Robotica (2011) volume 29, pp. 403–420. © Cambridge University Press 2010
doi:10.1017/S026357471000024X

Constrained motion planning for open-chain industrial robots

Gianluca Antonelli†*, Cataldo Curatella‡ and Alessandro Marino§

†*DAEIMI, Università degli Studi di Cassino, Via G. Di Biasio 43, 03043 Cassino (FR), Italy*
‡*Comau S.p.A., via Rivalta 30, 10095 Grugliasco, Torino, Italy. E-mail: cataldo.curatella@comau.com*
§*DIFA, Università degli Studi della Basilicata, Viale dell'Ateneo Lucano 10, 85100 Potenza, Italy.*
E-mail: alessandro.marino@unibas.it

(Received in Final Form: June 1, 2010. First published online: July 7, 2010)

SUMMARY

In the industrial environment, several constraints affect the robot motion planning. These are imposed by manufacturing considerations, such as, e.g., to strictly follow a given path, or by physical constraints, such as, e.g., to avoid torque saturation. Among the others, limitation on the velocity, acceleration, and jerk at the joints is often required by the robot manufacturers. In this paper, a motion planning algorithm for open-chain robot manipulators that takes into account several constraints simultaneously is presented. The algorithm developed approaches the motion planning algorithm from a wide perspective, solving systematically the joint as well as the Cartesian motion, both for the point-to-point and the fly movements. The validation has been performed first by numerical simulations and then by experiments on two different industrial manipulators, with different size, with and without the presence of a payload, by imposing demanding trajectories where all the constraints have been excited.

1. Introduction

Constrained motion for robot manipulators has been intensively studied from the researchers due to its crucial importance in robotics. This is particularly true in industrial production lines, where the manipulators generally continuously repeat cycles of welding, gluing, cutting, etc. and it is important to exploit the machine at its maximum to preserve the traveling time. For several reasons the manipulator's motion needs to be constrained, rough examples are given by the torque constraint at each joint and the path constraint in the Cartesian space. It is well known that fulfillment of several constraints at the same time is not obvious in the sense that, e.g., a brutal saturation on the torques generally violate the path constraint. Therefore, the motion trajectory should be carefully designed. In practical applications, in addition, it is often required the manipulator to go from an initial to a final configuration (in joint or Cartesian space) *flying* through an intermediate point (called via-point).

A via-point can be defined as an intermediate point that the trajectory passes close to, but does not necessarily pass through.[22] The last feature is required to avoid the manipulator stopping at the via-point; in fact, passing exactly through it would generally require the manipulator to stop to avoid velocity discontinuity, then a path modification is allowed in its proximity.

The path preservation is generally guaranteed by using the curvilinear abscissa as a parameter function of the geometric path. In ref. [4], this idea is used to express the dynamical model as function of the curvilinear abscissa and to build a proper two-dimensional phase space where a procedure gives the minimum-time optimal trajectory; only torque and velocity limits are taken into account and the proposed solution is iterative. Within the same framework it is worth noticing the works, such as refs. [23, 24].

Interesting works are represented by refs. [6–8] where only the torque and path constraints are considered. In few words, a secondary controller modifies the nominal trajectory during the motion to cope with the modeling uncertainties. The output of these methods are bang–bang-like trajectories that achieve the torque limit at least in one joint at every time instant, thus allowing to exploit the actuators at their maximum performance. A similar algorithm, solving the same constraints, is presented in ref. [17]. Details on these works are given in Section 4.

Very few papers take into account the jerk's characteristics. This is, however, very important for the mechanical preservation of the motor gear.[5,14,21] Among them, a very nice work is presented in ref. [14]; here the trajectory is designed by properly building concatenated quintic polynomials. The obtained trajectory is continuously up to the acceleration, moreover, the jerk discontinuity is very small and limited to the 10% of the maximum value. Finally, the velocity profile generated with this algorithm is *trapezoidal-like*, i.e., a suboptimum with respect to the optimal minimum-time trajectory. In ref. [9], an objective function for simultaneous minimization of jerk and execution time allows to get smooth trajectory avoiding an excessive duration. In ref. [11], the solutions for point-to-point minimum-jerk trajectories are analytically obtained; the optimization, obtained through the Pontryagin principle, concerns two objective functions, namely the time integral of the squared jerk and the maximum absolute value of the jerk.

Nakamura *et al.*[18] suggested a minimum jerk trajectory planner in the Cartesian space; in the case of fly movements an arc of a circle is used to join two consecutive straight lines. In order to counteract the curvature discontinuities at

* Corresponding author. E-mail: antonelli@unicas.it

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

the joining points, cubic interpolation techniques are used, leading to an undefined displacement with respect to the via-point.

With regards to the Cartesian space, in ref. [12], a free obstacle Cartesian space path is defined together with a time-optimal motion law and torque constraint for a simple SCARA manipulator. In ref. [15], an iterative learning control algorithm is used for the problem of path tracking rather than trajectory tracking in Cartesian space, and where no previous model of the manipulator is required. The solution proposed requires that the manipulator, repeatedly, executes the given path to learn how to execute it properly. The effectiveness of the algorithm is related to particular paths and manipulators, and maximum velocity along the path cannot be specified. In ref. [10], the analytical solution to the inverse kinematic problem of a three degrees of freedom (DOF) manipulator is used to lead to Cartesian acceleration and velocity limits starting from the joint torque and velocity constraints.

In ref. [5], also the time derivative of the torque is constrained. The mathematical model is written with respect to the third-order derivative of the curvilinear abscissa and solved with respect to the pseudo-jerk input.

The literature can be basically divided into off-line and on-line algorithms; in off-line approaches the trajectory is computed before the movement starts, whereas in on-line approaches it is modified during the robot motion based on the current sensor readings. One common drawback of both off-line and on-line above methods is that the obtained velocity profile of the optimal trajectory does not have a predictable behavior, which is not always allowable in industrial tasks, e.g., glue spreading or painting. In addition, on-line algorithm without any kind of knowledge of the future trajectory cannot guarantee fulfillment of the path constraints as it will be discussed in Section 4.

In this paper, a constrained motion planning algorithm for open-chain robot manipulators is presented. The list of the considered constraints is given in Section 2 and, despite the effort devoted at solving the constrained motion planning, at the best of the authors' knowledge, none of the papers in the literature addresses all the constraints addressed here simultaneously or, on the other hand, can be easily adapted or extended to include additional constraints.

The algorithm developed approaches the motion planning algorithm from a wide perspective, solving the joint as well as the Cartesian motion, both for the point-to-point and the fly movements. The submitted algorithm represents a significant improvement over a former work;[3] the list of constraints and the improvement with respect to ref. [3] are punctually discussed in Section 2. The validation has been extremely severe in view of a scheduled industrial implementation and concerned both numerical simulation and experiments on two industrial manipulators using their current control architecture and constraints values, with and without a known payload. The results confirm the achievement of the goals.

2. Constraints

In this section, the considered constraints are listed; obviously all the constraints need to be fulfilled simultaneously.

Common constraints for the joint and Cartesian spaces are:

- Smoothness. The trajectory needs to be of class $\mathcal{C}^2$, i.e., continuous up to the acceleration included;
- Limits on the absolute values of the joints torques, velocities, accelerations, and jerks;
- Possibility to generate both a point-to-point as well as fly motions;
- Possibility to switch between the joint and Cartesian commands without stopping the robot;
- Joint and Cartesian path must be always preserved;
- The trajectory needs to be *modular*, in the sense that it is possible to add new points for a multifly movement without modifying the already generated trajectory;
- Known computation load, i.e., avoid iterative or learning solutions.

Constraints specific to the joint space:

- *Optimal* trajectory. The given trajectory will be a *sub*optimum of the minimum time, bang–bang, movement and at least one constraint will be saturated during the motion.

Constraints specific to the Cartesian space:

- Given cruise velocity. In the point-to-point motion, the user may impose an end-effector cruise velocity or ask the algorithm to compute the maximum velocity compatible with the remaining constraints;
- There must be a biunivocal relationship between position and orientation for a given path and this relation must be the same independently from the chosen time law. In other words, for a given path at a given position must correspond always the same orientation despite the chosen cruise velocity and without memory of former command execution.

The last constraint needs few word of explanation. In industrial environment it is common that the end-user programs trajectory interactively by resorting to a *teach-pendant*. She/he firstly programs the path by moving the manipulator at low cruise velocity and then run it asking for its maximum performance, i.e., saturating one or more constraints.

In detail, assuming a six joint manipulator, the following limits will be assumed:

$$\begin{aligned}
q_{i,\min} \le q_i \le q_{i,\max}, &\quad i = 1, 2, \ldots 6, \\
-\dot{q}_{i,\max} \le \dot{q}_i \le \dot{q}_{i,\max}, &\quad i = 1, 2, \ldots 6, \\
-\ddot{q}_{i,\max} \le \ddot{q}_i \le \ddot{q}_{i,\max}, &\quad i = 1, 2, \ldots 6, \\
-\dddot{q}_{i,\max} \le \dddot{q}_i \le \dddot{q}_{i,\max}, &\quad i = 1, 2, \ldots 6, \\
-\tau_{i,\max} \le \tau_i \le \tau_{i,\max}, &\quad i = 1, 2, \ldots 6;
\end{aligned} \tag{1}$$

in the following, the first four groups of constraints will be denoted as the kinematic constraints, while the last relation will be referred as the dynamic constraint.

However, while velocity, acceleration, and torque limits are based on data furnished by actuator and gearbox manufacturer, nonstandard procedure exists to calculate jerk limits. Therefore, the joint jerk limits can be heuristically based on an estimate of the time it should take to ramp up to the maximum joint acceleration.[13] With respect to ref. [3],

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

several constraints have been added, the most significant are the requirement of smoothness, the presence of a jerk and acceleration limit on the joints, the possibility to safely mix joint and Cartesian commands and the possibility to impose a maximum distance from the via-points. In addition, in ref. [3] the fly in joint space is handled in a different and less general way, suitable only under the made assumptions and that cannot be extended to handle the new constraints (see Section 5.2). The Cartesian space has been only marginally resolved, i.e., only point-to-point motions have been considered and with a less efficient scaling technique without the possibility to calculate the maximum allowed velocity (see Section 5.4). Finally, the fly motion in Cartesian space and the maximum distance constraint are not at all addressed. The additional research work with respect to ref. [3] has been required by the continuous evolution of the industrial manufacturing environment. At the best of the authors' knowledge, neither ref. [3] nor the papers available in the literature (see Section 4) satisfy *simultaneously* the imposed constraints or can be easily customized.

3. Mathematical Background

Without loss of generality, in the following we will refer to the common case of a six-joint open-chain manipulator; nevertheless the proposed approach is independent of the degrees of mobility of the system.

The dynamic equation of an open-chain 6-DOF manipulator can be written as:[22]

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{F}\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}, \tag{2}$$

where the vector $\boldsymbol{q} \in \mathbb{R}^{6\times 1}$ ($\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$) is the joint position (velocity and acceleration), $\boldsymbol{\tau} \in \mathbb{R}^{6\times 1}$ is the torque, $\boldsymbol{B}(\boldsymbol{q}) \in \mathbb{R}^{6\times 6}$ is the symmetric and positive definite inertia matrix, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^{6\times 1}$ is the vector of gravity terms, $\boldsymbol{F} \in \mathbb{R}^{6\times 6}$ is a diagonal matrix containing the viscous friction coefficients, and $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{6\times 1}$ is a vector containing the centripetal and Coriolis term.

Let us consider a path in the parameterized form

$$\boldsymbol{q}(s) = \boldsymbol{f}(s), \tag{3}$$

where $\boldsymbol{f}(s) \in \mathbb{R}^{6\times 1}$ and s is the curvilinear abscissa, i.e., a scalar path parameter characterized by a strictly increasing time function such as

$$s(t) : [t_0, t_f] \subset \mathbb{R} \rightarrow [s_0, s_f], \tag{4}$$

where t_0 and t_f are the initial and final time instant, s_0 and s_f are initial and final curvilinear abscissa values corresponding to $\boldsymbol{q}_0$ and $\boldsymbol{q}_f$, respectively. In the following, we will assume, without loss of generality, that $s_0 = 0$ and $s_f = 1$. Moreover, the smoothness constraint implies that the function $s(t)$ is of class $\mathcal{C}^2$.

It is well known that Eqs. (2) and (3) allow to write the dynamic model in term of the curvilinear abscissa.[7] In fact, the joint position, velocity and acceleration vectors assume the following form in term of the path parameter s:

$$\boldsymbol{q}(t) = \boldsymbol{f}(s(t)), \tag{5}$$

$$\dot{\boldsymbol{q}}(t) = \frac{\partial \boldsymbol{f}(s)}{\partial s}\dot{s}(t) = \boldsymbol{f}'(s)\dot{s}, \tag{6}$$

$$\ddot{\boldsymbol{q}}(t) = \frac{\partial^2 \boldsymbol{f}(s)}{\partial s^2}\dot{s}(t)^2 + \frac{\partial \boldsymbol{f}(s)}{\partial s}\ddot{s}(t) = \boldsymbol{f}''(s)\dot{s}^2 + \boldsymbol{f}'(s)\ddot{s}. \tag{7}$$

Taking into account Eqs. (5)–(7), model (2) can be rewritten as

$$\tau = \boldsymbol{b}_3(s)\ddot{s} + \boldsymbol{b}_2(s)\dot{s}^2 + \boldsymbol{b}_1(s)\dot{s} + \boldsymbol{b}_0(s), \tag{8}$$

where

$$\begin{aligned} \boldsymbol{b}_3(s) &= \boldsymbol{B}(\boldsymbol{f}(s))\boldsymbol{f}'(s), \\ \boldsymbol{b}_2(s) &= \boldsymbol{B}(\boldsymbol{f}(s))\boldsymbol{f}''(s) + \boldsymbol{c}(\boldsymbol{f}(s), \boldsymbol{f}'(s)), \\ \boldsymbol{b}_1(s) &= \boldsymbol{F}\boldsymbol{f}'(s), \\ \boldsymbol{b}_0(s) &= \boldsymbol{g}(\boldsymbol{f}(s)). \end{aligned} \tag{9}$$

Equation (8) is the manipulator model along the path $\boldsymbol{f}(s)$ expressed as form of the only scalar parameter s. The advantage of such a representation is that it allows the mapping of the torque constraint on constraints on $\ddot{s}$ as will be shown in Section 5.

3.1. The trajectory parametrization

As common in motion planning for robots,[22] the trajectory is parameterized by resorting to the curvilinear abscissa $s(t)$ such that

$$\begin{cases} \boldsymbol{q} = \boldsymbol{f}(s(t)) \text{ joint space}, \\ \boldsymbol{x} = \boldsymbol{h}(s(t)) \text{ Cartesian space}, \end{cases} \tag{10}$$

where $\boldsymbol{x}$ is the manipulator end-effector pose, i.e., $\boldsymbol{x} = [\boldsymbol{p}^{\mathrm{T}}\ \boldsymbol{\phi}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{6\times 1}$ being the end-effector orientation expressed by means of some Euler's angle representation $\boldsymbol{\phi} \in \mathbb{R}^{3\times 1}$. In addition, the algorithm does not depend on the specific representation of the orientation. In fact, in the point-to-point motion choosing the Euler's angle, the axis-angle, the unitary quaternion, or any other representation, the evolution from the initial to the final orientation can be expressed as function of a scalar parameter, that is, the curvilinear abscissa s. However, as will be better described in Section 5, the proposed scaling approach acts only on the time law of the scalar s.

The mapping from the joint constraints to the curvilinear abscissa time law is influenced by the specific functions $\boldsymbol{f}$ and $\boldsymbol{h}$ considered and must be properly taken into consideration. In case of a Cartesian trajectory, moreover, the direct kinematics $\boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q})$, with its time derivatives, is also of concern. With regards to the point-to-point motions, in the following only segments have been considered in the joint space, and segments and circle's arc in the Cartesian space. Their analytical expressions with derivatives are well known and can be found, for example, in ref. [22].

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

The mapping from the Cartesian to the joint space needs also to be taken into consideration

$$\begin{cases} \boldsymbol{x} &= \boldsymbol{k}(\boldsymbol{q}), \\ \dot{\boldsymbol{x}} &= \dfrac{\partial \boldsymbol{k}(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \\ \ddot{\boldsymbol{x}} &= \dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}}, \\ \dddot{\boldsymbol{x}} &= \ddot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}} + 2\dot{\boldsymbol{J}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{J}(\boldsymbol{q})\dddot{\boldsymbol{q}}, \end{cases} \tag{11}$$

where $\boldsymbol{J}(\boldsymbol{q}) \in \mathbb{R}^{6\times 6}$ is the Jacobian matrix for a 6-DOF manipulator with six degrees of mobility;[22] Eq. (11) lead to a nonlinear, configuration dependent, mapping from s to $\boldsymbol{q}$.

3.2. The curvilinear abscissa time law

The class of functions to be used as basis for time law of $s(t)$ has been inherited from the seminal work of ref. [14]. In a few words, proper concatenation of fifth-order polynomials is used in order to provide a smooth trajectory between two-way points while preserving a trapezoidal-like velocity profile. The solution proposed is interesting even if its generalization to a wider set of constraints is not developed. The work[14] also proposes a procedure to handle Cartesian fly movements between two segments using sixth-order polynomials; it is not possible, however, to guarantee a constant cruise velocity and a maximum distance from the via-point. In detail, only a way to generate a smooth curvilinear abscissa is carried out. This abscissa can be used to generate motion in the joint or in the Cartesian space, but it is not explained how to map the kinematic and dynamic constraints of the manipulator in limits of the curvilinear abscissa. In addition, the fly movements are only slightly considered. The proposed algorithm inherits from ref. [14] the concept of quintic polynomials as basis functions to build the trajectory before its eventual scaling and, in this section, the main properties of the *concatenated quintics* are briefly illustrated.

Figure 1 reports the trajectory connecting to generic points: s_0 and s_f. It can be noted that the concatenation of quintics has been properly designed so as to approximate a trapezoidal-like trajectory while at the same time, providing a function of class $\mathcal{C}^2$.

The presence of concatenate quintic polynomials can be appreciated in Fig. 2 in which each interval individuated by a large (red) dot is build by a fifth-order polynomial.

The work[14] gives the equations needed to properly design the trajectory and explains how to define the intermediate points to be connected by the quintic polynomials.

In fact, with regards to the last aspects, as described in ref. [14], the concatenate quintic polynomials use a *sine wave template* to accomplish change in acceleration and proper initial and final conditions at each quintic polynomial to guarantee continuity.

It is demonstrated in the Appendix of ref. [13], that such an approach saturates the jerk during the acceleration and deceleration phase at the 98% of the input values. Proper higher jerk bounds can be used to overcome such a drawback. In addition, it is also demonstrated that for a trajectory formed of a concatenation of quintics, the worst discontinuity in jerk will occur during an acceleration pulse (a concatenation of two acceleration ramps) and that the magnitude of this discontinuity will be equal to 10% of the jerk limit.

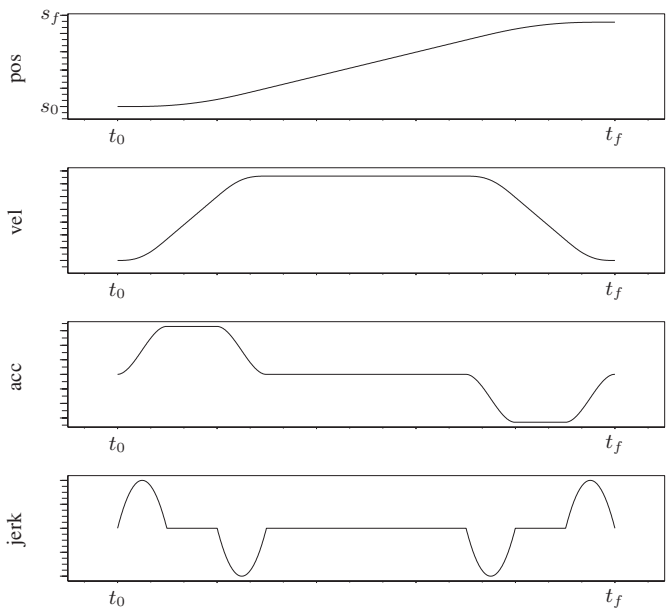


Fig. 1. Example of an overall trajectory obtained by properly concatenating fifth-order polynomials.

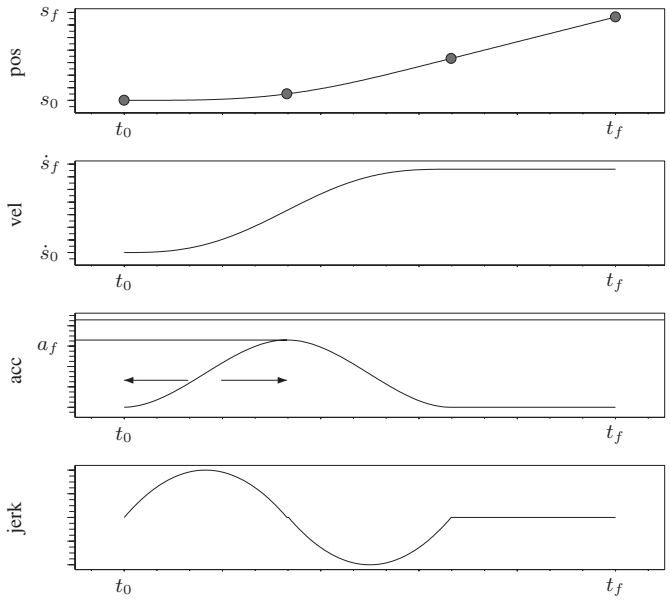


Fig. 2. Smooth semitrajectory generated with a sequence of quintics.

Moreover, our experimental results confirmed the increased performance of the controllers when using an *almost* $\mathcal{C}^3$ trajectory, in particular, the traveling time of the effective controlled variable is shorter with respect to a desired trajectory with the same kinematic limits but not limited in jerk.

Another advantage, not exploited in ref. [14], in using the fifth-order polynomials is given by the possibility to analytically *scale* the trajectory by resorting to a scalar factor

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

without the need to recompute it. Overall, the trajectory is fully defined by imposing the following quintuple $\{\dot{s}_{\max},\ \ddot{s}_{\text{acc,max}}, \dddot{s}_{\text{acc,max}}, \ddot{s}_{\text{dec,max}}, \dddot{s}_{\text{dec,max}}\}$:

- $\dot{s}_{\max}$: maximum cruise velocity;
- $\ddot{s}_{\text{acc,max}}$: maximum acceleration during the acceleration phase;
- $\dddot{s}_{\text{acc,max}}$: maximum jerk during the acceleration phase;
- $\ddot{s}_{\text{dec,max}}$: maximum acceleration during the deceleration phase;
- $\dddot{s}_{\text{dec,max}}$: maximum jerk during the deceleration phase.

4. Literature Comparison

The algorithm proposed in ref. [7] represents one of the reference papers in the literature among the on-line scaling algorithms. In ref. [7], however, the sole torque constraint at joints is considered, in addition, some drawbacks prevent its practical implementation.

As described in the following of this section, it is not possible to guarantee that the generated trajectory does not overshoot the final desired configuration. The same happens in the similar work.[17] The *overshoot* drawback is not peculiar to refs. [7, 17] but it is shared by most of the on-line algorithms. A simple automotive equivalent may be of help in understanding this aspect: let us imagine a car at high, constant velocity that suddenly encounters an *unknown* bend. Depending on the relative values between the bend's curvature and the car's constraints, it is not guaranteed that the car stays on the path during the bend. The same concept applies to the end effector of the manipulator and it is the main reason why an off-line approach has been considered for the proposed algorithm. In fact, when the algorithm computes a constraint's violation it has the temporal and computational capability to safely recompute the trajectory.

To understand the latter aspect, it is useful to show the key issues of the approach in ref. [7].

Let us consider the following trajectory tracking controller:

$$\boldsymbol{\tau} = \widehat{\boldsymbol{B}}(\boldsymbol{q})[\ddot{\boldsymbol{q}}_d + \boldsymbol{K}_v\dot{\boldsymbol{e}} + \boldsymbol{K}_p\boldsymbol{e}] + \widehat{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \widehat{\boldsymbol{F}}\dot{\boldsymbol{q}} + \widehat{\boldsymbol{g}}(\boldsymbol{q}), \tag{12}$$

where $\boldsymbol{K}_v, \boldsymbol{K}_p \in \mathbb{R}^{6\times 6}$ are symmetric positive definite matrices, the symbol $\widehat{\ }$ denotes available estimates of the corresponding quantities, and the vector $\boldsymbol{e}$ is the error defined as

$$\boldsymbol{e} = \boldsymbol{q}_d - \boldsymbol{q}, \tag{13}$$

being $\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d, \ddot{\boldsymbol{q}}_d \in \mathbb{R}^{6\times 1}$ the desired joint trajectories.

In order to retrieve minimum and maximum acceleration of the curvilinear abscissa along the path ($\ddot{s}_{\min}$, $\ddot{s}_{\max}$) a parameterized form of the primary controller needs to be computed by substituting Eqs. (5)–(7) in Eq. (12):

$$\boldsymbol{\tau} = \boldsymbol{a}_1\ddot{s} + \boldsymbol{a}_2, \tag{14}$$

where $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are 6×1 vectors depending on the measured quantities $\boldsymbol{q}, \dot{\boldsymbol{q}}$, (see ref. [8] for their explicit expressions). Nevertheless, other forms for the controller can be used without loss generality. The on-line bounds of $\ddot{s}$ are related to constant bounds on the input torques via Eq. (14) in such a way that

$$\tau_{i,\min} \le \tau_i = a_{1,i}\ddot{s} + a_{2,i} \le \tau_{i,\max}, \quad i = 1\ldots 6. \tag{15}$$

For the ith joint the upper and lower bounds on $\ddot{s}$ can be found as

$$\ddot{s}_{i,\max} = \begin{cases} (\tau_{i,\max} - a_{2,i})/a_{1,i}, & a_{1,i} > 0, \\ (\tau_{i,\min} - a_{2,i})/a_{1,i}, & a_{1,i} < 0, \\ \infty, & a_{1,i} = 0 \end{cases} \tag{16}$$

and

$$\ddot{s}_{i,\min} = \begin{cases} (\tau_{i,\min} - a_{2,i})/a_{1,i}, & a_{1,i} > 0, \\ (\tau_{i,\max} - a_{2,i})/a_{1,i}, & a_{1,i} < 0, \\ -\infty, & a_{1,i} = 0. \end{cases} \tag{17}$$

Therefore, the bounds on $\ddot{s}$ are the followings

$$\begin{aligned} \ddot{s}_{\max} &= \min_i s_{i,\max} \\ \ddot{s}_{\min} &= \max_i s_{i,\min}. \end{aligned} \tag{18}$$

Given a nominal trajectory $s_n(t)$, the corresponding joint trajectory is given by Eq. (5). Trajectory scaling is obtained by an outer control loop whose output is a scaled function $s(t)$, with its first and second derivatives, according to Eq. (18). The output trajectory is the input for the inner control loop, represented by Eq. (14). The scaled path acceleration $\ddot{s}$ is computed, according to ref. [8], as

$$\ddot{s}_r = \beta\ddot{s}_n(s) + \frac{1}{2}\alpha\left(\dot{s}_n^2(s) - \dot{s}^2\right), \tag{19}$$

$$\ddot{s} = \text{sat}\left(\ddot{s}_r, \ddot{s}_{\min}, \ddot{s}_{\max}\right), \tag{20}$$

where $\ddot{s}_r$ is a filter variable, α is a positive gain, β is a parameter that can be equal to 0 or 1 (see ref. [8]), and sat is the saturation function with bounds $\ddot{s}_{\min}$ and $\ddot{s}_{\max}$.

In the practical implementation of Eqs. (19)–(20) some problems arise. First, $\ddot{s}_n(s)$ and $\dot{s}_n(s)$ can be difficult or even impossible to compute in the case of nominal trajectories planned by using high-order polynomials, spline functions[14] or NURBS (Non-Uniform Rational B-Spline) curves.[20]

Moreover, in the presence of $\mathcal{C}^2$-trajectories (as in ref. [14]), $\ddot{s}_n(s_0)$, $\dot{s}_n(s_0)$, and $\dot{s}(t_0)$ are identically null in Eq. (19). Therefore, neither $\ddot{s}$, $\dot{s}$, and s, without an artificial initial perturbation, will change their initial values, resulting in a constant reference trajectory for the control loop equal to $\boldsymbol{q}_0$. Finally, in almost all the industrial applications, it is crucial to reach exactly the final point of the nominal trajectory, i.e., $\boldsymbol{q}_f$, without overshoots, ($s(t) \le s_f\ \forall\, t$). Equations (19)–(20) do not guarantee this property. One example is shown in Fig. 3, obtained by a numerical simulation where the nominal trajectory is beyond the torque limits, and the overshoot is equal to 3%, a not tolerable value in realistic industrial applications. The same effect seems to be present in the simulation case studies presented in ref. [16].

A similar algorithm is published in ref. [17]. The parametrization of the controller is the same as in ref. [7],

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

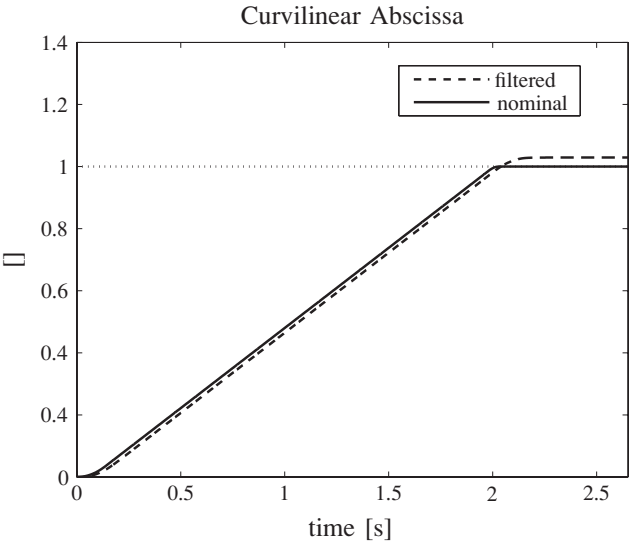


Fig. 3. Nominal and filtered curvilinear abscissa in Dahl–Nielsen's schema, example of overshoot.

and an on-line filter is in charge of scaling the trajectory according to the following equation:

$$\bar{s}(t) = c(t)s_n(t), \quad c(t) > 0 \ \forall t, \tag{21}$$

where $c(t)$ is a scaling factor. The time update law is:

$$\begin{cases} \widetilde{c} = r(t) - c(t), \\ u = \ddot{r} + k_{vc}\dot{\widetilde{c}} + k_{pc}\widetilde{c}, \\ \ddot{c} = \text{sat}(u, \ddot{c}_{\min}, \ddot{c}_{\max}), \end{cases} \tag{22}$$

where $\ddot{c}_{\min}$ and $\ddot{c}_{\max}$ are, respectively, the minimum and maximum allowed bounds on the second derivative of coefficient c, calculated considering the manipulator torque limits; k_{pc} and k_{vc} are positive gains.

Two major drawbacks affect this solution. Due to the scaling factor dynamic (see ref. [17]), if the torque exceeds the given bounds in a single point of the path, the scaling factor c becomes smaller than one and all the trajectory will be scaled, with obvious performance degradation. Moreover, taking into account boundedness of $s(t)$, $c < 1$, implies that $s(t) < s_f$, i.e., the final joints position $\boldsymbol{q}_f$ will never be reached. For this reason, the same author in ref. [16] proposes a different dynamic filter for the scaling factor:

$$\begin{cases} u = k_{vc}\dot{c} + k_{pc}(1 - c), \\ \ddot{c} = \text{sat}(u, \ddot{c}_{\min}, \ddot{c}_{\max}), \end{cases} \tag{23}$$

with k_{pc} and k_{vc} again positive gains. However, it is not clear how to properly choose k_{pc} and k_{vc} and, also in this case, it can be shown by numerical examples that overshoots of final configurations arise.

For all the algorithms reviewed in this section, the main difference with respect to the proposed algorithm, besides the analytical ones, is given by the number of constraints to be satisfied. As stated in the introduction, refs. [5, 7, 14, 17], in fact, consider only a subset of the constraints given in our problem and their generalization is not trivial.

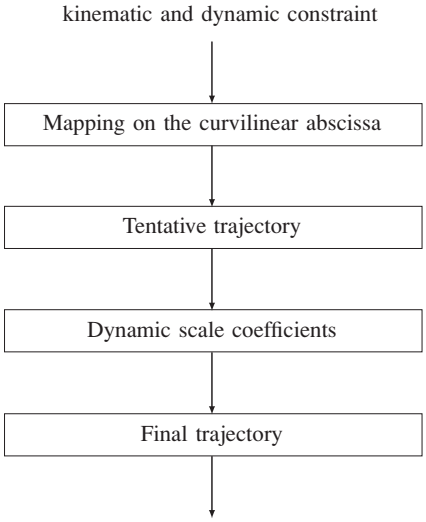


Fig. 4. Scheme of the trajectory planner for point-to-point joint motions.

5. Proposed Approach

In this section, an approach that fulfills all the constraints discussed in Section 2 is presented. For the reasons explained above, the algorithm runs off-line and it does not require any modification to the existing industrial controller, since it provides it the joint desired positions; the absence of iterative computations, moreover, allows its implementation by allocating a deterministic amount of computational resources.

The main steps in the application of the algorithm are shared by its Cartesian and joint-space implementation both for the point-to-point and fly commands; this conceptual overlapping directly applies to the practical design of the software: a common modular library has been developed. The main steps to design the trajectory, conceptually similar to ref. [3], are given below:

- Mapping of the kinematic constraints into corresponding constraints of the curvilinear abscissa;
- Build of the path according to the specific motion (joint/Cartesian or point-to-point/fly motions);
- Design of a *tentative* trajectory in the curvilinear abscissa by resorting to the concatenated quintic polynomials;[14]
- In the Cartesian space an inverse kinematics is required to retrieve the joint-space trajectory;
- Check for the dynamic constraints;
- Eventual scaling of the curvilinear abscissa using a proper, analytically derived, scaling factor.

In the following, the joint and Cartesian cases will be discussed both for the point-to-point as well as for the fly case.

5.1. The point-to-point joint-space case

The main steps required to implement the point-to-point motion in the joint space are given above. In detail, the path is a segment in the joint space, moreover the dynamic check is performed by resorting to the concept of *significant* points introduced in ref. [3]. This is the only case that is conceptually equal to ref. [3], its sketch diagram is reported in Fig. 4. Because of the property of the curvilinear abscissa

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

described in Section 3.2, the algorithm guarantees that the jerk maximum is always 98% of its limit , the cruise velocity is always its limit and the torque or the acceleration reach their limits at least in two points.

In particular, with regards to the mapping of kinematic constraints from joint space to limits on the curvilinear abscissa, assuming a segment from $\boldsymbol{q}_0$ to $\boldsymbol{q}_f$, the following limits have been assumed:

$$\begin{cases} \dot{s}_{\max} = \min_i \left\{ \dfrac{\dot{q}_{i,\max}}{|q_{i,f} - q_{i,0}|} \right\}, \\ \ddot{s}^{\text{tent}}_{\text{acc},\max} = \ddot{s}^{\text{tent}}_{\text{dec},\max} = \min_i \left\{ \dfrac{\ddot{q}_{i,\max}}{|q_{i,f} - q_{i,0}|} \right\}, \\ \dddot{s}_{\text{acc},\max} = \dddot{s}_{\text{dec},\max} = \min_i \left\{ \dfrac{\dddot{q}_{i,\max}}{|q_{i,f} - q_{i,0}|} \right\}, \end{cases} \tag{24}$$

where in the first step, maximum values of second and third derivatives of the curvilinear abscissa during the acceleration and deceleration phases have been assumed to be equal to the maximum value allowed by the joint kinematic constraints. These kinematic limits are used to generate a *tentative* curvilinear abscissa based on ref. [14]. Nevertheless the torque constraints still need to be taken into account. With regards to the dynamic check, the idea of *significant* points has been resorted from ref. [3]; based on the parameterized model (8) and on reasonable approximations, it is possible to show that the maximum torque occurs at given points on a trapezoidal-like velocity profile. The same conclusions can be drawn about the quasi-trapezoidal-like profile described in ref. [14]; in this way two scaling coefficients c_{acc} and c_{dec} respectively for the maximum acceleration and deceleration can be calculated leading to the following new limits:

$$\begin{cases} \ddot{s}_{\text{acc},\max} = \dfrac{\ddot{s}^{\text{tent}}_{\text{acc},\max}}{\max\{1, c_{\text{acc}}\}}, \\ \ddot{s}_{\text{dec},\max} = \dfrac{\ddot{s}^{\text{tent}}_{\text{dec},\max}}{\max\{1, c_{\text{dec}}\}}, \end{cases} \tag{25}$$

where scale coefficients smaller than one have been discarded to not overshoot kinematic acceleration limits.

5.2. *The fly joint-space case*

Given an initial, a final and an intermediate configuration in the joint space, the fly command requires that the manipulator moves, as fast as possible, passing *close* to the via-point. Since the trajectory is build in the joint space, there is not meaningful metric to define how far the trajectory may pass from the via-point. In addition, the fly command should be compatible with a multifly option and should be coherent with the already build trajectory.

In ref. [3], a solution has been proposed and successfully used. In few words it is necessary to design two, independent, point-to-point motions, from the initial to the intermediate and from the intermediate to the final configuration. This can be done by implementing the algorithm described in Section 5.1. The second movement is then properly *shifted* backward trying to maximize the *overlap* between the two trajectories and thus decreasing the total traveling time based on the dynamic constraints.

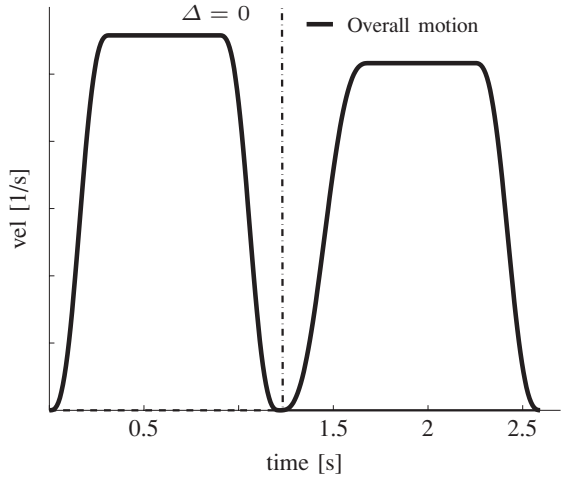


Fig. 5. Superimposition of velocity profile in joint fly motions. No superimposition is present.

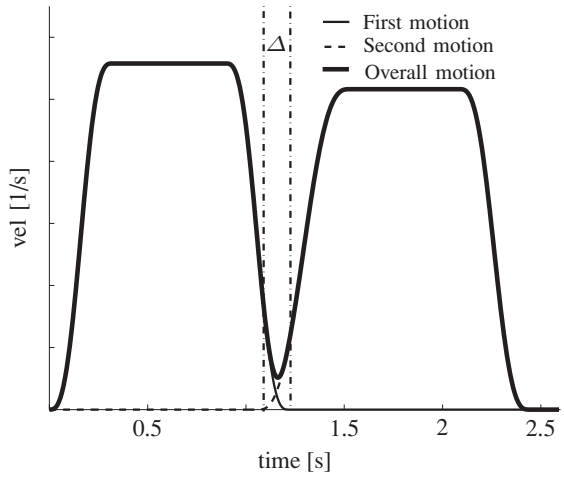


Fig. 6. Superimposition of velocity profile in joint fly motions. Partial superimposition.

As in the shifting it is not possible to take into account kinematic constraints, in this paper an alternative approach is used. Once the first point-to-point motion has been computed, the acceleration phase of the second motion is properly designed so as to satisfy the kinematic constraints. More in detail, let be $\boldsymbol{q}_0, \boldsymbol{q}_{via}, \boldsymbol{q}_f$ the initial, intermediate and final desired joint configuration, respectively. The strategy proposed here consists, first, in planning two point-to-point motions (from $\boldsymbol{q}_0$ to $\boldsymbol{q}_{via}$ and from $\boldsymbol{q}_{via}$ to $\boldsymbol{q}_f$) as in Section 5.1; then, the acceleration phase of the second movement is re-planned considering always the maximum superimposition $\Delta_{\max}$ and in such a way to saturate the kinematic or dynamic constraints on some joints.

Figures 5–7 show the three different situations that can arise with both the approaches described. In Fig. 5, a case where it is physically impossible to avoid to stop the machine in the intermediate configuration is represented, this can happen when, for example, a *limiting* joint is performing an

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

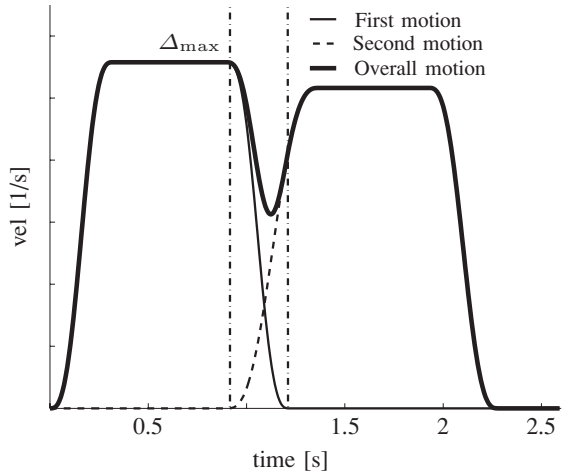


Fig. 7. Superimposition of velocity profile in joint fly motions. Maximum superimposition.

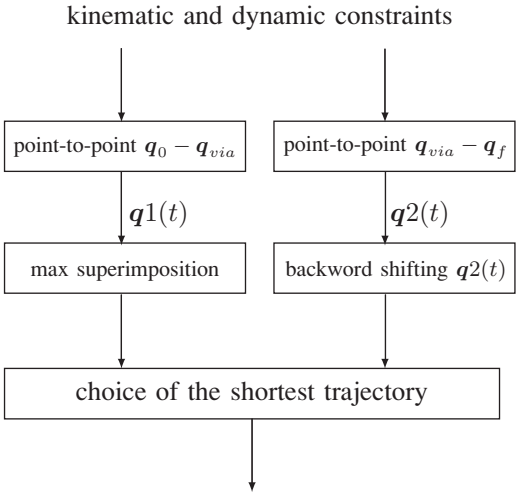


Fig. 8. Scheme of the trajectory planner for fly joint motions.

inverting motion (see ref. [3] for the definition of inverting motion and limiting joint). In Fig. 6, the situation of a partial allowed overlapping is drawn, while in Fig. 7 the case when it is possible to achieve the maximum overlapping between the two motions is shown.

It is possible to experimentally verify that the two strategies are not equivalent and that, in general, it is not possible to know in advance which is the better one in term of duration of the overall trajectory. This means that the overall motion is computed via both the two strategies and, finally, the shorter is used. This choice does not imply an high computational load. Even if the first motion is already an optimized point-to-point motion, by applying this strategy up to the 20% of the traveling time can be reduced.

Figure 8 reports a scheme summarizing the fly joint motion strategy that has been adopted.

5.3. *The point-to-point Cartesian space case*

Most of the industrial applications require a path planning in the Cartesian space. As described in Section 2, in addition to all the constraints considered in joint space, the end-effector cruise linear velocity is, in general, assigned.

It is necessary to map the kinematic constraints, naturally defined in the joint space, in the Cartesian space. The corresponding mapping function is nonlinear and configuration dependent. A possible choice consists in planning several joint movements planned as in Section 5.1 along all the manipulator workspace and, by means of Eqs. (11), calculating the corresponding trajectories in Cartesian space. Then, starting from these trajectories, approximative average limits in Cartesian space $\dot{\boldsymbol{x}}_{\max}$, $\ddot{\boldsymbol{x}}_{\max}$, $\dddot{\boldsymbol{x}}_{\max}$, can be calculated and mapped on the tentative curvilinear abscissa in the Cartesian space by following the same reasoning as in Eq. (24) to produce a *tentative* trajectory. For example, by taking into account the expression of a segment in the Cartesian space from $\boldsymbol{x}_0$ to $\boldsymbol{x}_f$:

$$\begin{cases} \dot{s}^{\text{tent}}_{\max} = \min_i \left\{ \dfrac{\dot{x}_{i,\max}}{|x_{i,f} - x_{i,0}|} \right\}, \\ \ddot{s}^{\text{tent}}_{\text{acc,max}} = \ddot{s}^{\text{tent}}_{\text{dec.max}} = \min_i \left\{ \dfrac{\ddot{x}_{i,\max}}{|x_{i,f} - x_{i,0}|} \right\}, \\ \dddot{s}^{\text{tent}}_{\text{acc,max}} = \dddot{s}^{\text{tent}}_{\text{dec,max}} = \min_i \left\{ \dfrac{\dddot{x}_{i,\max}}{|x_{i,f} - x_{i,0}|} \right\}, \end{cases} \tag{26}$$

where $\boldsymbol{x}_0 = [\boldsymbol{p}_0^{\mathrm{T}} \quad \boldsymbol{\phi}_0^{\mathrm{T}}]^{\mathrm{T}}$ and $\boldsymbol{x}_f = [\boldsymbol{p}_f^{\mathrm{T}} \quad \boldsymbol{\phi}_f^{\mathrm{T}}]^{\mathrm{T}}$ are the initial and final end-effector configurations. If the linear cruise velocity v_c has been assigned, the cruise value of the curvilinear abscissa can be calculated as

$$\dot{s}_{\max} = \frac{v_c}{\|\boldsymbol{p}_f - \boldsymbol{p}_0\|}, \tag{27}$$

which together with the equation of a segment in the Cartesian space[22] allows to obtain the desired linear velocity. Finally, the *tentative* curvilinear abscissa is s^{tent} : $\{\dot{s}^{\text{tent}}_{\max}, \ddot{s}^{\text{tent}}_{\text{acc,max}}, \dddot{s}^{\text{tent}}_{\text{acc,max}}, \ddot{s}^{\text{tent}}_{\text{dec,max}}, \dddot{s}^{\text{tent}}_{\text{dec,max}}\}$.

Given a tentative trajectory in the Cartesian space, the corresponding trajectory in the joint space is obtained by resorting to an inverse kinematics algorithm. The latter is then sampled to check the kinematic and dynamic limits' violation and eventually scaled. To this aim, considering Eq. (11), the following scaling law has been considered:

$$\begin{cases} \bar{t} &= c \cdot t, \\ \boldsymbol{x}(t) &= \boldsymbol{x}(\bar{t}) \quad \Leftrightarrow \boldsymbol{q}(t) = \boldsymbol{q}(\bar{t}), \\ \dot{\boldsymbol{x}}(t) &= c\dot{\boldsymbol{x}}(\bar{t}) \quad \Leftrightarrow \dot{\boldsymbol{q}}(t) = c\dot{\boldsymbol{q}}(\bar{t}), \\ \ddot{\boldsymbol{x}}(t) &= c^2\ddot{\boldsymbol{x}}(\bar{t}) \quad \Leftrightarrow \ddot{\boldsymbol{q}}(t) = c^2\ddot{\boldsymbol{q}}(\bar{t}), \\ \dddot{\boldsymbol{x}}(t) &= c^3\dddot{\boldsymbol{x}}(\bar{t}) \Leftrightarrow \dddot{\boldsymbol{q}}(t) = c^3\dddot{\boldsymbol{q}}(\bar{t}), \\ \boldsymbol{\tau}(t) &= c^2\boldsymbol{\tau}(\bar{t}), \end{cases} \tag{28}$$

where c is a positive scaling factor close to 1. The last equality in Eq. (28) comes considering the Eqs. (2), as described in ref. [22]. From Eqs. (28), it can be easily shown that a scaled trajectory in Cartesian space with a certain c, leads to a scaled trajectory in joint space with the same coefficient.

Given a tentative Cartesian curvilinear abscissa and the corresponding joint trajectory $\boldsymbol{q}^{\text{tent}}$, the Eqs. (28) imply that,

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

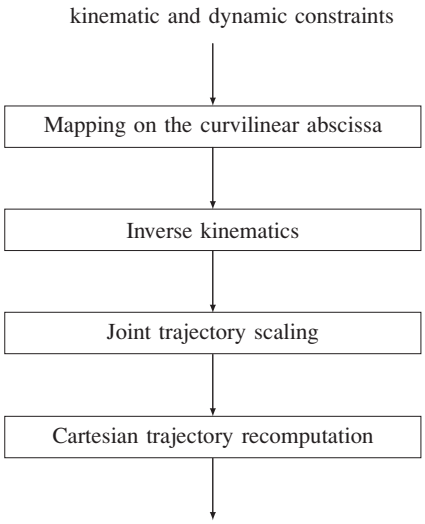


Fig. 9. Scheme of the trajectory planner for point-to-point Cartesian motions.

differently from the joint case, we cannot scale separately the Cartesian velocity, acceleration, and jerk; but we must consider only one scaling coefficient c calculated as

$$\begin{cases} c_1 = \max_i \max_t \dfrac{\left|\dot{q}_i^{\text{tent}}(t)\right|}{\dot{q}_{i,\max}}, \\ c_2 = \sqrt{\max_i \max_t \dfrac{\left|\ddot{q}_i^{\text{tent}}(t)\right|}{\ddot{q}_{i,\max}}}, \\ c_3 = \sqrt[3]{\max_i \max_t \dfrac{\left|\dddot{q}_i^{\text{tent}}(t)\right|}{\dddot{q}_{i,\max}}}, \\ c_4 = \sqrt{\max_i \max_t \dfrac{|\tau_i^{\text{tent}}(t)|}{\tau_{i,\max}}}, \\ c = \max\{c_1, \quad c_2, \quad c_3 \quad c_4\}. \end{cases} \tag{29}$$

Notice that, by exploiting the properties of the polynomial functions and the quintics, it is convenient to scale separately the acceleration, cruise, and deceleration phases, but the cruise phase should not be scaled if the cruise velocity corresponds to a technological requirement in case of, e.g., gluing. In such a case an error message is usually given do the operator and the maximum allowed cruise velocity for the planned motion is provided.

Finally, in Fig. 9 the graphical representation of the steps above is shown.

5.4. The fly Cartesian space case

In many industrial applications such as, e.g., gluing or welding, the user gives an initial, a final and an intermediate configurations together with a maximum distance $d_{\max} \in \mathbb{R}$ from the via points $\boldsymbol{p}_{via} \in \mathbb{R}^{3\times 1}$ (see Fig. 10). In addition, a constant, *technological* cruise velocity needs to be respected also during the *bends* and in case of multi-flies, i.e., if additional via-points are added when the movement has already started.

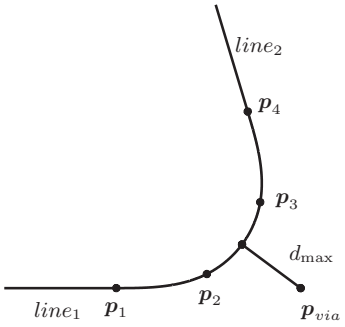


Fig. 10. Example of clotoid bend obtained by a segment, a clotoid, a circle's arc, a clotoid and a final segment.

In practice, the overall velocity profile is trapezoidal-like with an initial acceleration, a final deceleration and a long cruise at the given velocity; during the cruise the manipulator passes close to the via-point depending on the selected distance. From Fig. 10, it can be recognized that the maximum distance constraints, even if given as an upper bound, can be assumed at an equality constraint to avoid useless dynamic stress to the manipulator. This can be understood with a simple physical example: at a given velocity the centripetal forces are larger for smaller bends curvature. This is exactly what it is necessary to avoid and thus the manipulator end-effector passes exactly at $d_{\max}$ from $\boldsymbol{p}_{via}$.

The solution implemented for the fly in the joint space cannot be used, since it does not build a constant velocity profile in the Cartesian space. The solution proposed consists in defining a proper path that links the two segments connecting the via-point with the initial and final points; moreover, the path strictly satisfies the maximum distance from the via-point. In order to guarantee that the trajectory is of class $\mathcal{C}^2$, in addition, it is necessary that the path satisfies specific conditions on its curvature. A similar problem arises in the street and railways construction where the discontinuity in the centripetal acceleration needs to be avoided for several reasons; based on those results, the proposed path is then build by mean of a function called *clotoid*[1] defined by the equation $r \cdot s = A^2$, where s is the curvilinear abscissa, r is the curvature, and A is a parameter to be designed (see the Section A.1 in the Appendix for further details). The characteristic of this curve is that its curvature is continuous and, if the time law of the parameter s is also $\mathcal{C}^2$, then the designed trajectory is $\mathcal{C}^2$ and the smoothness constraint is satisfied. The details on the design customization of the clotoid blend between two segments given a maximum distance from the via point are given in Section A.2 in the Appendix.

It is worth noticing that alternative approaches might have been based on spline-based curves[19] that, however, cannot be considered due to the difficulty to exactly design the maximum distance from the via-point.

Once the clotoid-based path has been designed, a tentative curvilinear abscissa on the overall path needs to be designed ensuring the selected cruise velocity. To this aim, it is useful to notice that maximum acceleration and jerk $\ddot{s}_{\text{acc,max}}, \dddot{s}_{\text{acc,max}}$ of the acceleration phase and the maximum deceleration

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

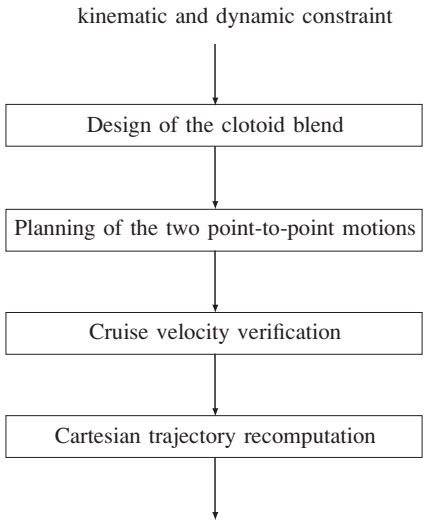


Fig. 11. Scheme of the trajectory planner for fly Cartesian motions.

and jerk of the deceleration phase $\ddot{s}_{\text{dec,max}}$, $\dddot{s}_{\text{dec,max}}$ can be inherited planning the point-to-point motions $\boldsymbol{x}_0$-$\boldsymbol{x}_{via}$ and $\boldsymbol{x}_{via}$-$\boldsymbol{x}_f$ respectively, as described in Section 5.3. Given a cruise velocity v_c, the maximum curvilinear abscissa velocity is $s_{\text{cruise,max}} = v_c/L$, where L is the length of the overall path. From the point-to-point motions in Section 5.3, information about the feasibility of the cruise velocity along the two segments can be retrieved.

Assuming that the cruise velocity v_c is feasible along the two segments, the constraint limits need to be verified along the clotoid blend. By means of Eqs. (29), a scaling coefficient c for the cruise velocity can be calculated and taken into account in the case of constraints' violation.

Once, a scaling coefficient has been calculated the overall curvilinear abscissa can be easily computed. The overall strategy is summarized in Fig. 11.

6. Validation

The validation of the algorithm has been extremely severe and concerned both numerical simulations and experiments. The numerical simulations involved two different kind of industrial manipulators (Fig. 12) characterized by a different size and kinematics, the Comau SIX and the Comau NJ 130. Both are 6-DOFs manipulators but Smart SIX handle up to 6 Kg with an open-chain structure while the Smart NJ 130 has a payload capacity of 130 kg, a parallelogram structure and the wrist kinematically and dynamically coupled.

Table I shows the Denavit–Hartenberg's parameters for both the manipulators.

For both the manipulators the algorithm was run with or without the presence of a known load at the end-effector. In addition, the algorithm is exactly the same, the only difference is in the upload of the proper dynamic model and set of constraints.

Dozen of movements were run taken from a set of industrial demanding trajectories owning to real applications both in the Cartesian and in the joint space. Overall, several hundreds of trajectories were successfully simulated. By properly interfacing the simulation environment with the control hardware of the robots, the same trajectories have been commanded experimentally, this, in turn, prove the validity of the mathematical model[2] and the effectiveness of the developed motion planning algorithm.

In the following, sample case studies, together with aggregate information, are reported and commented.

Table I. Denavit–Hartenberg's parameters. Smart SIX on the top, and Smart NJ 130 on the bottom.

Smart SIX

link #	α [rad]	d [m]	a [m]	θ [rad]
1	−1.5708	0.4500	0.1500	θ_1
2	0	0	0.5900	θ_2
3	−1.5708	0	0.1300	θ_3
4	1.5708	0.6471	0	θ_4
5	−1.5708	0	0	θ_5
6	0	0.0950	0	θ_6

Smart NJ30

link #	α [rad]	d [m]	a [m]	θ [rad]
1	1.5708	0.8300	0.3500	θ_1
2	0	0	1.1600	θ_2
3	0	0	0.3200	θ_3
4	1.5708	1.4238	0	θ_4
5	−1.5708	0	0	θ_5
6	0	0.2100	0	θ_6
7	0	0	1.1600	θ_7
8	1.5708	0	0.2779	θ_8



Fig. 12. Comau Smart SIX on the left and Comau Smart NJ 130 on the right (images in scale).

6.1. Case study 1: Point-to-point joint motion

As a first example, the results for the point-to-point joint motion are reported. The starting joint configuration is $\boldsymbol{q}_0 = [0, 1.5708, -2.3562, 0, 1.5708, 0]$ rad and the final one $\boldsymbol{q}_f = [0.6109, -0.7854, -0.7854, 0, 1.5708, 0]$ rad. In Fig. 13, the time history of the six joint velocities, accelerations, jerks, and torques for the Smart SIX are shown. The data are normalized with respect to their maximum limit so that the unitary value is the limit for each variable and each joint. For this specific movement, the velocity of joint 1

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

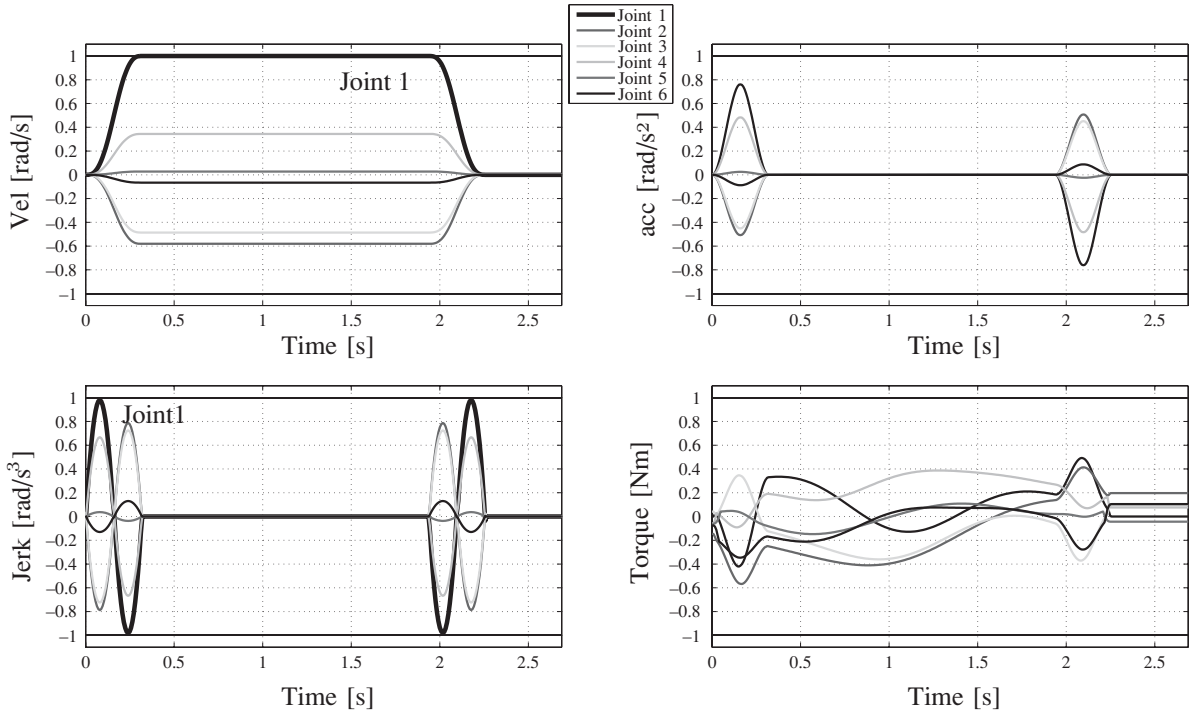


Fig. 13. Comau Smart SIX. Joint profiles for a point-to-point joint motion. The variables for all the joints are normalized to the unitary value, the limiting joint is drawn with a larger line.

Table II. Percentage of Constraints' fulfilment for a subset of significant and demanding point-to-point joint movements. Smart SIX on the top, and Smart NJ 130 on the bottom.

Smart SIX

Trajectory	Velocity	Acceleration	Jerk	Torque
1	100% Joint 1	76% Joint 1	98% Joint 1	57% Joint 2
2	100% Joint 2	74% Joint 2	98% Joint 2	55% Joint 2
3	100% Joint 2	74% Joint 2	98% Joint 2	44% Joint 3
4	100% Joint 1	76% Joint 1	98% Joint 1	73% Joint 1
5	48% Joint 2	51% Joint 2	98% Joint 2	52% Joint 2
6	79% Joint 3	67% Joint 3	98% Joint 3	47% Joint 3
7	100% Joint 4	100% Joint 4	98% Joint 4	87% Joint 5

Smart NJ30

Trajectory	Velocity	Acceleration	Jerk	Torque
1	100% Joint 1	82% Joint 1	98% Joint 1	99% Joint 1
2	100% Joint 2	94% Joint 2	98% Joint 2	101% Joint 2
3	100% Joint 2	94% Joint 2	98% Joint 2	102% Joint 2
4	100% Joint 1	78% Joint 1	98% Joint 1	100% Joint 1
5	57% Joint 2	58% Joint 2	98% Joint 2	99% Joint 2
6	100% Joint 3	78% Joint 3	98% Joint 3	101% Joint 3
7	100% Joint 4	100% Joint 4	98% Joint 4	49% Joint 4

is saturated most of the time together with the jerks in the acceleration and deceleration phases.

Table II (upper part) reports the velocity, acceleration, jerk and torque percentage constraints fulfilment together with the joint achieving the maximum percentage value for a subset of significant and demanding movements. These movements bring the manipulator from one side of the workspace to the other and are particularly challenging from the kinematic and/or dynamical point of view. It can be observed that, in this specific case study, the torques are not saturated and the jerk is the most limiting constraint.

In order to show the algorithm's result in a different configuration, the Smart NJ 130 with the payload of 130 kg is considered. The results of a subset of significant and demanding trajectories, reported in Table II (lower part), show that, as expected, because of the size of the manipulator and the presence of the heavy payload the torque is now the more limiting constraint.

Notice that, in both cases, the jerk always saturates. In addition, the approximations made, as the position of the significant points in the scaling procedure, cause the torques to violate its limit by 1%–2%; this drawback can be taken into account, for example, by setting via software the torque limit at the, e.g., 95% of its true value.

6.2. *Case study 2: Joint fly motion*

Figure 14 reports the normalized joint variables for a Comau Smart SIX manipulator holding a payload of 6 kg in the case of a specific fly movement. The starting point is $\boldsymbol{q}_0 = [2.9671, -0.7854, -1.7453, 0, 0, 0]$ rad, the via-point $\boldsymbol{q}_{via} = [0.7854, -0.5236, -1.1345, -1.5708, 0, 1.5708]$ rad and the final point $\boldsymbol{q}_f = [-1.5708, -0.9599, -1.2217, 3.1416, 3.1416, 3.1416]$ rad. The superimposition (*fly*) phase is highlighted with two vertical lines. In this case, the superimposition starts at about 1.3 s and ends at about 1.7 s, and it is limited by the jerk constraint on joint 3; the gain, in term of traveling time on the sum of traveling time of two optimized trajectories, is about 11%.

More extensively, Table III reports the traveling time percentage improvement for both the manipulators Smart SIX and NJ 130. A set of 23 fly movements has been

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

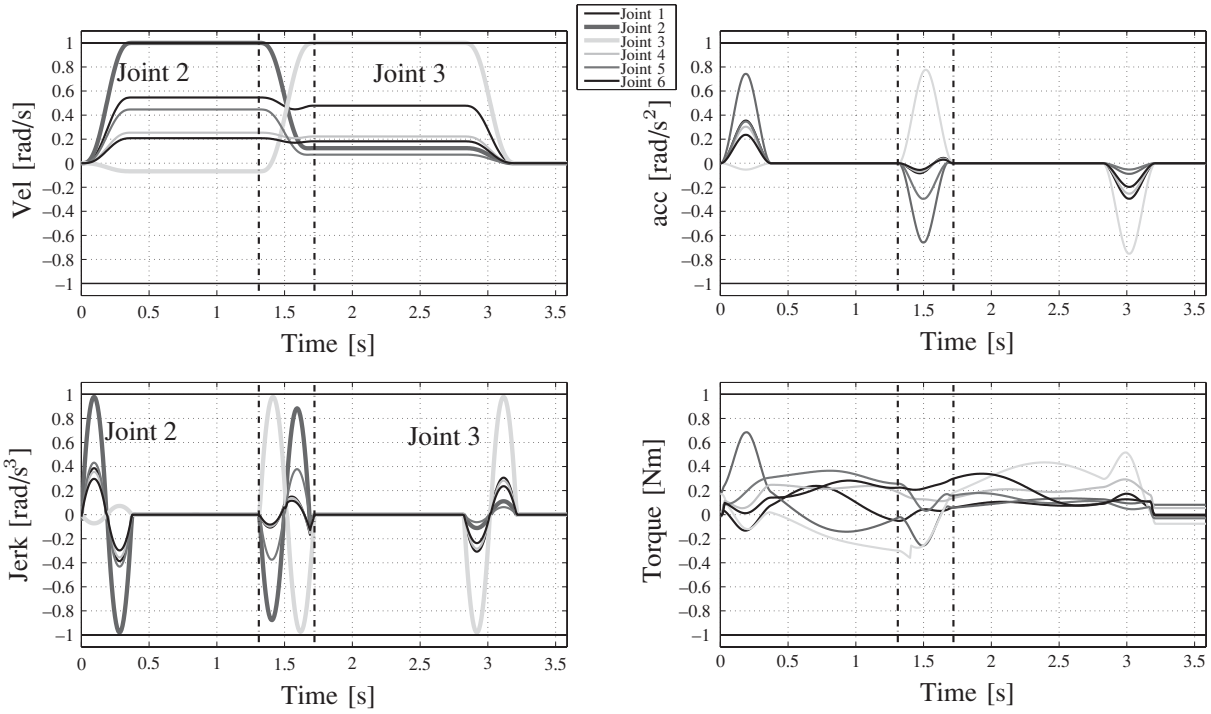


Fig. 14. Comau Smart SIX. Joint profiles for a fly joint movement. The variables for all the joints are normalized to the unitary value, the limiting joints are drawn with larger lines.

Table III. Percentage gain for fly joint movements and for both Comau Smart SIX and Smart NJ 130 manipulator, with or without payload..

	Manipulator			
Traj.	Six with load	Six no load	NJ 130 with load	NJ 130 no load
1	21%	21%	19%	19%
2	12%	12%	8%	12%
3	10%	10%	10%	10%
4	12%	13%	13%	13%
5	20%	20%	20%	20%
6	21%	22%	22%	24%
7	22%	22%	27%	27%
8	9%	9%	8%	10%
9	12%	13%	10%	13%
10	22%	22%	22%	23%
11	10%	10%	4%	7%
12	23%	23%	24%	24%
13	10%	10%	10%	10%
14	13%	13%	13%	13%
15	20%	20%	20%	20%
16	22%	22%	22%	24%
17	22%	22%	25%	27%
18	9%	9%	9%	10%
19	13%	13%	10%	13%
20	22%	22%	22%	23%
21	10%	10%	4%	7%
22	7%	7%	21%	8%
23	21%	21%	19%	19%

considered with and without maximum payload and the percentage gain is calculated, as in the previous case study, comparing the overall traveling time of the optimized fly movements obtained and the sum of the traveling times of the single optimized trajectories. The percentage improvements in the cases of presence or absence of payload for the Smart SIX manipulator are the same for the considered trajecotries; this is due to the fact that, for this type of manipulator, kinematic limits are more constraining than dynamical ones.

6.3. Case study: Point-to-point Cartesian motion

One Cartesian point-to-point motion for the Smart SIX is reported as significant case study. The starting end-effector configuration is $\boldsymbol{x}_0 = [-0.8\text{ m}, -0.9\text{ m}, 0.98\text{ m}, 0.6\text{ rad}, 1\text{ rad}, -0.1\text{ rad}]$ and the final one $\boldsymbol{x}_f = [-0.55\text{ m}, 0.45\text{ m}, -0.49\text{ m}, -2.2\text{ rad}, 0.9\text{ rad}, 0.8\text{ rad}]$, the orientation is expressed in Euler's angles and the path is a segment. The maximum velocity is considered, as automatically calculated by the algorithm. Figure 15 reports the corresponding Cartesian profiles. Figure 16 reports the normalized joint variables as obtained after the kinematic inversion of the planned trajectory. It can be observed that the joint velocity and jerk of joint 3 reach the maximum values respectively in the acceleration and deceleration phases. As stated in Section 5.3, differently from the joint case, because of the nonlinear mapping between joint and Cartesian space, in general it is not possible to guarantee that more than one joint limit (among velocity, acceleration, jerk, and torque constraints) is reached in each of the acceleration, cruise, and deceleration phases of the Cartesian motion.

6.4. Case study: Cartesian fly motion

As representative case study, one single fly command in the Cartesian space is reported. The starting end-effector pose is $\boldsymbol{x}_i = [0.5793\text{ m}, \ -0.2000\text{ m}, 0.8225\text{ m}, 0\text{ rad}, 2.0944\text{ rad},$

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

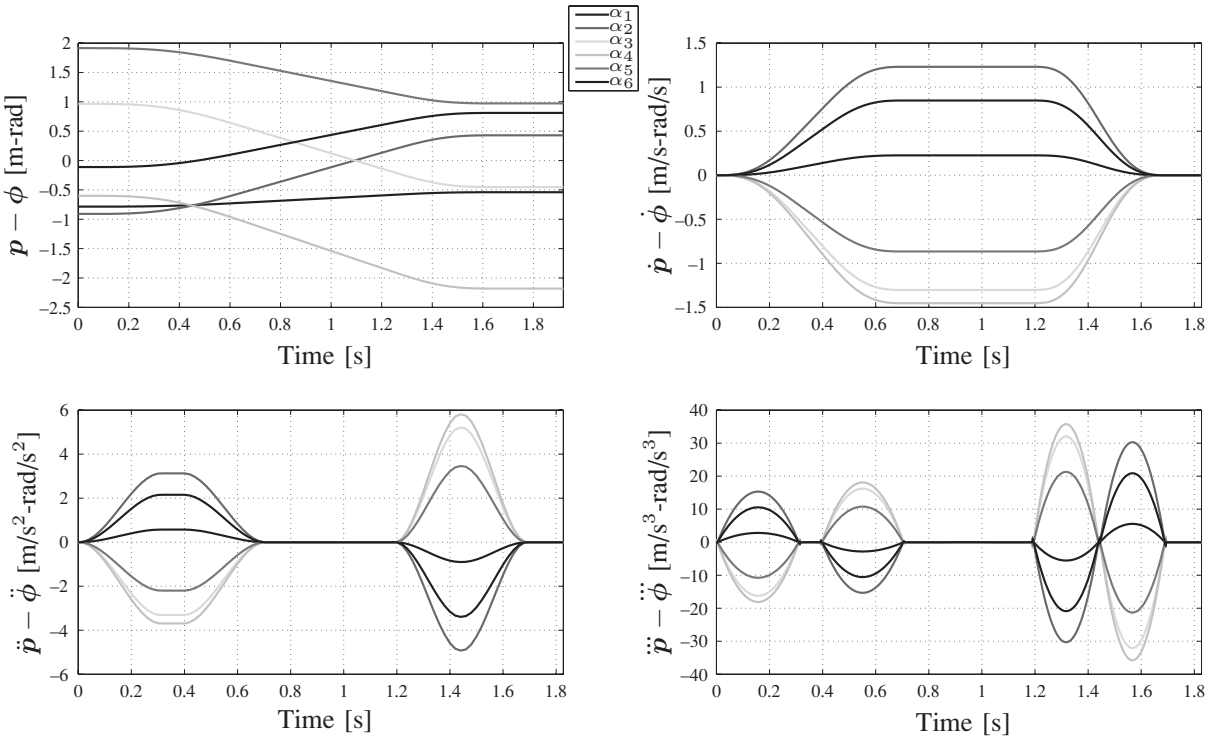


Fig. 15. Comau Smart SIX. Time history of end-effector configuration for a segment in Cartesian space. The variable α_i denotes the terms related to the ith component of vector $\boldsymbol{x}$ (i.e. x_i, $\dot{x}_i$, $\ddot{x}_i$, $\dddot{x}_i$).

0 rad], the via-point $\boldsymbol{x}_{via} = [0.5793\,\text{m}, 0\,\text{m}, 0.8225\,\text{m}, 0\,\text{rad}, 2.0944\,\text{rad}, 0\,\text{rad}]$ and the final pose $\boldsymbol{x}_f = [0.7174\,\text{m}, -0.1381\,\text{m}, 0.7793\,\text{m}, -0.1396\,\text{rad}, 2.0186\,\text{rad}, -0.0349\,\text{rad}]$. Figure 17 shows the designed path of a fly movement with a $d_{\max}$ of 0.02 m; in particular it can be seen as two segments from $\boldsymbol{p}_i$ to $\boldsymbol{p}_{via}$ and from $\boldsymbol{p}_{via}$ to $\boldsymbol{p}_f$ are joined with a clotoid bend. In Fig. 18, the time history of end-effector configuration has been reported, pointing out the time interval ($[0.8 \div 1.3]$ s) in which the clotoid bend occurs. In Fig. 19, the end-effector's linear velocity along the

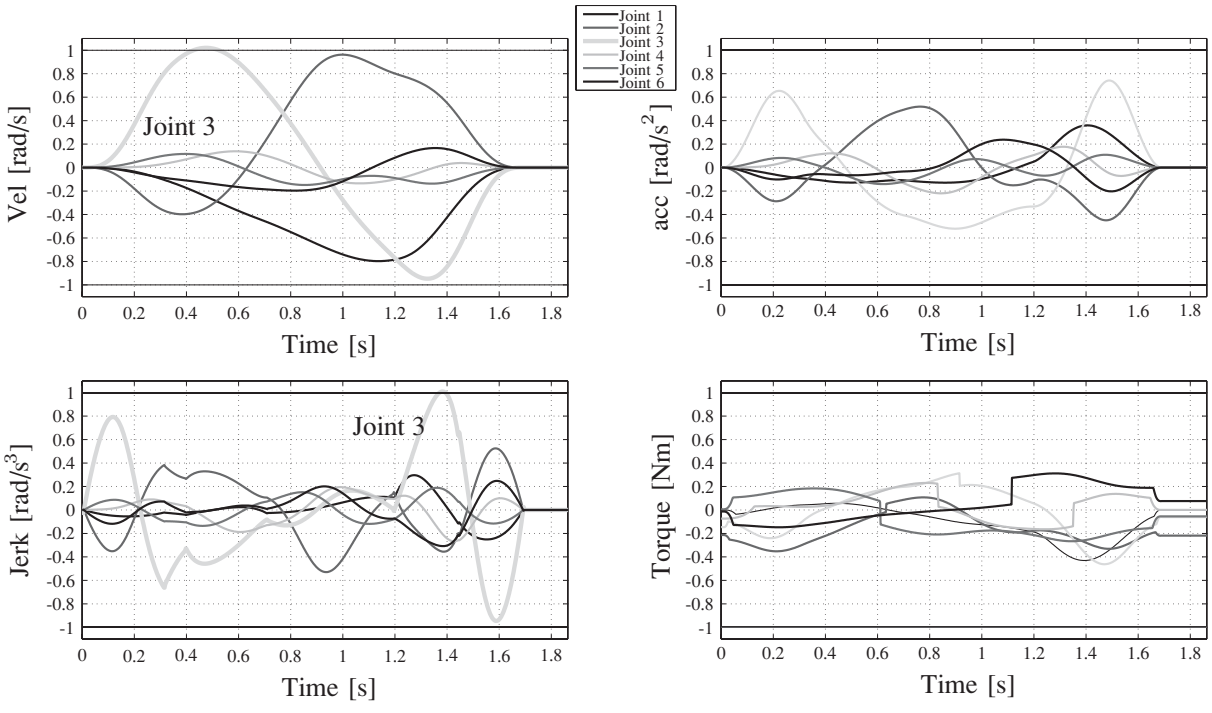


Fig. 16. Comau Smart SIX. Joint profiles after kinematic inversion of a segment in Cartesian space. The variables for all the joints are normalized to the unitary value, the limiting joint is drawn with a larger line.

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

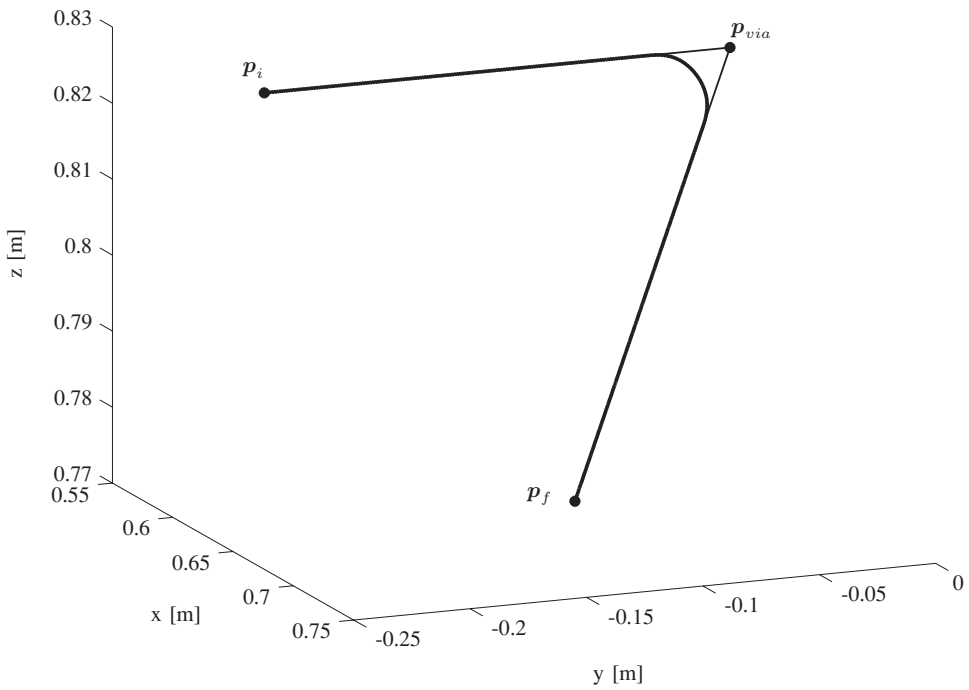


Fig. 17. Comau Smart SIX. Path in Cartesian space of a fly movement between two segment joined with a clotoid bend. $\boldsymbol{p}_i$, $\boldsymbol{p}_{via}$, $\boldsymbol{p}_f$ are initial, via and final end-effector position respectively.

path shows a *trapezoidal-like* profile, and that, as stated in Section 5.4, the overall orientation profile is obtained anticipating the execution of the second planned orientation trajectory. Finally, as regards to the manipulator physical constraints, Fig. 20 shows the time history of the joint profiles after the kinematic inversion of the planned Cartesian movement. In particular, the jerk on joint 1 reaches the maximum value in particular during the clotoid bend, meaning that the maximum allowable cruise velocity has been reached.

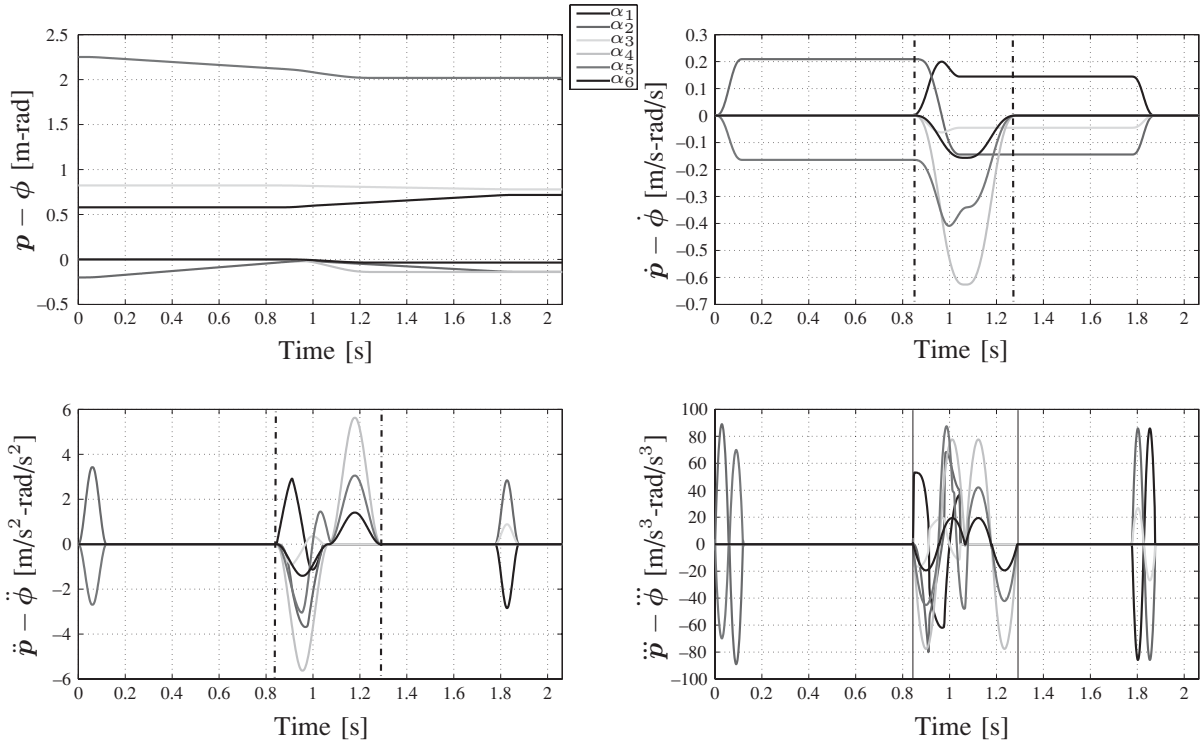


Fig. 18. Time history of end-effector configuration of a Cartesian fly movement between two segments joined with a clotoid bend. The variable α_i denotes the terms related to the ith component of vector $\boldsymbol{x}$ (i.e. x_i, $\dot{x}_i$, $\dot{x}_i$, $\ddot{x}_i$, $\dddot{x}_i$).

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

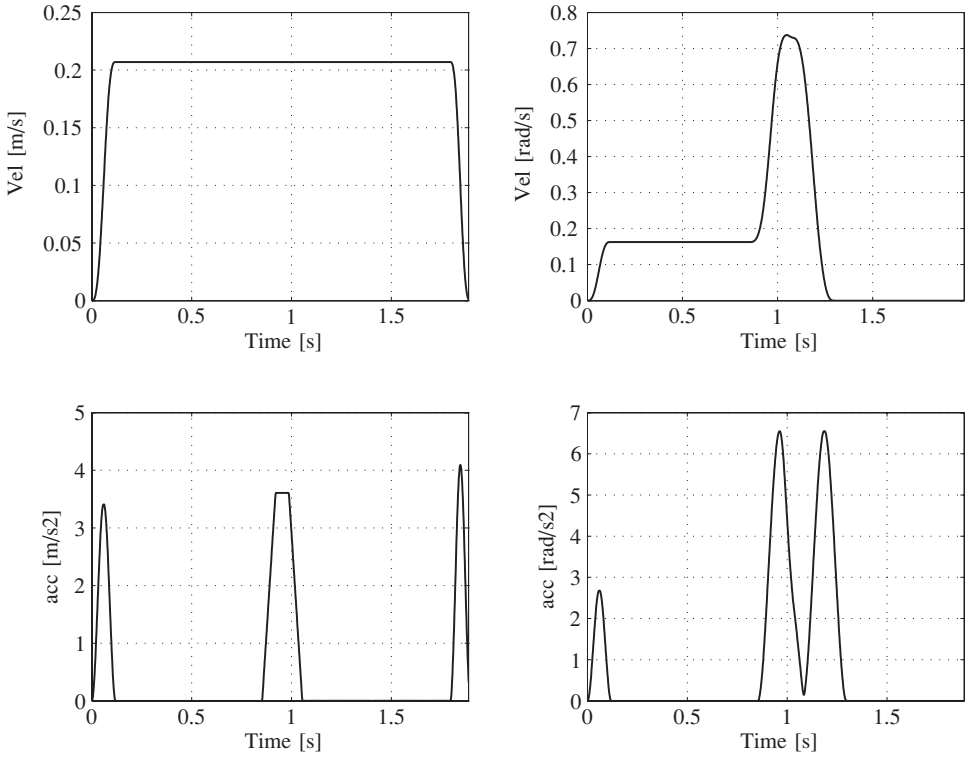


Fig. 19. Comau Smart SIX. Velocity and Acceleration module of a Cartesian fly movement between two segment joined with clotoid bend.

6.5. Case study: Experiments of multifly in joint space

As an example of the feasibility of the proposed approach, a multifly movement composed of 24 joint points has been executed on a Comau Smart SIX manipulator (see Section 6). In several machines, the trajectory may cause the rupture of joints' reducers. This is mainly due to the presence of several inverting joint motions at high speed and involving all joints together. Figure 21 shows time history of joints' profiles as obtained executing this movement; it is straightforward to see as all constraints have been *practically* satisfied. It can

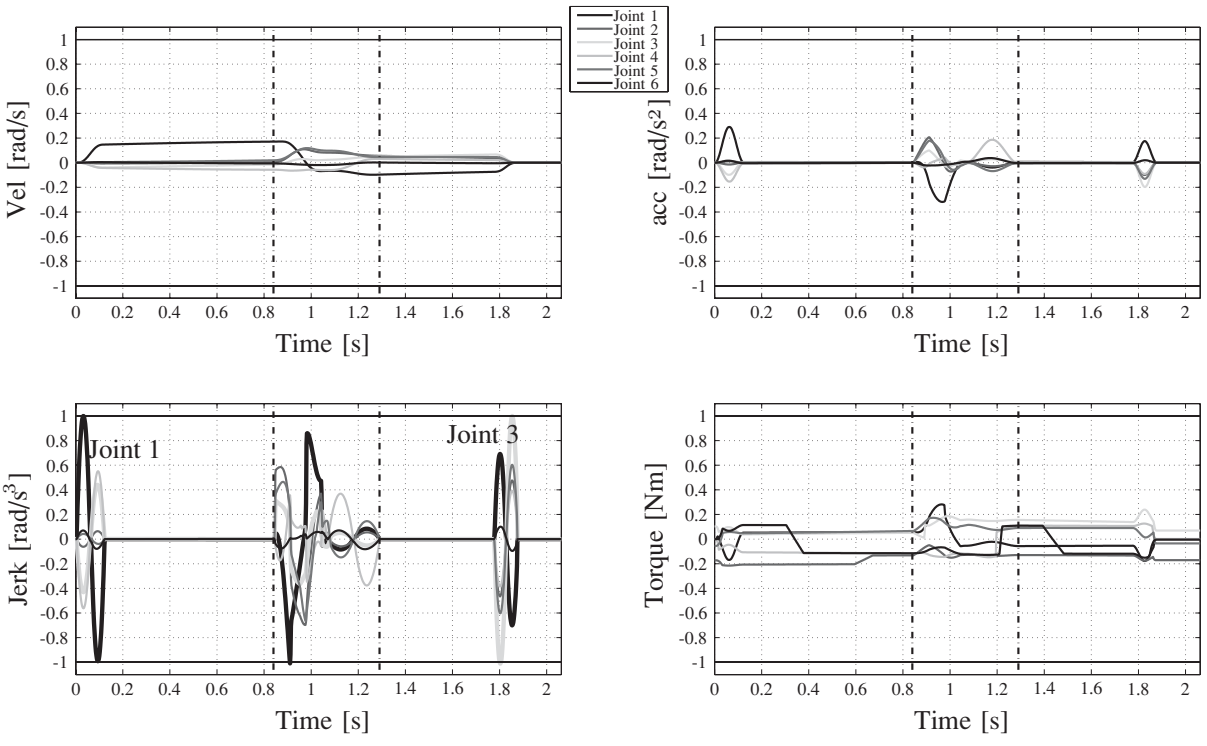


Fig. 20. Comau Smart SIX. Joint profiles after kinematic inversion of a Cartesian fly movement between two segments joined with a clotoid bend. The variables for all the joints are normalized to the unitary value, the limiting joint is drawn with a larger line.

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

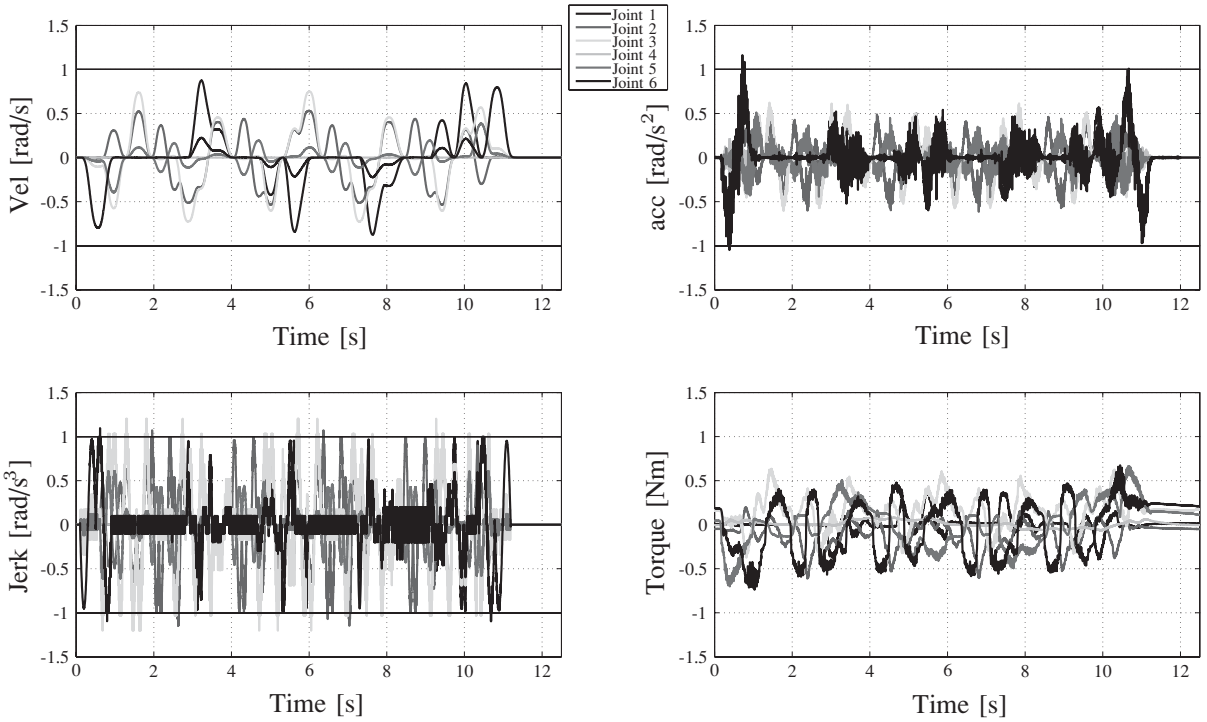


Fig. 21. Experimental validation on a Comau Smart SIX. Joint profiles of a multifly movement in the joint space.

be noticed, in fact, that several approximations arise such as the uncertainties in the dynamical parameter estimation, the unmodeled or partially modeled effects as static and dynamic friction, the joint/arm elasticity and numerical derivatives that slightly deteriorate the simulative results acceptable violation of the given constraints.

7. Conclusions

A constrained motion planning algorithm for robot manipulators has been presented in this paper. Under the same framework, Cartesian and joint-space motions, both point-to-point and fly can be handled while satisfying all the strict constraints that arise in an industrial environment. The proposed algorithm properly integrates the existing literature with original ideas in the design of the bend in the Cartesian space. The validation has been severe and intensive, hundreds of numerical simulations in widely different conditions for two manipulators as well as experimental tests confirm the effectiveness of the developed algorithm. The current implementation of the algorithm is off-line, i.e., the trajectory is computed before the movement starts, current work is devoted at adapting the algorithm to an on-line version to take into account real-time interaction with the operator.

Moreover, the applicability and the extension of the current approach to parallel manipulators will be investigated.

A. Appendix

A.1. The clotoid curve

The generic representation of a *multiparameter* clotoid curve is

$$r \cdot s^n = A^{(n+1)}, \tag{30}$$

where the scalar r is the curvature, the scalar s is the curvilinear abscissa, the constant A is a parameter, and n is the order of the clotoid function. In case of $n = 1$ the *standard* clotoid, used in this paper, is obtained. Notice that the curvature is null in the origin and tends continuously to ∞ as s tends to ∞.

In order to obtain the Cartesian coordinates of a clotoid curve, it is useful to parameterize the curve with respect to parameter τ, defined as the angle between the x-axis and the tangent to the curve.

It is straightforward to prove that the following relations are true:

$$\begin{cases} \displaystyle\int \mathrm{d}\tau = \frac{1}{A^2}\int s\,\mathrm{d}s, \\ \displaystyle\tau = \frac{s^2}{2A^2}. \end{cases} \tag{31}$$

Then, by some straightforward calculations, it is possible to demonstrate that the Cartesian equation of the clotoid blend is

$$\begin{cases} \displaystyle x = A\sqrt{2\tau}\sum_{i=1}^{\infty}\left[(-1)^{i+1}\frac{\tau^{2i-2}}{(4i-3)(2i-2)!}\right], \\ \displaystyle y = A\sqrt{2\tau}\sum_{i=1}^{\infty}\left[(-1)^{i+1}\frac{\tau^{2i-1}}{(4i-1)(2i-1)!}\right]. \end{cases} \tag{32}$$

Practically, given the desired τ, it is possible to calculate the corresponding values x and y by calculating the sums in Eq. (32) for a finite value of i until the contribution of term $(i+1)$th is less than a given threshold. It can be shown that if this threshold is equal to 10^{-8}, less than 10 terms of the

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

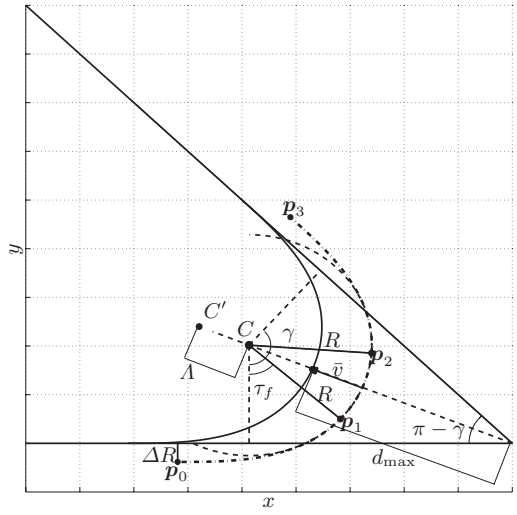


Fig. 22. Sketch of the customized construction of the clotoid-based blend.

sums in Eq. (32) are enough. Notice that 10^{-8} m is smaller than the accuracy of current industrial manipulators.

Moreover, given a point $\boldsymbol{p}$ belonging to the clotoid curve, the center of curvature $\boldsymbol{c}$ in this point is

$$\begin{cases} c_x = \dfrac{A\sqrt{2\tau}}{2} \displaystyle\sum_{i=1}^{\infty} \left[(-1)^{i+1} \frac{\tau^{2i-2}}{(4i-3)[(2i-1)!} \right] \\ \quad = A \cdot f_{c_x}(\tau), \\ c_y = \dfrac{A\sqrt{2\tau}}{2} \left\{ \dfrac{1}{\tau} + \displaystyle\sum_{i=1}^{\infty} \left[(-1)^{i+1} \frac{\tau^{2i-1}}{(4i-1)(2i)!} \right] \right\} \\ \quad = A \cdot f_{c_y}(\tau), \end{cases} \tag{33}$$

while, considering Eqs. (30) and (31), the curvature radius is

$$R = \frac{A}{\sqrt{2\tau}} = A \cdot f_R(\tau). \tag{34}$$

A.2. The clotoid blend

Given the definition of the clotoid function, it is necessary to design a proper blend between the segments $\boldsymbol{p}_0 - \boldsymbol{p}_{via}$ and $\boldsymbol{p}_{via} - \boldsymbol{p}_f$ defined in a three-dimensional space. It is not reductive to assume that the segments lie on a plane and they are not parallel. Considering a proper reference frame $O'(x, y, z)$ and a roto-translation matrix, it is possible to approach the problem in the two-dimensional while imposing the point $\boldsymbol{p}_0$ as the origin of the frame O'; in addition, the end of the first segment lies on the positive x-axis and the ending point of the second segment has a positive ordinate in the frame O'. As the curvature r of a clotoid curve goes from 0 to ∞, it is not possible to blend two segments with one single clotoid curve; then, as depicted in Fig. 22, the solution is to build a sequence of proper functions, in detail a clotoid curve (from $\boldsymbol{p}_0$ to $\boldsymbol{p}_1$) with a curvature from 0 to a finite value $\bar{r}$ followed by a circle's arc (from $\boldsymbol{p}_1$ to $\boldsymbol{p}_2$) with curvature $\bar{r}$ and another clotoid curve (from $\boldsymbol{p}_2$ to $\boldsymbol{p}_3$) to approach the second segment with a curvature from $\bar{r}$ to 0. Given the maximum distance $d_{\max}$ of the blend from the via-point, there are two ways to design the sequence clotoid–circle–clotoid.The first one consists in:

- assigning the center $\boldsymbol{c}'$ and the radius of the circle in such a way the circle is at a distance $d_{\max}$ from the via-point;
- imposing that the center of curvature of the first clotoid in Eq. (33) is equal to the center C' of the circle with unknown parameters A and τ;
- calculating the second clotoid blend by considering that the two clotoid blends are symmetric with respect to the bisector of the angle between the two segments.

The analytical solution of the Eq. (33) does not exist. For this reason, another way is possible and is based on the following steps:

- imposing τ in Eqs. (32)–(33) equal to $\tau_f = \gamma/4$ to calculate the first clotoid blend from $\boldsymbol{p}_0$ to $\boldsymbol{p}_1$;
- calculating $\boldsymbol{c}$ from Eq. (33). This will be the center of arc of circle from $\boldsymbol{p}_1$ to $\boldsymbol{p}_2$ and with radius R given by Eq. (34) evaluated in τ_f;
- calculating the second clotoid blend from $\boldsymbol{p}_2$ to $\boldsymbol{p}_3$ by considering that the two clotoid blends are symmetric with respect to the bisector of the angle between the two segments;
- shifting the overall clotoid–circle–clotoid blend along the direction $\bar{v}$ given by bisector of the angle between the two segments of the quantity needed to lead ΔR to zero.

All these steps must be designed in order to get the desired distance $d_{\max}$ from the via-point. To this aim, we must consider that, imposing the value of τ in the Eqs. (32)–(34), the only unknown parameter is A, then the problem can be reformulated in the following way: to find the parameter A in order that the overall clotoid–arc–clotoid blend, after the shifting along the versor $\bar{v}$ is at a distance $d_{\max}$ from the via-point; then, we must design the parameter A. It can be shown that ΔR in Fig. 22 is given by

$$\Delta R = \frac{A\sqrt{2\tau}}{24} \sum_{i=1}^{\infty} \left[(-1)^{i+1} 6 \frac{\tau^{2i-2}}{(4i-1)(2i)!} \right] = A \cdot f_{\Delta R}(\tau). \tag{35}$$

To superimpose the clotoid blends to the two segments the shift quantity Λ in Fig. 22 is

$$\Lambda = \frac{\Delta R}{\sin\left(\frac{\pi-\gamma}{2}\right)}, \tag{36}$$

then the following relation must be true:

$$p_{x,via} - \left[c_x + (R + d_{\max} - \Lambda) \cos\left(\frac{\pi - \gamma}{2} \right) \right] = 0, \tag{37}$$

that, taking into account Eqs. (33)–(36), can be rewritten as

$$p_{x,via} - A \cdot f_{c_x}(\tau_f) - \left[A \cdot f_R(\tau_f) + d_{\max} - A \cdot \frac{f_{\Delta R}(\tau_f)}{\sin(\frac{\pi-\gamma}{2})} \right] \cos\left(\frac{\pi - \gamma}{2} \right) = 0, \tag{38}$$

leading to a linear equation in the unknown parameter A.

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press

References

1. M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables* (Dover, New York, 1964).
2. G. Antonelli, F. Caccavale and P. Chiacchio, "A systematic procedure for the identification of dynamic parameters of robot manipulators," *Robotica* **17**, 427–435 (1999).
3. G. Antonelli, S. Chiaverini, M. Palladino, G. Gerio and G. Renga, "SmartMove4: An industrial implementation of trajectory planning for robots," *J. Ind. Robot* **34**(3), 217–224 (May 2007).
4. J. E. Bobrow, S. Dubowsky and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *Int. J. Robot. Res.* **4**(3), 3–17 (1985).
5. D. Constantinescu and E. A. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *J. Robot. Syst.* **17**(5), 233–249 (2000).
6. O. Dahl, "Path constrained robot control: Experimental evaluation," *Mechatronics (Oxford)* **4**(2), 173–198 (1994).
7. O. Dahl, "Path-constrained robot control with limited torques-experimental evaluation," *Robot. Autom. IEEE Trans.* **10**(5), 658–669 (1994).
8. O. Dahl and L. Nielsen, "Torque-limited path following by online trajectory time scaling," *Robot. Autom. IEEE Trans.* **6**(5), 554–561 (1990).
9. A. Gasparetto and V. Zanotto, "A technique for time-jerk optimal planning of robot trajecotries," *Robot. Comput.-Integr. Manuf. 2008* **24**, 415–426 (2008).
10. A. Kasiriski and J. Wencel, "Constraints Satisfying Trajectory Tracking Algorithm for the 3 dof Manipulator," *Proceedings of the Fifth International Workshop on Robot Motion and Control*, Dymaczewo, Poland (23–25 June 2005) pp. 111–116.
11. K. J. Kyriakopoulos and G. N. Saridis, "Minimum Jerk Path Generation," *Robotics and Automation, 1988. Proceedings of the 1988 IEEE International Conference on* (1988) pp. 364–369.
12. T. T. Lee and C. L. Shih, "An Approach for Robot Dynamic Motion Planning with Control Torquesand Obstacles Constraints," *Sixteenth Annual Conference of IEEE Industrial Electronics Society*, Pacific Grove, CA (27–30 Nov. 1990), vol. 1, pp. 433–438.
13. S. Macfarlane. On-Line Smooth Trajectory Planning for Manipulators *PhD thesis* (New Brunswick, Canada: The University of New Brunswick, 2001).
14. S. Macfarlane and E. A. Croft, "Jerk-bounded robot trajectory planning-design for real-time applications," *IEEE Trans. Robot. Autom.* **19**(1), 42–52 (2003).
15. K. L. Moore, M. Ghosh and Y. Q. Chen, "Spatial-based iterative learning control for motion control applications," *Meccanica* **42**(2), 167–175 (2007).
16. J. Moreno-Valenzuela, "Time-scaling of trajectories for point-to-point robotic tasks," *ISA Trans.* **45**(3), 407–418 (2006).
17. J. Moreno-Valenzuela, "Tracking Control of On-line Time-Scaled Trajectories for Robot Manipulators Under Constrained Torques," *Proceedings 2006 IEEE International Conference on Robotics and Automation*, Orlando, FL (May 2006) pp. 19–24.
18. S. Munasinghe, M. Nakamura, T. Iwanaga, S. Goto and N. Kyura, "Precise, Jerk-Free Contouring of Industrial Robot Arms Withtrajectory Allowance Under Torque and Velocity Constraints," *Industrial Electronics Society, 2001. IECON'01. The 27th Annual Conference of the IEEE*, vol. 1 (2001).
19. A. Piazzi, C Guarino Lo Bianco and M. Romano, "η^3-splines for the smooth path generator of wheeled mobile robots," *IEEE Trans. Robot.* **23**(5), 1089–1095 (2007).
20. L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed. (Springer, Heidelberg, 1997).
21. T. A. Rieswijk, G. G. Brouwn and G. Honderd, "A robust and efficient approach for the time optimization of pathconstrained motions of robotic manipulators incorporating actuatortorque and jerk constraints," *Proceedings of the 1992 IEEE International Symposium on Intelligent Control*, Glasgow, Scotland, (Aug. 1992) pp. 507–513.
22. L. Sciavicco and B. Siciliano, *Modeling and Control of Robot Manipulators*, 2nd edn. (Springer-Verlag, London, 2000).
23. K. G. Shin and N. D. McKay, "Minimum time control of robotic manipulators along specified paths," *IEEE Trans. Robot. Autom.* **30**(6), 531–541 (1985).
24. J.-J. E. Slotine and H. S. Yang, "Improving the efficiency of time-optimal path-following algorithms," *IEEE Trans. Robot. Autom.* **34**, 118–124 (1989).

https://doi.org/10.1017/S026357471000024X Published online by Cambridge University Press