whose constituent elements will be described below. It assumes that the input to the path-tracking system is the desired motion parameter speed $\dot{\lambda}_d$; such input signal will be "conditioned" in such a way that saturation is avoided.

The kinematic control block of Fig. 2 represents a path tracking control scheme as the one depicted by Fig. 1 (but not necessarily identical). Now, a saturation block has been included between the desired joint speeds ($\dot{\mathbf{q}}_d$) and the achievable ones ($\dot{\mathbf{q}}_{ds}$). As already mentioned, *directionality preserving* saturation will be assumed present in the referred block, as it is the most common approach in robotic tracking schemes to address joint speed limitations. However, it is worth mentioning that the validity of the proposed supervisory technique is completely independent of how saturation is performed.

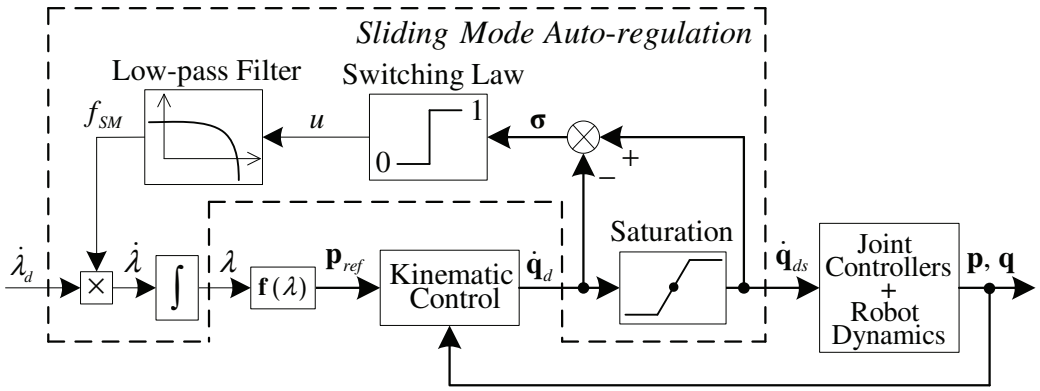

Figure 2: Proposed sliding mode auto-regulation technique for robotic path tracking (saturation block is assumed to be directionality preserving)

From the saturation block, an auxiliary loop is added to the conventional path tracking configuration in order to generate the conditioned tracking speed. To this end, the discontinuous signal u is determined by means of the commutation law:

$$u = \begin{cases} u^+ = 1 & \text{if } \boldsymbol{\sigma}(\mathbf{x}) = \mathbf{0}_n \\ u^- = 0 & \text{otherwise} \end{cases} \tag{17}$$

where $\mathbf{0}_n$ denotes the null vector of dimensions $n \times 1$, and the switching function vector $\boldsymbol{\sigma}$ is defined as:

$$\boldsymbol{\sigma}(\mathbf{x}) \;=\; \dot{\mathbf{q}}_{ds} - \dot{\mathbf{q}}_d \tag{18}$$

The maximum tracking speed $\dot{\lambda}$ which avoids inconsistencies between the kinematic controller outputs and the real robot input signals is then generated from u by means of a first-order low-pass filter

$$\dot{f}_{SM} = -a_f \, f_{SM} + a_f \, u, \tag{19}$$

with the filter output f_{SM} being a "sliding mode factor" such that $\dot{\lambda}$ is obtained as $\dot{\lambda} = f_{SM}\dot{\lambda}_d$. The filter has unit gain at low frequencies and its bandwidth a_f needs to be chosen sufficiently fast for quick stops to be allowed, but slow enough in order to smooth out $\dot{\lambda}$. As we will see in Section 4, the best choice for the filter bandwidth is strongly related with the path to be followed.

Actually, the switching function vector (18) determines two boundary sliding surfaces for each one of the n coordinates:

$$\begin{array}{l} \overline{\mathcal{S}}_i = \{\mathbf{x} | \dot{q}_{d,i} - \dot{q}_{max,i} = 0\} \\ \underline{\mathcal{S}}_i = \{\mathbf{x} | \dot{q}_{d,i} - \dot{q}_{min,i} = 0\} \end{array} \qquad i = 1, ..., n \tag{20}$$

From (2), (5), (7), (8), and (19) it can be easily deduced that these surfaces are only reached provided the desired motion parameter speed satisfies

$$\dot{\lambda}_d \geq \frac{1}{\left\| F_N \; J^{-1} \; \dfrac{\partial \mathbf{f}}{\partial \lambda} \right\|_\infty} \tag{21}$$

as, indeed, if the above condition does not hold in at least one point of the path, the actuation limits will not be reached. These conditions are the equivalent conditions for sliding mode establishment (13). Then, if $\dot{\lambda}_d$ verifies (21) on *all points of the path* at least one of the actuators will be saturating at maximum speed and, hence, the conditioning algorithm implicitly provides the maximum-speed directionality-preserving path tracking solution.

Note that this reaching condition (21) is not enforced by the switching action. Indeed, equation (21) shows how the maximal achievable tracking speed depends on the own path reference $\mathbf{f}$, the robot kinematics J, the actual robot configurations (J is function of $\mathbf{q}$) and, naturally, the saturation levels F_N. Note that (21) is particularly small for robot configurations close to singularities or against abrupt path changes. In fact, in the former case some singular values (and vectors) of J^{-1} are very big; in the latter case, it is $\partial \mathbf{f}/\partial \lambda$ who takes a very large value.

Hence, according to (17) and (18), when a joint speed reaches its maximum value, the discontinuous signal u is forced equal to zero in order to avoid surpassing the actuator bound. This switching makes the motion parameter speed to slow down and the corresponding control action $\dot{q}_{d,i}$ to fall below its limit $\dot{q}_{max,i}$, which in turn produces by means of the switching law (17) that $u = 1$ again. In this way, as the forward evolution on the reference path continues being limited by actuator i, the signal u will be switching between 0 and 1 at high frequency and a sliding regime will transiently establish on either the surface $\overline{\mathcal{S}}_i$ or $\underline{\mathcal{S}}_i$, depending on whether the upper or lower saturation limit was going to be exceeded.

As a consequence of this sliding mode, the tracking speed given by $\dot{\lambda}$ will be continuously adjusted in such a way that the control action $\dot{q}_i$ sent to

joint i does not exceed the actuator limit. If because of the desired motion parameter speed the robot is forced to saturate other joint actuator, say actuator $j \neq i$, then the same reasoning can be followed for surfaces $\overline{\mathcal{S}}_j$ or $\underline{\mathcal{S}}_j$, one of which will be from then on the responsible of the path following speed attenuation.

Observe that for the trivial sliding function vector (18) to satisfy the unitary relative-degree necessary condition, apart from the filter being a first-order one, the controller C_p must be biproper. For the case of (very unusual) strictly-proper controllers, the strategy is still applicable but the sliding function should be redefined in order to include additional controller states (see e.g. the switching functions defined in [12] to delimited crossed interactions in decentralized control of multivariable systems).

3.3. Some implementation issues

3.3.1. Switching frequency and chattering

As in all sliding-mode controls, the theoretically infinite switching frequency cannot be achieved in practice because all physical systems have finite bandwidth. In analog sliding-mode implementations (i.e., switching with operational amplifiers set up as comparators) the actual switching frequency will depend on the bandwidth of the electronic components. In computer implementations, the switching frequency is directly the inverse of the sampling period. Finite-frequency commutation makes the system leave the theoretical sliding mode and, instead, its states evolve inside a "band" around $\sigma = 0$.

In direct sliding-mode control the switching is in the main control action of the controlled process: high-power, high-frequency sharply-discontinuous inputs are not suitable for some processes (such as mechanical actuators subject to fatigue issues) and some remedies to the situation must be put in place to overcome these so called chattering problems [25].

Contrarily to the above, in reference conditioning setups, such as the particular case of the proposed path tracking algorithm, sliding mode is confined to the low-power side of the system. Hence, fast electronic devices can be used to implement the discontinuous action (in effect, the algorithm could be even implemented via analog electronics using a dual operational-amplifier chip). What is more, the sliding regime and its switching law could actually be a few program lines of a microprocessor. Thus, differing from conventional sliding mode control, the current application presents a continuous speed command to the actuator subsystem so the chattering issues are

greatly alleviated. Only a residual band due to discrete implementations will remain, as discussed below.

As previously discussed, computer implementations must operate at a finite frequency, so the sliding surface $\sigma_i = 0$ implicit in (20) gets converted into a band $|\sigma_i| \leq \Delta_i$.

Let us consider now the common case in which a sampling period T_s is given by the discrete implementation of the main position control loop. We are interested in estimating an upper bound for the chattering amplitude Δ_i so that the condition $|\sigma_i| \leq \Delta_i$ is guaranteed during SM on, say, the surface $\overline{\mathcal{S}}_i$.

The basic idea is considering that, for instance, an Euler-integration of the first-order filter (19) results in the unity-gain discrete system:

$$f_{SM}(k) = (1 - a_f T_s) f_{SM}(k-1) + a_f T_s u(k) \tag{22}$$

Hence when $u(k)$ switches from zero to one, approximately, if T_s is reasonably small so that $(1 - a_f T_s) \approx 1$, the sample-to sample increment of f_{SM} is $a_f T_s$.

This would lead to an increment on the motion parameter speed of $a_f T_s \dot{\lambda}_d$ and, if the kinematic control in Fig. 1 were used, it would translate to a sample-to-sample increment of the speed command approximately given by the expression:

$$\Delta_i \approx \left[J^{-1}(\mathbf{q})\right]_{i,*} \frac{\partial f}{\partial \lambda} a_f T_s \dot{\lambda}_d \tag{23}$$

where $\left[J^{-1}(\mathbf{q})\right]_{i,*}$ denotes the ith row of the inverse Jacobian. The above expression could be used to select parameters a_f and T_s such that the speed-command increments are below a predefined thresholds. In broad terms, in order to reduce Δ_i, the filter bandwidth must be decreased or the sampling rate must be increased.

The filter bandwidth also influences the supervisor "reaction time", i.e., the time required to completely adapt to trajectory changes requiring an abrupt speed change. In order for the proposed scheme to work in practice, the filter bandwidth must be significantly higher than the frequency content of the initially desired trajectory $\dot{\mathbf{p}}_{ref} = \frac{\partial f}{\partial \lambda} \dot{\lambda}_d$. Indeed, if the desired trajectory included high-curvature fast-acceleration movements, a low bandwidth of the supervision mechanism would result in tracking error, as intuitively expected. Nevertheless, as shown in the case study section, reasonable fast sampling rate and bandwidth make the above problems negligible in practical implementation.

3.3.2. Choice of sampling periods

Note that, by inserting sampler and zero-order-hold elements wherever they might be needed, the sampling rates of the tracking speed auto-regulation algorithm, kinematic control and robot dynamic control may be different. In particular, the SM based algorithm may be run at a faster sampling rate than that of the joint controllers in order to achieve a better approximation to the continuous-time developments, for instance, with a reduced chattering in (23).

For the sake of simplicity, in the simulations of Section 4 the same value is used for all sampling periods, even if in an industrial robot application the joint controllers might likely work at a lower sampling frequency.

3.3.3. Computing the inverse of the robot Jacobian

Note that the proposed sliding-mode speed regulation technique does *not* need any Jacobian computation as it is independent of the underlying kinematic control loop, which is one of the distinctive advantages of the proposal. However, in order to clarify issues about its behavior in singularities, and to explain in a clearer way the design choices later made in the case study section, the inverse Jacobian computation is discussed below.

As seen in Section 2, the robot Jacobian J relates the joint speeds vector $\dot{\mathbf{q}}$ to the workspace speed $\dot{\mathbf{p}}$ of the robot end-effector (EE). Now, concerning its inverse, it is recalled that the inversion of the robot Jacobian J gives rise to numerical problems when the determinant of the Jacobian vanishes, which as already known occurs at singular points. Then, a modified Jacobian inverse is employed in this section to cope with singularities. Particularly, the Jacobian inverse block of Fig. 1 has been implemented for the simulations of next section as

$$\mathrm{Inv}(J) = \frac{\mathrm{sign}(\det(J))}{\max(|\det(J)|, \epsilon)} \mathrm{adjoint}\left(J^T\right), \tag{24}$$

i.e., the well-known adjoint-transpose formula for the matrix inverse has the determinant in the denominator replaced by a minimum value ϵ when its absolute value tends to zero. This approximate inverse preserves the Jacobian inverse directionality. In other words, near singular points the direction of the Jacobian inverse is preserved but its modulus is bounded in order to avoid numerical ill-conditioning.

It is noticed here that although (24) was chosen to evaluate the proposed methodology, any other alternative for avoiding Jacobian singularity could

have been employed, such as those ones described in [3].

3.3.4. Effect of the feedback terms

The proposed kinematic control has two components, a feedback error-based control and a feedforward auto-regulated trajectory generator (2-DOF structure). In theory, the proposed motion-parameter conditioning generates a trajectory reference which can be followed by the robot under the prescribed saturation constraints. In that sense, if the robot started on exactly the same point as the reference path does, it would track the rest of the path with no error (assuming negligible modeling error).

Then, as in any 2-DOF setup, the effect of the error-based controller corrects two issues:

- Initial conditions not in the starting point of the path
- Modeling error and process/sensor noise.

It is easy to understand that the proposed motion-parameter conditioning technique stops the set-point movement if initial conditions or transient tracking error are big. Basically, the supervisor blocks reduce the maximum motion-parameter speed as tracking error increases. The intensity of the effect (i.e., the error figures forcing the path-generator to stop updating reference points) depend on the gain of the feedback regulator: the larger the regulator gain, the smaller the error bound which stops the motion parameter ($\dot{\lambda} = 0$). This is in agreement to what it is expected from conventional feedback regulators (the larger the gains, the smaller the errors); hence, the feedback-gain tuning conveniently tunes both the closed-loop behaviour regarding disturbance rejection and modeling error and the regulation of the reference path's speed if errors are high.

4. Case study: 6DOF robot arm

In this section the main features of the proposed SM auto-regulation technique are illustrated through simulation results on the well-know 6DOF robotic arm PUMA-560, which is a classical 6R serial manipulator with spherical wrist. A path passing through a wrist singularity point is considered in Subsection 4.2 in order to show the effectiveness of the method under hard control conditions.

The results shown have been obtained with the *Robotics Toolbox* (Release 7.1) for MATLAB® developed by P. Corke [26], which is available to download for free, and which includes the kinematic model of the PUMA-560 robot.

4.1. Conditions and parameters for the simulations

Simulations were run under the following conditions:

i) A kinematic framework is considered, i.e. the dynamics given by the joint controllers is considered much faster than the dynamics given by the position loop, and therefore the actual joint speed vector $\dot{\mathbf{q}}$ is assumed approximately equal to the saturated desired joint speed vector $\dot{\mathbf{q}}_{ds}$.

ii) The six elements of the workspace coordinate vector have been defined as follows: the cartesian $[x\ y\ z]$ coordinates have been chosen as representation of the end-effector position; the roll-pitch-yaw Euler angles $[\alpha\ \beta\ \gamma]^{\mathrm{T}}$ have been chosen for the end-effector orientation. The units for linear and angular dimensions are meters and radians, respectively.

iii) Four cases have been considered: in the first three cases the tracking speed $\dot{\lambda}$ is constant, while in the fourth case the tracking speed is auto-regulated with the proposed SM technique. Moreover, in the first case the joint speeds are ideally unconstrained, whereas in the other three cases the joint speeds are constrained. In this regard, the second case uses a direct saturation for the joint speeds, whereas the third and fourth cases use a directionality-preserving saturation, see Subsection 2.2.

iv) A maximal tracking speed was aimed, i.e. the desired motion $\dot{\lambda}_d$ does always satisfy condition (21).

v) For the sake of simplicity, a proportional controller has been used for the correction of the position error, i.e. $C_p = K_p$.

In all the simulations, the linear position of the reference path is given by the following helicoidal path:

$$\begin{cases} x_{ref}(\lambda) = x_{ref\_ini} + 0.1\,(\cos(\lambda) - 1) \\ y_{ref}(\lambda) = y_{ref\_ini} + 0.1\sin(\lambda) \\ z_{ref}(\lambda) = z_{ref\_ini} - 0.1\lambda, \end{cases} \tag{25}$$

with $\lambda = 0...2\pi$, while and the end-effector orientation $[\alpha_{ref}(\lambda)\ \beta_{ref}(\lambda)\ \gamma_{ref}(\lambda)]^{\mathrm{T}}$ is defined in the corresponding subsections.

It is important to recall that for the PUMA-560 manipulator the Z-axis of the robot base frame is aligned with the first joint and its origin is located at the same height of the second joint, i.e. the shoulder joint.

Furthermore, the following parameter values have been used in the simulation: determinant bound $\epsilon = 10^{-5}$, controller gain in all coordinates $K_p = 10$. The sampling period T_s used in the first three simulated cases was 2.5 milliseconds. The maximum/minimum joint speed limitations are assumed as $\dot{q}_{max,i} = -\dot{q}_{min,i} = 0.5\,\text{rad/s}$ with $i = 1, \ldots, 6$.

4.2. Simulation for a regular reference path

Firstly, we consider a constant orientation reference for the robot EE and initial robot pose error. We take in particular $\mathbf{p}_{ref}(0) = [0.58 \ -0.15 \ 0.14 \ 0 \ \pi \ 0]^{\mathrm{T}}$ as the initial value for the path reference, whilst the robot pose starts at $\mathbf{p}(0) = [0.68 \ -0.25 \ 0.19 \ 0.2 \ 2.94 \ 0.1]^{\mathrm{T}}$.

Fig. 3 to Fig. 6 compare the results obtained with the four considered cases. Dotted lines depict the ideal unconstrained case, dashed-dotted and dashed lines draw respectively the direct saturated and directionality preserving saturated cases, and solid curves show the SM auto-regulation achievements.

For comparative purposes, the constant tracking speed used in the first three simulated cases has been taken as $\dot{\lambda} = 1.26$, such that they take exactly the same time as the SM auto-regulation technique for completing the required helicoidal path. Naturally, these conventional approaches can be sped up by increasing $\dot{\lambda}$, but at the expense of greater position errors. In fact, Fig. 3 reveals that, despite their greater initial errors, both the direct saturated and directionality saturated cases lose the reference path after having (or almost having) reached it. This can also be appreciated in the 3D and 2D views of the desired and followed paths shown in Fig. 6. The reason of these path deviations is that the robot is unable to follow the given path at the specified constant speed because of joint speed saturation. Indeed, these conventional methods would require "hand-tuning" $\dot{\lambda}$ to achieve the goal of fast tracking without drifting out of the desired path. This is what the proposed auto-regulation transparently performs.

The solid lines of the figures show the effectiveness of the auto-regulation proposal. Fig. 3 and Fig. 6 confirm that, once the initial error is reduced making use of the directionality preserving saturation, the method avoids path errors by regulating the motion parameter speed $\dot{\lambda}$. For this particular case in which $\lambda_d = 5$ was taken, the algorithm gives the maximal tracking speed compatible with the actuator constraints, thus minimizing the time required to complete the path. This is shown by Fig. 4, where it can be seen

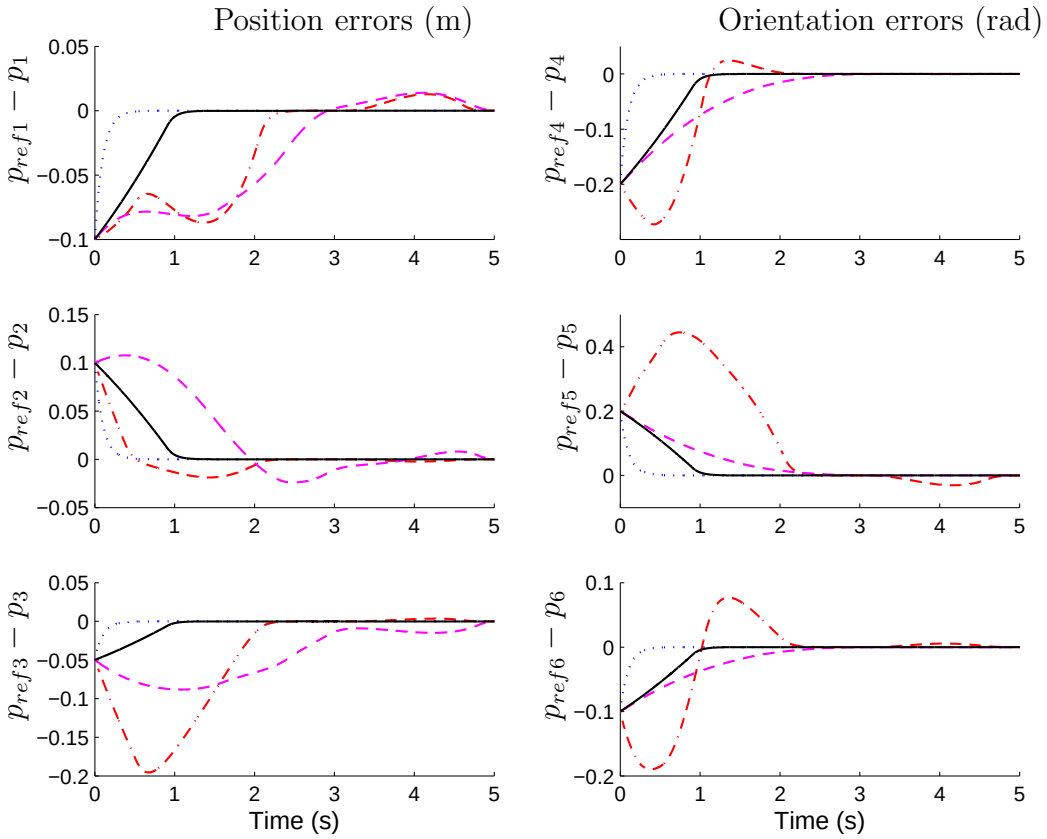

Figure 3: End-effector position and orientation errors (workspace coords.) for: ideal unconstrained case (dotted), direct saturation (dashed-dotted), directionality-preserving saturation (dashed) and SM auto-regulation technique (solid).

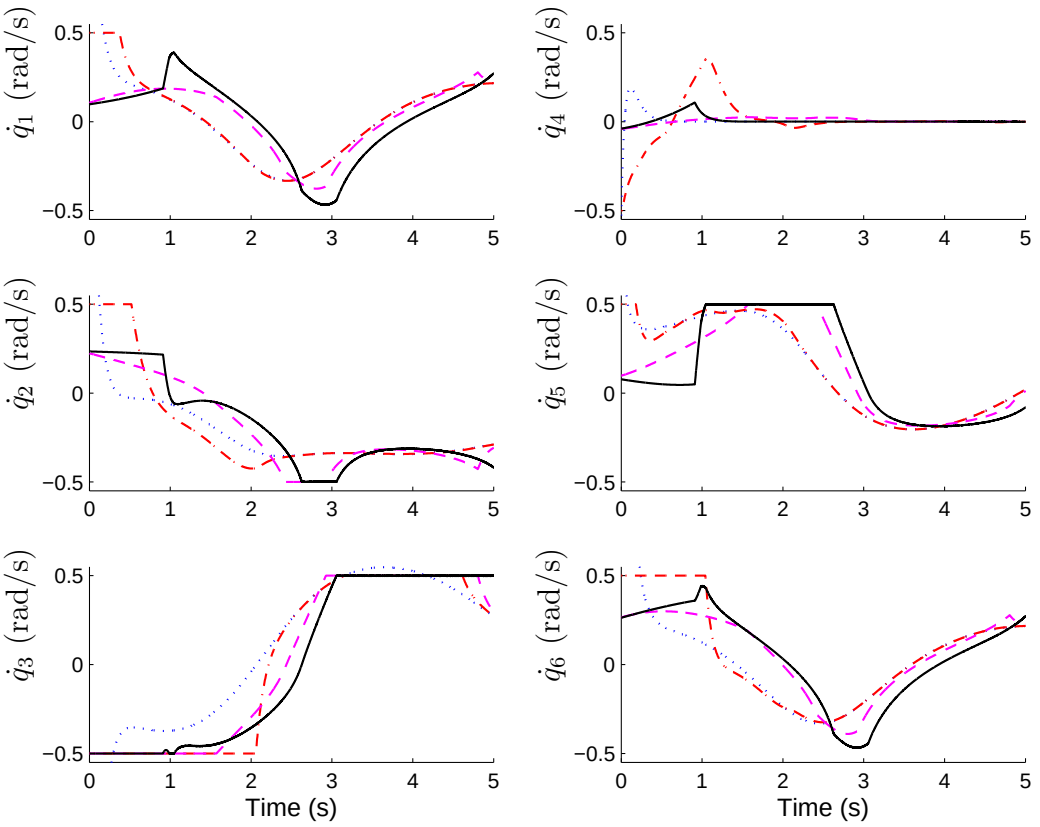

Figure 4: Joint velocities for: ideal unconstrained case (dotted), direct saturation (dashed-dotted), directionality-preserving saturation (dashed) and SM auto-regulation technique (solid).

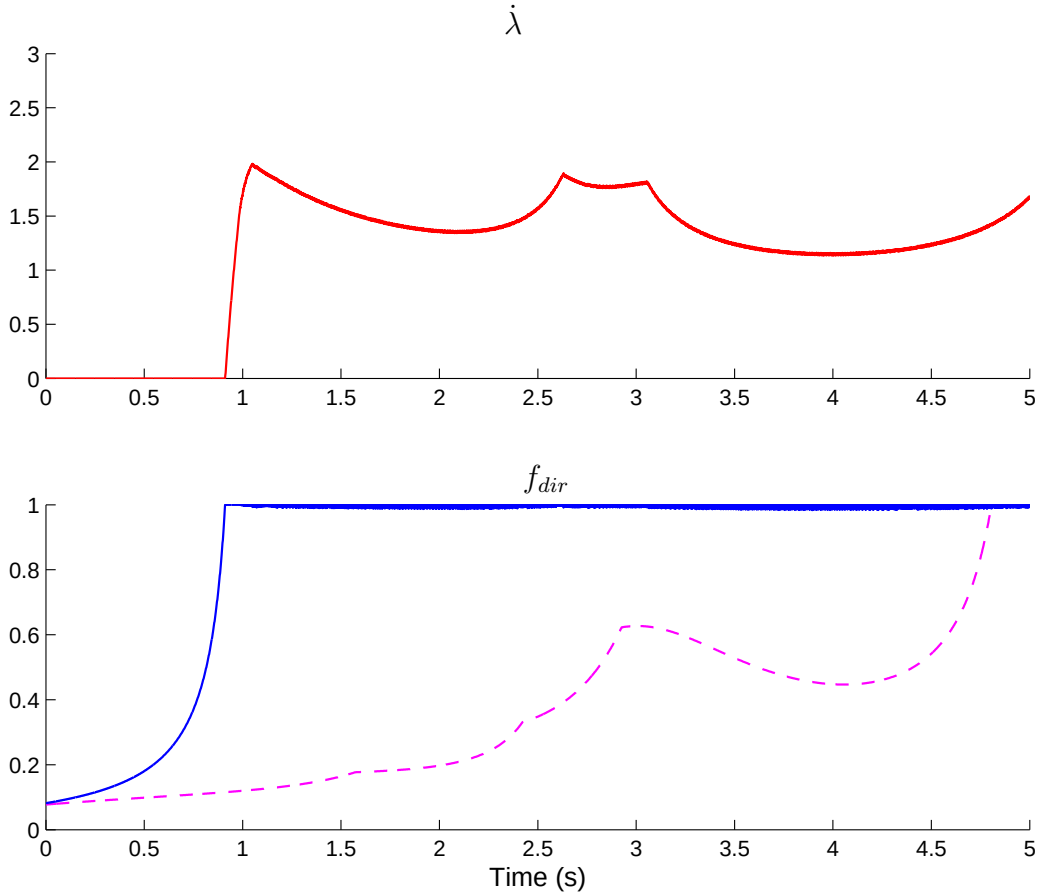

Figure 5: Top box: Speed profile $\dot{\lambda}$ produced by SM auto-regulation. Bottom box: directionality factor f_{dir} with (solid) and without (dashed) SM auto-regulation.

Position Errors	Case 1	Case 2	Case 3	Case 4
$RMS(\{e_{p1}\})$	0.0101	0.0497	0.0523	0.0266
$RMS(\{e_{p2}\})$	0.0101	0.0188	0.0517	0.0266
$RMS(\{e_{p3}\})$	0.0051	0.0834	0.0540	0.0133
$RMS(\{e_{p4}\})$	0.0202	0.1013	0.0642	0.0531
$RMS(\{e_{p5}\})$	0.0202	0.2127	0.0642	0.0531
$RMS(\{e_{p6}\})$	0.0101	0.0722	0.0321	0.0266
$RMS(\{\|\mathbf{e}_p\|_2\})$	0.0339	0.2655	0.1326	0.0891

Table 1: Comparison of the root mean square errors corresponding to the four cases considered in Fig. 3. Case 1: ideal (unconstrained); case 2: direct-saturated; case 3: directionality-preserving saturated; case 4: SM regulation.

how the method enforces at all times at least one joint speed to reach its limit value.

The corresponding motion parameter $\dot{\lambda}$ and directionality factor f_{dir} are depicted in the Fig. 5. In this case, a sampling period T_s of one millisecond and a low-pass filter with a cutoff frequency of 5 rad/s were employed. As shown by equation (23), slower filter bandwidth or faster sampling rates could be used if chattering reduction were aimed. However, with the chosen sampling period and bandwidth, the motion parameter speed (and, hence, position) appear smooth enough in the referred plot: the chattering issues discussed on Subsection 3.3.1 are irrelevant in practice if fast sampling rates are possible.

As discussed on Subsection 3.3.4, the controller gain K_p gives rise to an error band or tolerance inside which the trajectory reference starts moving forwards. In Fig. 5 the effect is noticeable in the initial phase: the trajectory is automatically stopped until $t \approx 0.9$, when the (directionality preserving) saturation movement reaches the vicinity of the starting reference point.

Finally, Table 1 compares the root mean square errors of each coordinate and of the corresponding Euclidean norm for the four cases considered in Fig. 3. As expected, the auto-regulation technique reduces the errors produced by conventional constrained tracking algorithms.

4.3. Simulation for a reference path with a singular point

In order to evaluate the proposal under hard tracking conditions, we consider now an initial robot pose $\mathbf{p}(0) = \mathbf{p}_{ref}(0) = [0.57 \ -0.15 \ 0.07 \ 0 \ \pi/2 \ 0]^{\mathrm{T}}$

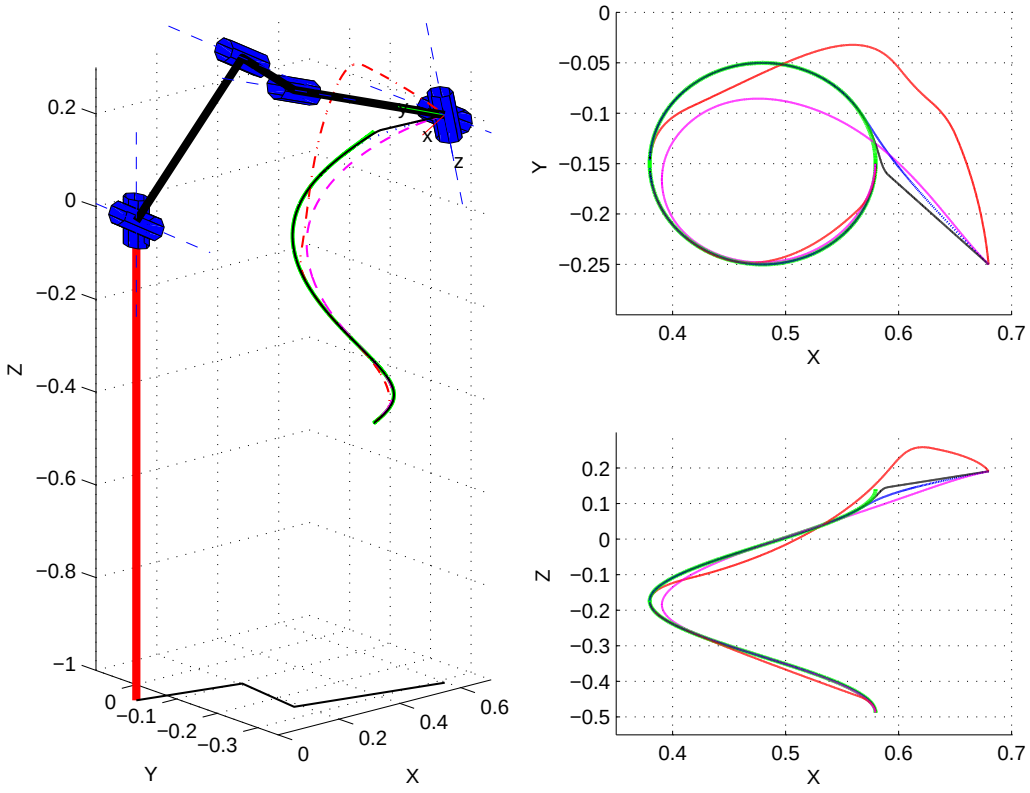

Figure 6: 3D and 2D views of followed paths and desired path.

(zero initial error) and a reference orientation for the robot EE given by:

$$\begin{aligned}
&\alpha_{ref}(\lambda) = 0 \\
&\beta_{ref}(\lambda) = \begin{cases} \pi/2 + \lambda/2 & \text{if} \quad \lambda \leq \pi \\ \pi & \text{if} \quad \lambda > \pi \end{cases} \\
&\gamma_{ref}(\lambda) = \begin{cases} 0 & \text{if} \quad \lambda \leq \pi \\ (\lambda - \pi)/2 & \text{if} \quad \lambda > \pi, \end{cases} \qquad \text{with} \quad \lambda = 0...2\pi,
\end{aligned} \tag{26}$$

for which there is a wrist singularity at $\lambda = \pi$, since the fourth and sixth joints are aligned on the same axis at that point and therefore the determinant of the Jacobian vanishes. Note that the approaching to the wrist singularity is in the non-degenerated direction and the departure from the wrist singularity is in a degenerated direction, i.e. the singularity is of the ordinary type [3].

In this case, the maximum/minimum joint speeds were taken as $\dot{q}_{max,i} = -\dot{q}_{min,i} = 2\,\text{rad/s}$, with $i = 1, \ldots, 6$, in order to allow greater tracking speeds. The constant motion parameter used in the first three cases was then set as $\dot{\lambda} = 3.1$, again for comparative aims; whereas the fourth case has been simulated with $T_s = 0.25$ ms, $\omega_c = 150$ rad/s and $\dot{\lambda}_d = 12$.

Fig. 7 to Fig. 9 show the corresponding results. The following points are worthy of highlighting:

(i) As intuitively expected, in the first unconstrained case the joint speeds of the fourth and sixth joints are extremely large at the singular point (although truncated in Fig. 8, their magnitude orders are given by $1/\epsilon$).

(ii) In conventional constrained cases significant path errors appear since the singular point is crossed (see Fig. 7), and these deviations remain during all the second half-cycle of the helicoidal path.

(iii) In order to prevent the robot from leaving the desired path, the SM auto-regulation technique completely stops the forward movement ($\dot{\lambda} = 0$) at the singular point, and it only restarts moving forward once the 4th and 6th joints have finished their reorientations (Fig. 8 and Fig. 9). Note that with the SM tracking algorithm the singularity is reached earlier than with conventional approaches because of the higher initial tracking speed.

5. Conclusions

A variable structure algorithm for path tracking speed auto-regulation was proposed using sliding mode related concepts. The strategy acts as a supervisory loop, shaping the speed reference along the path so that it is always compatible with joint actuator constraints. In this manner, the

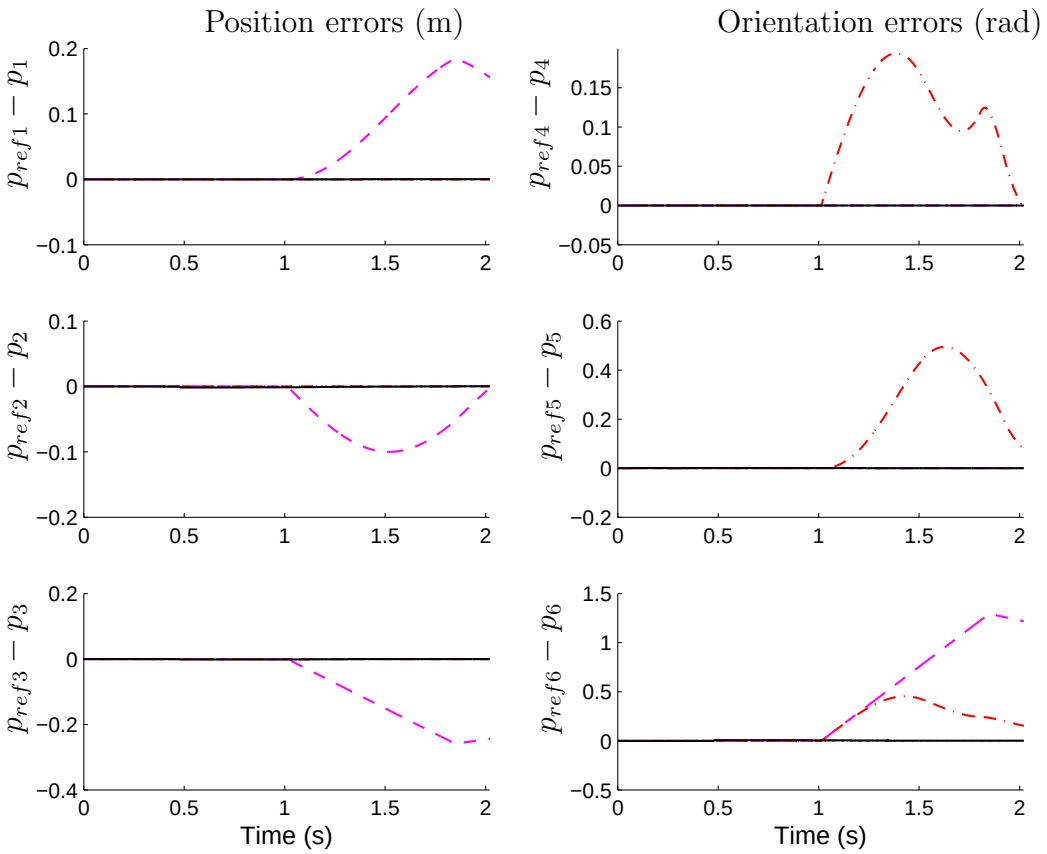

Figure 7: Position and orientation errors when tracking a path including wrist singularity for: ideal unconstrained case (dotted), direct saturation (dashed-dotted), directionality-preserving saturation (dashed) and SM auto-regulation technique (solid).

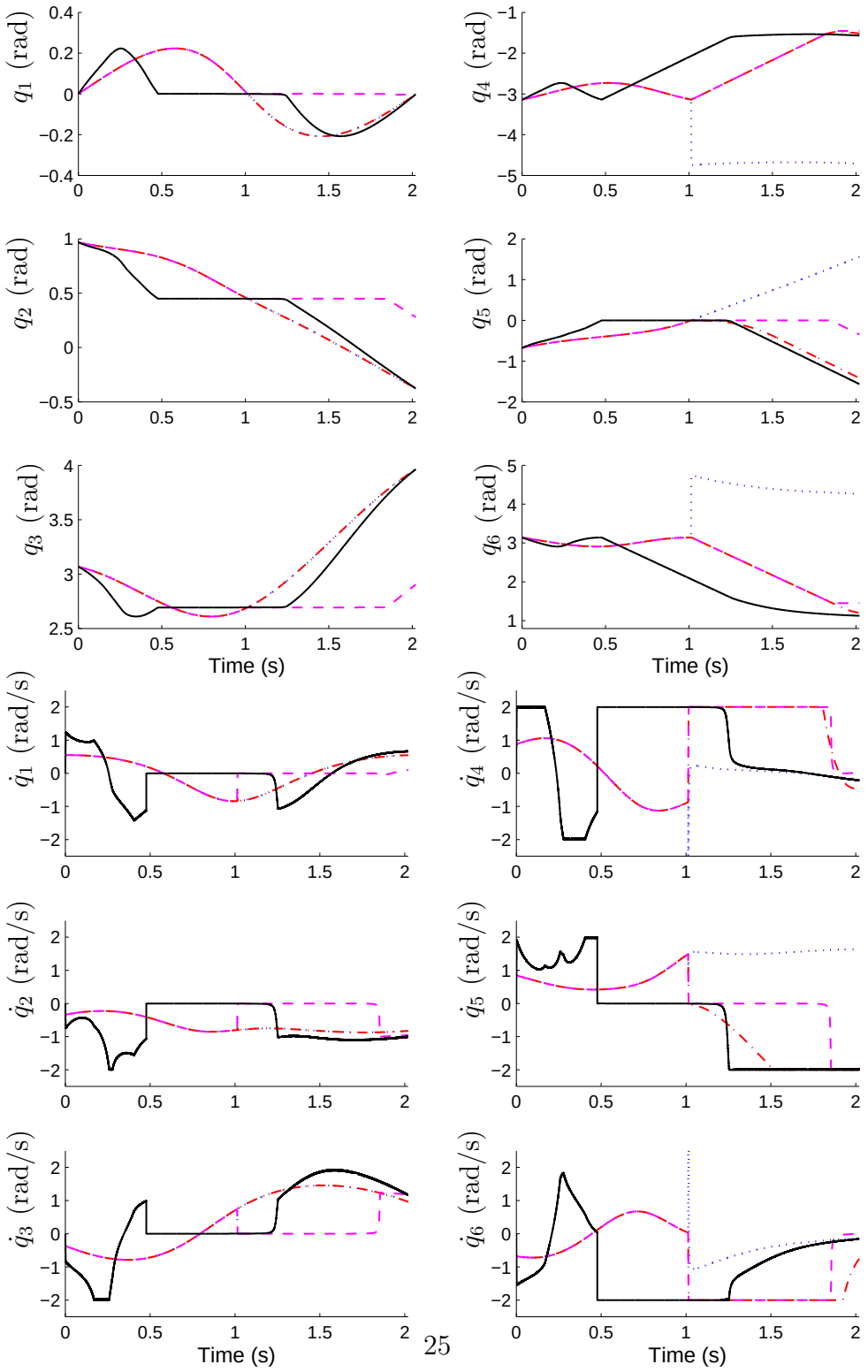

Figure 8: Joint positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$ when tracking a path including wrist singularity for: ideal unconstrained case (dotted), direct saturation (dashed-dotted), directionality-preserving saturation (dashed) and SM auto-regulation technique (solid).

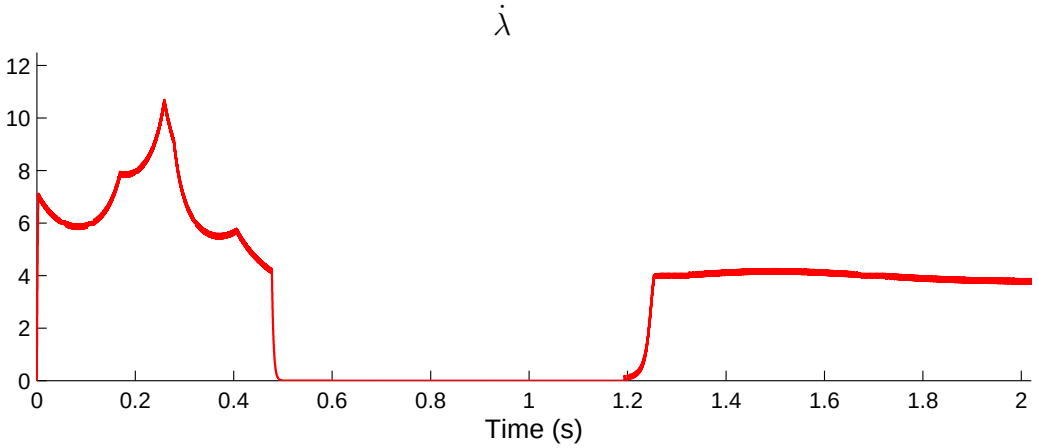

Figure 9: Speed profile $\dot{\lambda}$ produced by SM auto-regulation when tracking a path including wrist singularity.

algorithm does only activate when the nominal speed reference leads a control signal to its maximum value, slowing down as much as necessary in order to avoid path deviations. The proposal can be easily added as an auxiliary loop to conventional robotic path-tracking schemes, and its implementation is extremely easy. Importantly, singularities are gracefully handled.

Although the algorithm was illustrated for a particular kinematic controller and 6R robot, the conclusions drawn for the tracking speed auto-regulation method also apply to any other kinematic controller and/or robotic system accepting joint speed commands. One suggestion for further work would be to extend the proposed technique to include acceleration and/or torque constraints.

Acknowledgements

Research in this area is partially supported by DISICOM project PROMETEO 2008/088 of Generalitat Valenciana (Spain), research project DPI2008-06731-C02-01 of the Spanish Government (Spain), Technical University of Valencia (Spain), National University of La Plata (Argentina) and CONICET (Argentina).

References

[1] J. Pires, Industrial Robots Programming: Building Applications for the Factories of the Future, Springer-Verlag, Berlin, 2007.

[2] Y. Shimon (Editor), Handbook of Industrial Robotics, Wiley, New York, NJ, 1999.

[3] L. Gracia, J. Andres, J. Tornero, Trajectory tracking with a 6R serial industrial robot with ordinary and non-ordinary singularities., International Journal of Control, Automation, and Systems 7 (2009) 85–96.

[4] J. Kieffer, Differential analysis of bifurcations and isolated singularities for robots and mechanisms, IEEE Transactions on Robotics and Automation 10 (1994) 1–10.

[5] D. Nenchev, Tracking manipulator trajectories with ordinary singularities: a null space-based approach, International Journal of Robotics Research 14 (1995) 399–404.

[6] D. Nenchev, M. Uchiyama, Singularity-consistent path planning and motion control through instantaneous self-motion singularities of parallel-link manipulators, Journal of Robotic Systems 14 (1997) 27–36.

[7] L.-J. Huang, M. Tomizuka, A self-paced fuzzy tracking controller for two-dimensional motion control, IEEE Transactions on Systems, Man and Cybernetics 20 (1990) 1115–1124.

[8] A. Bemporad, T.-J. Tarn, N. Xi, Predictive path parameterization for constrained robot control, IEEE Transactions on Control Systems Technology 7 (1999) 648–656.

[9] G. Antonelli, S. Chiaverini, G. Fusco, A new on-line algorithm for inverse kinematics of robot manipulators ensuring path tracking capability under joint limits, IEEE Transactions on Robotics and Automation 19 (2003) 162–167.

[10] S. Ma, A kinetostatic index to measure the task-executing ability of robotic manipulators with limit-driven characteristics of actuators, Advanced Robotics 18 (2004) 401–414.

[11] R. Skjetne, T. Fossen, P. Kokotovic, Robust output maneuvering for a class of nonlinear systems, Automatica 40 (2004) 373 – 383.

[12] F. Garelli, R. Mantz, H. De Battista, Limiting interactions in decentralized control of MIMO systems, Journal of Process Control 16 (2006) 473–483.

[13] F. Garelli, R. Mantz, H. De Battista, Sliding mode reference conditioning to preserve decoupling of stable systems, Chemical Engineering Science 62 (2007) 4705–4716.

[14] K. Gupta, Mechanics and Control of Robots, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[15] W. Khalil, E. Dombre, Modeling, Identification and Control of Robots, Taylor & Francis, Inc., Bristol, PA, USA, 2002.

[16] J. Angeles, Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms, Springer-Verlag, New York, NJ, 3rd. edition, 2007.

[17] R. Kelly, V. Santibáñez, A. Loría, Control of Robot Manipulators in Joint Space, Springer-Verlag, London, UK, 2005.

[18] C. Edwards, S. K. Spurgeon, Sliding Mode Control: Theory and Applications, Taylor & Francis, UK, 1st edition.

[19] J. Hung, W. Gao, J. Hung, Variable structure control: a survey, IEEE Transactions on Industrial Electronics 40 (1993) 2–22.

[20] V. Utkin, J. Guldner, J. Shi, Sliding Mode Control in Electromechanical Systems, Taylor & Francis, London, 1st edition.

[21] W. Bessa, M. Dutra, E. Kreuzer, An adaptive fuzzy sliding mode controller for remotely operated underwater vehicles, Robotics and Autonomous Systems 58 (2010) 16 – 26.

[22] C. Chen, T. Li, Y. Yeh, C. Chang, Design and implementation of an adaptive sliding-mode dynamic controller for wheeled mobile robots, Mechatronics 19 (2009) 156 – 166.

[23] L. García-Valdovinos, V. Parra-Vega, M. Arteaga, Observer-based sliding mode impedance control of bilateral teleoperation under constant unknown time delay, Robotics and Autonomous Systems 55 (2007) 609 – 617.

[24] J. Shi, H. Liu, N. Bajcinca, Robust control of robotic manipulators based on integral sliding mode, International Journal of Control 81 (2008) 1537–1548.

[25] M. Park, K. Kim, Chattering reduction in the position control of induction motor using the sliding mode, IEEE Transactions on Power Electronics 6 (1991) 317–125.

[26] P. Corke, A robotics toolbox for matlab, IEEE Robotics and Automation Magazine 3 (1996) 24–32.