UNIVERSITÀ DI PARMA

DOTTORATO DI RICERCA IN
"TECNOLOGIE DELL'INFORMAZIONE"

CICLO XXXV

Cartesian trajectory planners for robotic applications

Coordinatore:
Chiar.mo Prof. Marco Locatelli

Tutor:
Chiar.mo Prof. Corrado Guarino Lo Bianco

Dottorando: Andrea Tagliavini

Anni 2019/2023

UNIVERSITÀ DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXXV Ciclo

Cartesian trajectory planners for robotic applications

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Corrado Guarino Lo Bianco

Dottorando: *Andrea Tagliavini*

2019/2023

To Elena,
for supporting me
in all circumstances.

Contents

List of Figures

List of Tables

Introduction

Autonomous systems can be found in many different environments: from highly technological industry to everyday life. All robots share common goals: they must assure the highest possible precision, efficiency and repeatability. Some well known examples are represented by industrial robots, computer numerically controlled (CNC) machines and laser guided vehicles (LGV). One of the keystones in order to pursue the aforementioned goals resides in motion planning, in other words, the need to impose a trajectory that allows the robot to move while maintaining some user-defined criteria. These features mainly concern smoothness and efficiency, in order to improve repeatability and productivity, which could be settled by using different kind of planning primitives. One of the most used strategy needs the robot to stop or drastically reduce its velocity to overcome direction changes, whose principal drawback is the dilation of working time. For this reason in latest years many researchers have investigated a wide range of possibilities trying to improve both precision tracking and optimality criteria.

This thesis will deal with the aforementioned problem by developing a trajectory planning primitive able to guarantee high smoothness while restraining the computational burden. To achieve such goals the path-velocity decomposition has been used: practically trajectory and velocity primitives have been separately considered. Moreover the first problem, trajectory planning, has been itself divided in path and orientation in order to deliver a more complete approach that can be used for several kind of robots. The main applicative case study in this thesis will be a six degree of freedom anthropomorphic manipulator.

This thesis is divided in two parts:

I. In the first part a path primitive, η^{3D}-splines, is theorized, developed and finally tested. In comparison with existing literature mainly composed of $\mathscr{C}^1$ and $\mathscr{C}^2$ planners like the ones proposed in [1–4] this work will deliver a planner able to guarantee $\mathscr{C}^3$ continuity along the whole trajectory. Another main advantage of the η^{3D}-splines is the capability of evaluating their parameters within the sample time of most robotic systems. This useful combination allows a wide range of possibilities, for example planning a precise trajectory to avoid objects, emulate other geometric primitives and even re-evaluating trajectory on the fly. More details on the existent literature and the novelties introduced can be found in section 1.

II. The second part is relative to an orientation primitive conceived for any three-dimensional rotational frame. After considering different representations, well summarized by [5], the end effector orientation is expressed by using an Euler quaternion. Moreover, in order to obtain a more general purpose strategy, each quaternion component is planned singularly and is later normalized. By exploiting this feature any one-dimensional primitive can be used to plan each of the quaternion components, still obtaining a high continuity degree. As well as for the path planner, the main advantage of the orientation primitive resides in combining low computational burden with high order continuity, which allows to obtain a flexible and robust orientation planner. More details on the existent literature and the novelties introduced can be found in section 4.

Part I

Path Planner

Chapter 1

State of the Art

1.1 Cartesian Paths

Cartesian paths for robotic systems must be planned by accounting for their continuity properties. Indeed, non-smooth primitives worsen the controllers performances and simultaneously stress the system mechanics. The path geometric continuity problem is widely discussed in the literature and it is managed through apposite planners. Early works were focused on the generation of planar curves mainly conceived for autonomous mobile robots. The emphasis was initially posed on the generation of minimum length paths between assigned points [1, 2, 6, 7]. The problem has been continuously revised along the years by also considering alternative cost indexes [8]. The resulting composite routes, obtained by joining linear segments and circular arcs, were generated so as to guarantee the continuity of the path tangent. In the same years, other authors [3, 4, 9–12] introduced the concept of second order geometric continuity, thus including the path smoothness among the criteria to be considered: not only the path tangent, but also its curvature must be continuous. The second order continuity was achieved by means of apposite primitives like clothoids, polar curves or cubic spirals. Later, to the same purpose, other flexible path primitives appeared in the literature. For example, the Bezier curves adopted in [13–15], the B-splines proposed in [16], and, finally, the η^2-splines devised in [17].

The common denominator of the papers just cited is that smoothness reasons motivate the adoption of continuous curvature paths. However, in [18] it was shown that the smooth control of unicycle vehicles actually requires an additional continuity level: the generation of continuous control signals necessarily imposes paths whose curvature-derivative is continuous, so that a novel path planning primitive, named η^3-splines [19, 20], was developed, later followed by η^4-splines [21].

In more recent years, the smooth path planning problem has been extended to three-dimensional (3D) Cartesian curves. The fields of application of 3D trajectories are substantially two: the generation of collision-free routes in cluttered environments [22, 23] and the generation of paths with specific geometric characteristics, like the ones required, for example, by Computer Numerical Control (CNC) machines [24–26] or for arc welding and laser cutting applications. In the first case, the commonly adopted solution is based on the generation of collision-free, discrete, Cartesian paths, whose points are chosen by means of algorithms – like, for example, Rapidly-exploring Random Trees (RRT or RRT*) – which optimize a proper cost index. The desired degree of smoothness is typically achieved by means of spline trajectories, which interpolate the given points directly in the Cartesian space or, alternatively, in the configuration space. For such class of problems, the actual shape of the Cartesian path is not rigidly imposed in advance [27–30].

A similar solution can also be adopted for the management of the second class of problems. In case of applications which allow relatively high computational times, a path with the prescribed geometry can be generated by means of a dense sequence of pass-through points [31–33]. In all the other cases, an actual Cartesian path, with the required geometric continuity, must be planned. For a long time, linear segments and circular arcs were the sole path primitives considered in robotic contexts. As earlier pointed out, they only allow continuous-tangent paths. In advanced applications, like the ones involving CNC machines, such limit can not be admitted, so that, recently, many works concerning the generation of smooth curves have appeared in the literature: Bezier curves are used in [34, 35] for the synthesis of continuous-curvature paths, while B-splines are adopted in [36] for a jerk bounded application requiring continuous curvature derivatives.

1.2 Corner Smoothing

Complex geometric profiles are handled by CNC machines by dividing them into sets of linear and circular segments. If such primitives are not properly joined, the working tool must stop at the junction points in order to fulfill the dynamic limits of the actuators and to guarantee a good tracking.

Two alternative strategies can be adopted in order to mitigate possible problems: an optimal feed-rate must be planned for the sequence of points [37, 38] or a corner smoothing method must be adopted. In the literature, several papers investigated the latter solution. Some works propose a global approach involving the whole composite path [39, 40], but the majority of them devise local strategies focusing on the management of single junction points. All techniques replace corner points of the original trajectories with appropriate junction curves. The main primitives used to this purpose are the B-splines [36, 41–43] and the Bezier curves [35, 44], but also other alternative solutions have been proposed such as, for example, the parametric curves [45] and the Pythagorean Hodographs [46, 47].

It is worth to mention that junction curves, by themselves, are not sufficient to prevent rough movements of the actuators, which could cause quality losses and machinery failures. Better performances can be achieved by also ensuring that the overall composite path is sufficiently smooth. Such result can be obtained by guaranteeing that the actuators signals are continuous together with their first and second time derivatives. Advanced approaches also impose the jerk continuity. Planning strategies often adopt the path/velocity decomposition paradigm, which allows to separately plan the timing law and the path profile. The trajectory smoothness is achieved by assuming a timing law with a high degree of continuity and a path with proper geometric characteristics. To this purpose, the concept of geometric continuity is fundamental. A path admits a $\mathscr{G}^n$ geometric continuity if its n-th derivative w.r.t. the arc length parameter is continuous. Many papers in the literature propose, for the solution of the corner smoothing problem, $\mathscr{G}^2$ curves [35, 43–46], i.e., curves admitting continuous tangent vectors and curvatures, while just a few works allow a higher smoothness level [36, 47].

Chapter 2

η^{3D}-splines

The path planning primitive proposed in this work, named η^{3D}-splines and based on 7th order polynomials, is conceived for applications belonging to the second class of problems. They guarantee the same smoothness level of [36], but are conceived for more general application contexts. In particular, while in CNC applications the emphasis is primarily posed on the creation of junction curves which smoothly join a sequence of linear segments, for a general purpose robotic application the target is the generation of flexible curves able to connect generic primitives by guaranteeing the required continuity level. Indeed, if junction curves are not properly designed, reference signals for joint velocities, accelerations, and jerks may be discontinuous, thus worsening the systems performances. Continuity problems can be handled by stopping the manipulator at the end of each segment of the trajectory, but such solution is clearly inefficient. Conversely, η^{3D}-splines, by guaranteeing the required geometric continuity of the path, allow joint reference signals which are $\mathscr{C}^3$, so that composite trajectories can be executed with no stops.

The strengths of η^{3D}-splines, if compared with other planning primitives, are essentially two: the planning method, which allows an efficient online evaluation of the curve coefficients directly from the interpolating conditions, and their straightforward emulation capabilities.

Concerning the first point, the third-order geometric continuity can be obtained by assigning, at the start and at the end points of the curve, the desired Frenet-frames, curvatures, curvature-derivatives and torsions. Such interpolating conditions directly appear in the closed form expressions synthesized for the efficient computation of the spline coefficients: the same continuity level achieved in [36] can be reached at a negligible computational cost. This allows one to easily manage situations in which the Cartesian path must be re-planned on-the-fly during the movement – see for example [48] [49] – by simultaneously maintaining the overall geometric continuity. The planning method is more immediate and intuitive than the ones adopted for the synthesis of B-spline or Bezier curves, which require a proper choice of the control points in order to satisfy the assigned continuity properties and to model the curves shape.

As anticipated, η^{3D}-splines can also emulate other common primitives. More precisely, they exactly generate linear segments, while they can emulate, with good approximation, curves like circular arcs, clothoids, helical curves, and conic spirals. Additionally, any 7th order polynomial curve, for example a Bezier curve, satisfying the given interpolating conditions, can be replaced by an analogous η^{3D}-spline with the same coefficients and vice-versa [50]. Such emulation capabilities are carried out through a set of six independent parameters which can be used to finely shape the curve.

The emulation capabilities of η^{3D}-splines have been exploited for the implementation of a Cartesian planner entirely based on such primitive, and suited for applications in which the path must be generated on-the-fly. Such planner is able to generate, in a simple and intuitive way, complex composite paths with third-order geometric continuity.

The chapter is organized as follows. Section 2.1 shows that jerk continuous reference signals for robotic manipulators can be obtained by planning paths with third order geometric continuity. The same Section further introduces some preliminary considerations concerning 3D curves and recalls the definitions of geometric continuity. Section 2.2 proposes the closed-form equations which are used for the evaluation of the η^{3D}-splines coefficients. In the same section, two important properties of the novel primitive are enunciated. In Section 2.3, a simple method is proposed for the selection of the shaping parameters, and the emulation capabilities of the novel path primitive are discussed. In Section 2.4, the η^{3D}-splines are experimentally tested by generating a composite 3D path for an industrial manipulator. Final conclusions are drawn in Chapter 5.3.

2.1 Preliminary considerations on the geometric continuity of 3D curves

In robotic applications, paths are typically planned so as to guarantee the smoothness of the actuators reference signals. In particular, if $\mathbf{q} := [q_1\, q_2, \ldots, q_n]^T$ is the vector of the joint variables and n is the number of joints, $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ should be continuous, but advanced applications also impose the jerk continuity. Such additional continuity level is introduced to reduce the mechanical solicitations acting on the manipulator structure and to improve the controller performances. For Cartesian trajectories, such continuity requests naturally lead to equivalent requirements involving the motion of the origin of the tool frame, i.e., $\mathbf{p} := [p_x\ p_y\ p_z]^T$. The conditions which must be satisfied by a Cartesian curve in order to guarantee the continuity of the joints jerks are derived in the reminder of this section. Furthermore, some geometric expressions used in next Section 2.2 are briefly recalled.

Cartesian trajectories are typically planned by avoiding kinematic singularities, so that the Jacobian matrix of the system (a 6-DoF manipulator), i.e., $\mathbf{J}(\mathbf{q})$, in this work will be supposed non singular and the inverse kinematic function, i.e., $\mathbf{q} = \mathbf{q}(\mathbf{p})$, will be assumed continuous. By virtue of the last hypothesis, the Cartesian path continuity guarantees, in turn, the continuity of $\mathbf{q}$. Furthermore, for industrial manipulators, $\mathbf{J}(\mathbf{q})$ is a continuously differentiable function of class $\mathscr{C}^\infty$.

Linear velocities can always be evaluated through a Jacobian matrix, $\mathbf{J}(\mathbf{q})$, according to the following equation

$$\dot{\mathbf{p}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}\,. \tag{2.1}$$

Such equation can be reorganized as follows

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\,\dot{\mathbf{p}}\,. \tag{2.2}$$

According to the premises, the Jacobian matrix is not singular, so that its inverse exists. Moreover, since $\mathbf{J}(\mathbf{q}) \in \mathscr{C}^\infty$ and bearing in mind the inverse function theorem, $\mathbf{J}^{-1}(\mathbf{q})$ is certainly a continuous function. Since $\mathbf{q}$ is continuous, (2.2) makes it possible to conclude that the continuity of $\dot{\mathbf{q}}$ is achieved by assuming a continuous $\dot{\mathbf{p}}$.

The same reasoning can be used with the higher order derivatives. The differentiation of (2.1) leads to the following equations

$$\begin{aligned}\ddot{\mathbf{p}} &= \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}}\,,\\ \dddot{\mathbf{p}} &= \ddot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} + 2\dot{\mathbf{J}}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{J}(\mathbf{q})\,\dddot{\mathbf{q}}\,,\end{aligned}$$

which can be rewritten as follows

$$\ddot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\,[\ddot{\mathbf{p}} - \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}]\,, \tag{2.3}$$

$$\dddot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\,[\,\dddot{\mathbf{p}} - \ddot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} - 2\dot{\mathbf{J}}(\mathbf{q})\ddot{\mathbf{q}}]\,. \tag{2.4}$$

Since $\mathbf{J}(\mathbf{q}) \in \mathscr{C}^\infty$, (2.3) and (2.4) allow one asserting that the continuity of $\ddot{\mathbf{q}}$ and $\dddot{\mathbf{q}}$ is obtained by assuming that also $\ddot{\mathbf{p}}$ and $\dddot{\mathbf{p}}$ are continuous. Practically, if $\mathbf{p}$ is $\mathscr{C}^3$ then $\mathbf{q}$ is $\mathscr{C}^3$ as well.

Cartesian trajectories are very commonly planned by adopting the path-velocity decomposition approach [51], so that they are obtained by combining a path, $\mathbf{p}(s)$, with a timing law, $s(t)$. s is the so-called curvilinear coordinate and it coincides with the distance from the beginning of the curve, measured along the curve itself. Consequently, $s \in [0, s_f]$ for a curve whose length is s_f. Bearing in mind such considerations, the time derivatives of $\mathbf{p}$ can be written as follows

$$\dot{\mathbf{p}} = \frac{d}{dt}\mathbf{p}[s(t)] = \frac{d\mathbf{p}}{ds}\frac{ds}{dt} = \frac{d\mathbf{p}}{ds}\dot{s}\,, \tag{2.5}$$

$$\ddot{\mathbf{p}} = \frac{d^2}{dt^2}\mathbf{p}[s(t)] = \frac{d^2\mathbf{p}}{ds^2}\dot{s}^2 + \frac{d\mathbf{p}}{ds}\ddot{s}\,, \tag{2.6}$$

$$\dddot{\mathbf{p}} = \frac{d^3}{dt^3}\mathbf{p}[s(t)] = \frac{d^3\mathbf{p}}{ds^3}\dot{s}^3 + 3\frac{d^2\mathbf{p}}{ds^2}\dot{s}\ddot{s} + \frac{d\mathbf{p}}{ds}\dddot{s}\,. \tag{2.7}$$

Clearly, (2.5)–(2.7) imply that the continuity of $\dot{\mathbf{p}}$, $\ddot{\mathbf{p}}$, and $\dddot{\mathbf{p}}$ requires the continuity of $(d\mathbf{p})/(ds)$, $(d^2\mathbf{p})/(ds)^2$, and $(d^3\mathbf{p})/(ds)^3$, as well as of $\dot{s}$, $\ddot{s}$, and $\dddot{s}$. Hence, the continuity problem can be split into two separate sub-problems, the first one involving the timing law, the second one concerning the geometric characteristics of the path. In particular, for paths which are parametrized as function of the curvilinear coordinate, i.e., s, the two concepts of analytic and geometric continuity coincide, so that if $\mathbf{p}(s)$ is a continuous function, i.e., if $\mathbf{p}(s) \in \mathscr{C}^0$, then it also admits 0 order geometric continuity – or, equivalently, $\mathbf{p}(s) \in \mathscr{G}^0$. The same concept also applies to higher order derivatives, so that if $\mathbf{p}(s) \in \mathscr{C}^1$, then $[d\mathbf{p}(s)]/(ds)$ admits a geometric continuity of the first order, i.e., $\mathbf{p}(s) \in \mathscr{G}^1$, and so on. This implies that, for the problem at hand, the focus is posed on paths belonging to $\mathscr{G}^3$ in order to achieve the continuity of (2.5)–(2.7).

For the reader convenience, the reminder of this section will recall some geometric implications of the $\mathscr{G}^3$ continuity concepts. For practical reasons, curves are often defined through a function $\mathbf{p}(u) \in \mathbb{R}^3$, where $u \in [u_0, u_f]$ is a generic scalar parameter which is used instead of s. By construction [52], $\mathbf{p}'(u) := [d\mathbf{p}(u)]/(du)$ is a vector which is tangent in u to the curve. For regular curves, i.e., curves for which $\|\mathbf{p}'(u)\| > 0, \forall u \in [u_0, u_f]$, there always exists a direct, bijective relationship between u and s, which can be expressed as follows

$$s = \int_{u_0}^{u} \|\mathbf{p}'(\tau)\| \, d\tau \,. \tag{2.8}$$

Consequently, if $\mathbf{p}(u)$ is continuous in u, then $\mathbf{p}[u(s)]$ is continuous in s – $u(s)$ is the inverse function of (2.8) – and the curve is $\mathscr{G}^0$. Equation (2.8) immediately allows one writing

$$\frac{ds}{du} = \|\mathbf{p}'(u)\| \,,$$

so that the path derivative w.r.t. s can be written as follows

$$\frac{d\mathbf{p}(u)}{ds} = \frac{d\mathbf{p}(u)}{du}\frac{du}{ds} = \mathbf{p}'(u)\frac{1}{\frac{ds}{du}} = \frac{\mathbf{p}'(u)}{\|\mathbf{p}'(u)\|} = \mathbf{t}(u), \tag{2.9}$$

Practically, the $\mathscr{G}^1$ continuity implies that unit vector $\mathbf{t}(u)$, which is by construction tangent to the curve (see also Fig. 2.1), is a continuous function of u.

For which concerns the second order geometric continuity, i.e., the continuity of $(d^2\mathbf{p})/(ds^2)$, the following equation is obtained by differentiating (2.9):

$$\frac{d^2\mathbf{p}(u)}{ds^2} = \frac{d}{ds}\left[\frac{d\mathbf{p}(u)}{ds}\right] = \frac{d}{ds}[\mathbf{t}(u)] = \kappa(u)\mathbf{n}(u). \tag{2.10}$$

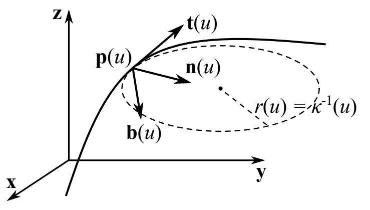


Figure 2.1: A generic curve $\mathbf{p}(u)$ (solid line), together with its Frenet frame and its osculating circle (dashed line) shown for a generic u.

The last equality in (2.10) is due to the first Frenet-Serret equation [52]. $\mathbf{n}(u)$ is the *normal unit vector* which points toward the center of the osculating circle (see also Fig. 2.1), i.e., the circle which best approximates the curve in u. $\kappa(u)$ is the *curvature* in u, i.e., it is the reciprocal of the radius of the osculating circle. They are defined as follows [52]

$$\mathbf{n}(u) = \frac{[\mathbf{p}'(u) \times \mathbf{p}''(u)] \times \mathbf{p}'(u)}{\|\mathbf{p}'(u) \times \mathbf{p}''(u)\| \, \|\mathbf{p}'(u)\|}, \tag{2.11}$$

$$\kappa(u) = \frac{\|\mathbf{p}'(u) \times \mathbf{p}''(u)\|}{\|\mathbf{p}'(u)\|^3}. \tag{2.12}$$

Evidently, $\mathbf{n}(u)$ is well defined if the curve is *biregular*, i.e., if it satisfies condition $\|\mathbf{p}'(u) \times \mathbf{p}''(u)\| > 0, \forall u \in [u_0, u_f]$. By virtue of (2.10), a curve is $\mathscr{G}^2$ if $\mathbf{n}(u)$ and $\kappa(u)$ are continuous functions.

By further differentiating (2.10), the following equation is obtained

$$\begin{aligned}\frac{d^3\mathbf{p}(u)}{ds^3} &= \frac{d}{ds}[\kappa(u)\mathbf{n}(u)] = \frac{d\kappa(u)}{ds}\mathbf{n}(u) + \kappa(u)\frac{d\mathbf{n}(u)}{ds} \\ &= \frac{\kappa'(u)}{\|\mathbf{p}'(u)\|}\mathbf{n}(u) + \kappa(u)\frac{d\mathbf{n}(u)}{ds}.\end{aligned} \tag{2.13}$$

The second Frenet-Serret equation [52] asserts that

$$\frac{d\mathbf{n}(u)}{ds} = -\kappa(u)\mathbf{t}(u) + \tau(u)\mathbf{b}(u), \tag{2.14}$$

where

$$\mathbf{b}(u) := \mathbf{t}(u) \times \mathbf{n}(u) = \frac{\mathbf{p}'(u) \times \mathbf{p}''(u)}{\|\mathbf{p}'(u) \times \mathbf{p}''(u)\|} \tag{2.15}$$

is the so called *binormal unit vector* and

$$\tau(u) = \frac{[\mathbf{p}'(u) \times \mathbf{p}''(u)] \cdot \mathbf{p}'''(u)}{\|\mathbf{p}'(u) \times \mathbf{p}''(u)\|^2} \tag{2.16}$$

is the *torsion* of the curve in u. Unit vectors $\mathbf{t}(u)$, $\mathbf{n}(u)$, and $\mathbf{b}(u)$ are each other orthogonal and form the so-called *Frenet frame* associated to point u along the curve. From (2.13) and (2.14) it immediately descends that the $\mathscr{G}^3$ continuity is achieved if a curve is $\mathscr{G}^2$, i.e., $\mathbf{t}(u)$, $\mathbf{n}(u)$, $\kappa(u)$ are continuous, and, furthermore, if the continuity is also guaranteed for

$$\begin{aligned}\frac{d\kappa(u)}{ds} = \frac{\kappa'(u)}{\|\mathbf{p}'(u)\|} &= \frac{[\mathbf{p}'(u) \times \mathbf{p}''(u)] \cdot [\mathbf{p}'(u) \times \mathbf{p}'''(u)]}{\|\mathbf{p}'(u) \times \mathbf{p}''(u)\| \, \|\mathbf{p}'(u)\|^4} \\ &- 3[\mathbf{p}'(u) \cdot \mathbf{p}''(u)] \frac{\|\mathbf{p}'(u) \times \mathbf{p}''(u)\|}{\|\mathbf{p}'(u)\|^6} \\ &= \mathbf{b}(u) \cdot \frac{\mathbf{p}'(u) \times \mathbf{p}'''(u)}{\|\mathbf{p}'(u)\|^4} - 3\kappa(u) \frac{\mathbf{p}'(u) \cdot \mathbf{p}''(u)}{\|\mathbf{p}'(u)\|^3}\end{aligned} \tag{2.17}$$

and for $\tau(u)$. By virtue of (2.9), (2.11), (2.12), and (2.16), the $\mathscr{G}^3$ continuity imposes that $\mathbf{p}(u)$ and its derivatives w.r.t. u, up to the third order, must be continuous, i.e., $\mathbf{p}(u) \in \mathscr{C}^3$.

Summarizing, (2.2)–(2.7) allow one asserting that the continuity of $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, and $\dddot{\mathbf{q}}$ can be obtained by planning $\mathscr{G}^3$ curves, i.e., by imposing that $\mathbf{t}(u)$, $\mathbf{n}(u)$, $\mathbf{b}(u)$, $\kappa(u)$, $\kappa'(u)$, and $\tau(u)$ are continuous functions, and, moreover, by generating a timing law such that the continuity is also guaranteed for s, $\dot{s}$, $\ddot{s}$, and $\dddot{s}$. The timing law generation problem is not addressed in this work, but possible approaches can be found in the literature (see for example [53]).

2.2 The η^{3D}-splines

The proposed primitive is based on a 7th order vector polynomial defined as follows

$$\mathbf{p}(u) := \boldsymbol{\chi}_0 + \boldsymbol{\chi}_1 u + \boldsymbol{\chi}_2 u^2 + \boldsymbol{\chi}_3 u^3 + \boldsymbol{\chi}_4 u^4 + \boldsymbol{\chi}_5 u^5 + \boldsymbol{\chi}_6 u^6 + \boldsymbol{\chi}_7 u^7, \tag{2.18}$$

where $u \in [0,1]$, while $\boldsymbol{\chi}_i := [\alpha_i \, \beta_i \, \gamma_i]^T \in \mathbb{R}^3$ are properly defined vector coefficients. The $\mathscr{G}^3$ continuity is certainly achieved $\forall u \in (0,1)$ since, for constant values of $\boldsymbol{\chi}_i$,

Table 2.1: Interpolating conditions for the η^{3D}-spline curve. (2.50)

A	B
$\mathbf{p}(0) = \mathbf{p}_A$	$\mathbf{p}(1) = \mathbf{p}_B$
$[\mathbf{t}(0)\ \mathbf{n}(0)\ \mathbf{b}(0)] = [\mathbf{t}_A\ \mathbf{n}_A\ \mathbf{b}_A]$	$[\mathbf{t}(1)\ \mathbf{n}(1)\ \mathbf{b}(1)] = [\mathbf{t}_B\ \mathbf{n}_B\ \mathbf{b}_B]$
$\kappa(0) = \kappa_A$	$\kappa(1) = \kappa_B$
$\tau(0) = \tau_A$	$\tau(1) = \tau_B$
$\frac{d\kappa}{ds}(0) = \bar{\kappa}_A$	$\frac{d\kappa}{ds}(1) = \bar{\kappa}_B$

the derivatives of $\mathbf{p}(u)$ of any order are continuous in such interval. However, this work aims at generating $\mathscr{G}^3$ composite paths obtained by joining several curves, so that coefficients $\boldsymbol{\chi}_i$ must be assigned so as to also impose the $\mathscr{G}^3$ continuity in the junction points between adjacent curves. According to the discussion in Section 2.1, such result can be achieved by imposing that, at the end of each curve, $\mathbf{t}$, $\mathbf{n}$, $\mathbf{b}$, κ, κ', and τ coincide with the homologous terms of the subsequent one. Practically, the $\mathscr{G}^3$ continuity is obtained if the interpolating conditions shown in Table 2.1 can be arbitrarily imposed to (2.18) (subscripts A and B indicate the assigned interpolating conditions at the beginning and at the end of the curve, respectively). Evidently, such boundary conditions can be imposed by solving an appropriate system of equations but, more conveniently, this work proposes closed form expressions for the immediate and efficient evaluation of the $\boldsymbol{\chi}_i$ parameters. In particular, the coefficients of the seventh order polynomial curve, satisfying the boundary conditions specified in

Table 2.1, can be immediately obtained by means of the following expressions

$$\boldsymbol{\chi}_0 = \mathbf{p}_A, \tag{2.19}$$

$$\boldsymbol{\chi}_1 = \eta_1 \mathbf{t}_A, \tag{2.20}$$

$$\boldsymbol{\chi}_2 = (1/2)[\kappa_A \eta_1^2 \mathbf{n}_A + \eta_3 \mathbf{t}_A], \tag{2.21}$$

$$\boldsymbol{\chi}_3 = \frac{1}{6}[\kappa_A \tau_A \mathbf{b}_A \eta_1^3 + \left(\bar{\kappa}_A \eta_1^2 + 3\,\kappa_A \eta_1 \eta_3\right)\mathbf{n}_A] + \eta_5 \mathbf{t}_A, \tag{2.22}$$

$$\begin{aligned}\boldsymbol{\chi}_4 = &-(2/3)\,\kappa_A \tau_A \mathbf{b}_A \eta_1^3 - (1/6)\,\kappa_B \tau_B \mathbf{b}_B \eta_2^3 \\ &-(1/3)\left[\eta_1^2\,(2\,\bar{\kappa}_A + 15\,\kappa_A) + 6\,\kappa_A \eta_1 \eta_3\right]\mathbf{n}_A \\ &-(1/6)\left[\eta_2^2\,(\bar{\kappa}_B - 15\,\kappa_B) + 3\,\kappa_B \eta_2 \eta_4\right]\mathbf{n}_B \\ &-(20\,\eta_1 + 5\,\eta_3 + 4\,\eta_5)\mathbf{t}_A - 35\,\mathbf{p}_A + 35\,\mathbf{p}_B \\ &-(1/2)\,(30\,\eta_2 - 5\,\eta_4 + 2\,\eta_6)\mathbf{t}_B,\end{aligned} \tag{2.23}$$

$$\begin{aligned}\boldsymbol{\chi}_5 = &\,\kappa_A \tau_A \mathbf{b}_A \eta_1^3 + \frac{1}{2}\,\kappa_B \tau_B \mathbf{b}_B \eta_2^3 \\ &+\left[\eta_1^2\,(\bar{\kappa}_A + 10\,\kappa_A) + 3\,\kappa_A \eta_1 \eta_3\right]\mathbf{n}_A \\ &+(1/2)\left[\eta_2^2\,(\kappa_B - 14\,\kappa_B) + 3\,\kappa_B \eta_2 \eta_4\right]\mathbf{n}_B \\ &+(45\,\eta_1 + 10\,\eta_3 + 6\,\eta_5)\mathbf{t}_A \\ &+(39\,\eta_2 - 7\,\eta_4 + 3\,\eta_6)\mathbf{t}_B + 84\,\mathbf{p}_A - 84\,\mathbf{p}_B,\end{aligned} \tag{2.24}$$

$$\begin{aligned}\boldsymbol{\chi}_6 = &-(2/3)\,\kappa_A \tau_A \mathbf{b}_A \eta_1^3 - (1/2)\,\kappa_B \tau_B \mathbf{b}_B \eta_2^3 \\ &-(1/6)\left[\eta_1^2\,(4\,\bar{\kappa}_A + 45\,\kappa_A) + 12\,\kappa_A \eta_1 \eta_3\right]\mathbf{n}_A \\ &-(1/2)\left[\eta_2^2\,(\bar{\kappa}_B - 13\,\kappa_B) + 3\,\kappa_B \eta_2 \eta_4\right]\mathbf{n}_B \\ &-(1/2)\,(72\,\eta_1 + 15\,\eta_3 + 8\,\eta_5)\mathbf{t}_A - 70\,\mathbf{p}_A + 70\,\mathbf{p}_B \\ &-(1/2)\,(68\,\eta_2 - 13\,\eta_4 + 6\,\eta_6)\mathbf{t}_B,\end{aligned} \tag{2.25}$$

$$\begin{aligned}\boldsymbol{\chi}_7 = &\,(1/6)(\,\kappa_A \tau_A \mathbf{b}_A \eta_1^3 + \,\kappa_B \tau_B \mathbf{b}_B \eta_2^3) \\ &+(1/6)\left[\eta_1^2\,(\bar{\kappa}_A + 12\,\kappa_A) + 3\,\kappa_A \eta_1 \eta_3\right]\mathbf{n}_A \\ &+(1/6)\left[\eta_2^2\,(\bar{\kappa}_B - 12\,\kappa_B) + 3\,\kappa_B \eta_2 \eta_4\right]\mathbf{n}_B \\ &+(10\,\eta_1 + 2\,\eta_3 + \eta_5)\mathbf{t}_A \\ &+(10\,\eta_2 - 2\,\eta_4 + \eta_6)\mathbf{t}_B + 20\,\mathbf{p}_A - 20\,\mathbf{p}_B.\end{aligned} \tag{2.26}$$

By analyzing (2.19)–(2.26), it can be noticed that the $\boldsymbol{\chi}_i$ coefficients only depend on the interpolating conditions and on a set of six independent parameters which can be conveniently packed into vector $\boldsymbol{\eta} := [\eta_1\, \eta_2\, \eta_3\, \eta_4\, \eta_5\, \eta_6]^T$, where $\eta_1, \eta_2 \in \mathbb{R}^+$ and $\eta_3, \eta_4, \eta_5, \eta_6 \in \mathbb{R}$. Consequently, given the interpolating conditions and vector $\boldsymbol{\eta}$, the η^{3D}-splines coefficients can be obtained at a negligible computational cost. The selection of $\boldsymbol{\eta}$ influences the curve shape and will be discussed in Section 2.3.

Since the procedure for the synthesis of (2.19)–(2.26) only consists in solving a linear system it is not proposed but, more conveniently, the curve properties and its strengths are pointed out in the following by means of two propositions. In particular, Proposition 1 asserts that η^{3D}-splines can generate, through a proper choice of $\boldsymbol{\eta}$, all possible 7th order polynomial curves which satisfy the assigned interpolating conditions. The proof of Proposition 1, points out another important characteristic which is stated in Proposition 2: the interpolating conditions, which are normally assigned so as to guarantee the $\mathscr{G}^3$ continuity, are always satisfied independently from the choice of $\boldsymbol{\eta}$. This implies that the curve can be modeled by means of $\boldsymbol{\eta}$, but the choice of $\boldsymbol{\eta}$ does not affect the continuity properties of the combined path.

Proposition 1 *Given a generic 7th order polynomial $\mathbf{p}(u)$, expressed by means of (2.18), which satisfies the interpolating conditions of Table 2.1, through a proper choice of vector $\boldsymbol{\eta}$ it is always possible to find an η^{3D}-spline, namely $\mathbf{p}_{\boldsymbol{\eta}}(u)$, such that $\mathbf{p}_{\boldsymbol{\eta}}(u) = \mathbf{p}(u)$.*

Proof – The coefficients of a generic 7th order spline $\mathbf{p}(u)$ satisfying the interpolating conditions specified in Table 2.1 will be indicated as $\widetilde{\boldsymbol{\chi}}_i, i = 0, 1, \dots, 7$, while the coefficients of an η^{3D}-spline $\mathbf{p}_{\boldsymbol{\eta}}(u)$ satisfying the same interpolating conditions will be indicated as $\boldsymbol{\chi}_i$. In order to prove Property 1, it is necessary to demonstrate that, through a proper choice of vector $\boldsymbol{\eta}$, it is always possible to obtain $\mathbf{p}_{\boldsymbol{\eta}}(u) = \mathbf{p}(u)$ or, equivalently, $\boldsymbol{\chi}_i = \widetilde{\boldsymbol{\chi}}_i, i = 0, 1, \dots, 7$.

Interpolating condition $\mathbf{p}(0) = \mathbf{p}_A$ is evidently satisfied for a curve like (2.18) if $\widetilde{\boldsymbol{\chi}}_0 = \mathbf{p}_A$. According to (2.19), the first coefficient of $\mathbf{p}_{\boldsymbol{\eta}}(u)$ is given by $\boldsymbol{\chi}_0 = \mathbf{p}_A$, so that, evidently, $\boldsymbol{\chi}_0 = \widetilde{\boldsymbol{\chi}}_0$.

By virtue of (2.9), the following condition applies for any generic curve

$$\mathbf{p}'(u) = \mathbf{t}(u) \left\|\mathbf{p}'(u)\right\|. \tag{2.27}$$

If the same curve satisfies initial condition $\mathbf{t}(0) = \mathbf{t}_A$ then, for $u = 0$, (2.27) can be written as follows

$$\mathbf{p}'(0) = \mathbf{t}_A \left\|\mathbf{p}'(0)\right\|. \tag{2.28}$$

By differentiating (2.18), it is possible to evince that the first derivative of $\mathbf{p}(u)$, computed for $u = 0$, is given by $\mathbf{p}'(0) = \widetilde{\boldsymbol{\chi}}_1$, so that from (2.28) it descends that

$$\mathbf{p}'(0) = \widetilde{\boldsymbol{\chi}}_1 = \mathbf{t}_A \left\|\mathbf{p}'(0)\right\|. \tag{2.29}$$

Evidently, coefficient $\boldsymbol{\chi}_1$ of $\mathbf{p}_{\boldsymbol{\eta}}(u)$ is given by (2.20). By assuming

$$\eta_1 = \left\|\mathbf{p}'(0)\right\| \tag{2.30}$$

condition $\boldsymbol{\chi}_1 = \widetilde{\boldsymbol{\chi}}_1$ is satisfied.

Any orthogonal Frenet frame is a base for the 3D Cartesian space, so that the following expression is certainly true

$$\mathbf{p}''(0) = \alpha \mathbf{t}_A + \beta \mathbf{n}_A + \gamma \mathbf{b}_A, \tag{2.31}$$

where α, β, and γ are proper scalar.

Bearing in mind (2.28) and (2.31), if a generic curve satisfies condition $\mathbf{n}(0) = \mathbf{n}_A$, then (2.11), after a few algebraic manipulations, can be written as follows

$$\mathbf{n}_A = \frac{[\mathbf{p}'(0) \times \mathbf{p}''(0)] \times \mathbf{p}'(0)}{\|\mathbf{p}'(0) \times \mathbf{p}''(0)\| \, \|\mathbf{p}'(0)\|} = \frac{\beta \mathbf{n}_A + \gamma \mathbf{b}_A}{\|\beta \mathbf{b}_A - \gamma \mathbf{n}_A\|}.$$

Evidently, such expression is true only if

$$\gamma = 0. \tag{2.32}$$

If the curve also fulfills $\kappa(0) = \kappa_A$, then (2.12) – together with (2.28), (2.31), and (2.32) – allows one writing the following expression

$$\kappa_A = \frac{\|\mathbf{p}'(0) \times \mathbf{p}''(0)\|}{\|\mathbf{p}'(0)\|^3} = \frac{\|\mathbf{p}'(0)\| \, \|\mathbf{t}_A \times (\alpha \mathbf{t}_A + \beta \mathbf{n}_A)\|}{\|\mathbf{p}'(0)\|^3} = \frac{\beta}{\|\mathbf{p}'(0)\|^2}$$

and, consequently,

$$\beta = \kappa_A \left\|\mathbf{p}'(0)\right\|^2. \tag{2.33}$$

By virtue of (2.32) and (2.33), for any curve which satisfies the conditions in Table 2.1, (2.31) assumes the following form

$$\mathbf{p}''(0) = \alpha\mathbf{t}_A + \kappa_A \left\|\mathbf{p}'(0)\right\|^2 \mathbf{n}_A. \tag{2.34}$$

On the other side, for a generic polynomial curve $\mathbf{p}(u)$ like (2.18), the following expression holds

$$\mathbf{p}''(0) = 2\widetilde{\boldsymbol{\chi}}_2, \tag{2.35}$$

so that the interpolating conditions are satisfied by imposing

$$\widetilde{\boldsymbol{\chi}}_2 = \frac{1}{2}(\alpha\mathbf{t}_A + \kappa_A \left\|\mathbf{p}'(0)\right\|^2 \mathbf{n}_A). \tag{2.36}$$

From (2.21), it descends that the same result can be achieved for $\mathbf{p}_{\boldsymbol{\eta}}(u)$ by acting on $\boldsymbol{\eta}$. In particular, condition $\boldsymbol{\chi}_2 = \widetilde{\boldsymbol{\chi}}_2$ is satisfied by imposing (2.30) and by further assigning

$$\eta_3 = \alpha. \tag{2.37}$$

A similar reasoning can be used for $\boldsymbol{\chi}_3$. Vector $\mathbf{p}'''(0)$ can be expressed through its components in the Frenet frame

$$\mathbf{p}'''(0) = \hat{\alpha}\mathbf{t}_A + \hat{\beta}\mathbf{n}_A + \hat{\gamma}\mathbf{b}_A \tag{2.38}$$

where $\hat{\alpha}$, $\hat{\beta}$, and $\hat{\gamma}$ are proper scalars. If a curve fulfills $\tau(0) = \tau_A$, by virtue of (2.16) the following condition holds

$$\tau_A = \frac{[\mathbf{p}'(0) \times \mathbf{p}''(0)] \cdot \mathbf{p}'''(0)}{\|\mathbf{p}'(0) \times \mathbf{p}''(0)\|^2}. \tag{2.39}$$

By further considering (2.28), (2.34), and (2.38), and after a few algebraic manipulations, (2.39) can be rearranged as follows

$$\tau_A = \frac{\hat{\gamma}}{\|\mathbf{p}'(0)\|^3 \kappa_A},$$

which implies that the interpolating conditions are fulfilled if the following expression is satisfied

$$\hat{\gamma} = \tau_A \left\|\mathbf{p}'(0)\right\|^3 \kappa_A. \tag{2.40}$$

By imposing $\frac{d\kappa}{ds}(0) = \bar{\kappa}_A$ to (2.17), evaluated for $u = 0$, one can write

$$\bar{\kappa}_A = \mathbf{b}_A \cdot \frac{\mathbf{p}'(0) \times \mathbf{p}'''(0)}{\|\mathbf{p}'(0)\|^4} - 3\kappa_A \frac{\mathbf{p}'(0) \cdot \mathbf{p}''(0)}{\|\mathbf{p}'(0)\|^3}. \tag{2.41}$$

By considering (2.28), (2.34), and (2.38), (2.41) simplifies as follows

$$\bar{\kappa}_A = \frac{\hat{\beta}}{\|\mathbf{p}'(0)\|^3} - 3\kappa_A \frac{\alpha}{\|\mathbf{p}'(0)\|^2},$$

so that, necessarily, $\hat{\beta}$ must assume the following structure

$$\hat{\beta} = \bar{\kappa}_A \left\|\mathbf{p}'(0)\right\|^3 + 3\kappa_A \alpha \left\|\mathbf{p}'(0)\right\|. \tag{2.42}$$

Expressions (2.40) and (2.42) are valid for generic curves, but can be specialized for polynomial functions. In particular, from (2.18) it descends that

$$\mathbf{p}'''(0) = 6\widetilde{\boldsymbol{\chi}}_3, \tag{2.43}$$

so that, bearing in mind (2.38), (2.40), and (2.42), the following expression for $\widetilde{\boldsymbol{\chi}}_3$ is obtained

$$\widetilde{\boldsymbol{\chi}}_3 = \frac{1}{6}\hat{\alpha}\mathbf{t}_A + \left(\frac{1}{6}\bar{\kappa}_A \left\|\mathbf{p}'(0)\right\|^3 + \frac{1}{2}\kappa_A \alpha \left\|\mathbf{p}'(0)\right\|\right)\mathbf{n}_A + \frac{1}{6}\tau_A \left\|\mathbf{p}'(0)\right\|^3 \kappa_A \mathbf{b}_A. \tag{2.44}$$

The same value can be assumed by an η^{3D}-spline by assigning $\eta_5 = \hat{\alpha}/6$, and by recalling that (2.30) and (2.37) simultaneously hold: condition $\boldsymbol{\chi}_3 = \widetilde{\boldsymbol{\chi}}_3$ is certainly satisfied.

An analogous procedure can be used to demonstrate that $\boldsymbol{\chi}_i = \widetilde{\boldsymbol{\chi}}_i$, for $i = 4,5,6,7$. To this purpose, it is necessary to consider the interpolating conditions at the curve endpoint ($u = 1$). The process requires much more algebraic manipulations, so that, for space reasons, it is omitted.

It is worth highlighting that the demonstration also allows one asserting that the choice of parameters $\boldsymbol{\eta}$ never affect interpolating conditions, which are always satisfied independently from the values it assumes: this property is exploited in the thesis in order to obtain curves with the desired shape. ∎

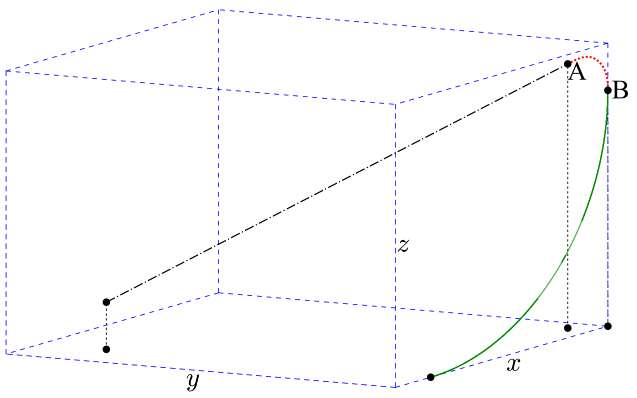


Figure 2.2: An η^{3D}-spline (dotted red line) is used to smoothly join a linear segment (dash-dotted black line) with a circular arch (solid green line). The composite path is $\mathscr{G}^3$.

Proposition 2 *Any path primitive $\mathbf{p}_{\boldsymbol{\eta}}(u)$, obtained by selecting the coefficients of (2.18) through (2.19)–(2.26), always satisfies the interpolating conditions of Table 2.1, independently from the choice of $\boldsymbol{\eta}$.*

Proof – It is an immediate consequence of the previous demonstration.

Propositions 1 and 2 suggest two possible applications for the η^{3D}-splines. For example, the novel planning primitive can be used to generate composite $\mathscr{G}^3$-paths or, alternatively, through an appropriate choice of vector $\boldsymbol{\eta}$ (see Section 2.3), for the emulation of 3D curves.

In industrial contexts, the first application is probably the most common one. An example case is proposed in Fig. 2.2: a composite $\mathscr{G}^3$-path is easily obtained by joining a linear segment (dash-dotted black line) and a circular arc (solid green line), generically located in a 3D environment, through the η^{3D}-splines (dotted red line). The interpolating conditions for the η^{3D}-splines are directly obtained from the path primitives which need to be joined. In particular, for point A, the interpolating conditions are the same of the-end point of the linear segment, while in B they coincide with the initial ones of the circular arc. More in details, $\mathbf{p}_A$ is the end-point of the linear segment, while $\mathbf{t}_A$ coincides with its characteristic unit vector. In any point of a linear segment, including its extremes, curvature, curvature derivative, and torsion are zero, so that $\kappa_A = \overline{\kappa}_A = \tau_A = 0$. Since $\kappa_A = 0$ and $\tau_A = 0$, then $\mathbf{n}_A$ and $\mathbf{b}_A$ can be freely selected: they must only satisfy conditions $\mathbf{n}_A \cdot \mathbf{b}_A = 0$ and $\mathbf{n}_A \times \mathbf{b}_A = \mathbf{t}_A$. Concerning point B, $\mathbf{p}_B$ coincides with the starting point of the circular arc and $\mathbf{t}_B$ is the unit tangent in the same point. For a circular arc the curvature is constant and equal to r^{-1}, where r is the radius of the primitive. Consequently, $\kappa_B = r^{-1}$. Normal vector $\mathbf{n}_B$ points towards the center of the circular arc, while the binormal vector is evaluated as $\mathbf{b}_B = \mathbf{t}_B \times \mathbf{n}_B$. Curvature derivative and torsion are equal to zero, i.e., $\overline{\kappa}_B = \tau_B = 0$. According to Proposition 2, the imposition of such boundary conditions guarantees the $\mathscr{G}^3$ continuity of the composite path. However, the curve shape can still be modeled through the choice of vector $\boldsymbol{\eta}$. Details on the selection of $\boldsymbol{\eta}$ are given in next Section 2.3. More precisely, possible strategies will be proposed for the generation of generic junction profiles or in order to let η^{3D}-splines emulate other planning primitives.

2.3 Considerations on the selection of η

The curve shape can be imposed by means of $\boldsymbol{\eta}$. To this purpose, several alternative strategies can be proposed. For example, it may be chosen by means of nonlinear programming algorithms [54], in order to satisfy some given optimal criteria. However, in many practical cases, simple heuristic strategies are sufficient for the generation of curves with interesting geometric characteristics. The advantage of such heuristic rules is represented by their efficiency, which allows the online generation of complex $\mathscr{G}^3$ paths. In particular, curves with nice geometric properties can be obtained by assigning $\boldsymbol{\eta}$ as follows

$$\eta_1 = \eta_2 = s_f, \tag{2.45}$$

$$\eta_3 = \eta_4 = \eta_5 = \eta_6 = 0, \tag{2.46}$$

where s_f is the curve length.

Such rule of thumb emerged during the studies on the emulation capabilities of the η^{3D}-splines. As stated in the Introduction, $\boldsymbol{\eta}$-splines can emulate, with very good approximation, many path primitives like, for example, circular arcs and clothoids.

Let us consider a set of circular arcs with different length $s_f = (r\pi)/(2i)$, where r is the radius and $i = 1, 2, \dots, 6$. For each arc, the inner angle is clearly given by $\widetilde{\theta} := s_f/r$ (see also Fig. 2.3). The emulation capability of η^{3D}-splines can be measured through the following approximation error

$$e(s) := \min_{u \in [0,1]} \|\mathbf{p}_{\boldsymbol{\eta}}(u) - \mathbf{p}(s)\|, \tag{2.47}$$

where $\mathbf{p}_{\boldsymbol{\eta}}(u)$ is the spline curve used to emulate the actual arc, i.e., $\mathbf{p}(s)$, $s \in [0, s_f]$. Practically, $e(s)$ is the minimum Euclidean distance between $\mathbf{p}_{\boldsymbol{\eta}}(u)$ and $\mathbf{p}(s)$ measured for a given $s \in [0, s_f]$.

Very good emulation results have been verified even when the available degrees of freedom are only partially exploited. In particular, by assigning interpolating conditions compatible with an arc, $\boldsymbol{\eta}$ can be chosen as follows

$$\begin{aligned} \eta_1 &= \eta_2 = \eta^*(\widetilde{\theta}), \\ \eta_3 &= \eta_4 = \eta_5 = \eta_6 = 0, \end{aligned} \tag{2.48}$$

where $\eta^*(\widetilde{\theta})$ is found by solving the following minimax problem

$$\min_{\eta^* \in \mathscr{S}} \max_{s \in [0, s_f]} \{|e(s)|\},$$

where $\mathscr{S}$ is a proper search interval. The problem was solved for different values of r and for $i = 1, 2, \dots, 6$. The nonlinear programming algorithm always converged to optimal values of η^* which were very close to s_f (they are not reported for conciseness).

The optimization results revealed that amplitudes of the emulation errors depend on r and on $\widetilde{\theta}$. In particular, the dependence on r is perfectly linear, so that the normalized error, defined as $e_n(s) := e(s)/r$, $s \in [0, s_f]$, is only function of $\widetilde{\theta}$. Average and maximum normalized errors for the 6 values of $\widetilde{\theta}$ are listed in the first two rows of Table 2.2: they are generally small and become negligible as $\widetilde{\theta}$ decreases.

The optimal values of η^* have been subsequently used to obtain, through a least square nonlinear regression, the following analytic expression for η_1 and η_2

$$\eta_1 = \eta_2 = s_f(\alpha\widetilde{\theta}^2 + \beta\widetilde{\theta} + \gamma), \tag{2.49}$$

Table 2.2: Average and maximum normalized errors $e_n(s)$ for several values of $\widetilde{\theta}$ and $\eta_1 = \eta_2$.

$\widetilde{\theta}$	$\pi/2$	$\pi/4$	$\pi/6$	$\pi/8$	$\pi/10$	$\pi/12$
$\eta_1 = \eta_2 = s^*$						
avg	$5.5 \cdot 10^{-6}$	$7.7 \cdot 10^{-8}$	$4.6 \cdot 10^{-9}$	$3.2 \cdot 10^{-9}$	$3.1 \cdot 10^{-10}$	$5.5 \cdot 10^{-11}$
max	$9.6 \cdot 10^{-6}$	$1.3 \cdot 10^{-7}$	$9.2 \cdot 10^{-9}$	$6.4 \cdot 10^{-9}$	$1.1 \cdot 10^{-9}$	$1.6 \cdot 10^{-10}$
$\eta_1 = \eta_2 = s_f(\alpha\widetilde{\theta}^2 + \beta\widetilde{\theta} + \gamma)$						
avg	$5.2 \cdot 10^{-6}$	$2.7 \cdot 10^{-6}$	$1.7 \cdot 10^{-6}$	$8.6 \cdot 10^{-7}$	$2.7 \cdot 10^{-7}$	$2.1 \cdot 10^{-8}$
max	$9.2 \cdot 10^{-6}$	$6.7 \cdot 10^{-6}$	$4.2 \cdot 10^{-6}$	$2.2 \cdot 10^{-6}$	$6.9 \cdot 10^{-7}$	$5.3 \cdot 10^{-8}$
$\eta_1 = \eta_2 = s_f$						
avg	$3.2 \cdot 10^{-3}$	$2.1 \cdot 10^{-4}$	$4 \cdot 10^{-5}$	$1.4 \cdot 10^{-5}$	$6.0 \cdot 10^{-6}$	$4.0 \cdot 10^{-6}$
max	$7.8 \cdot 10^{-3}$	$5.0 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$3.3 \cdot 10^{-5}$	$1.4 \cdot 10^{-5}$	$7.0 \cdot 10^{-6}$

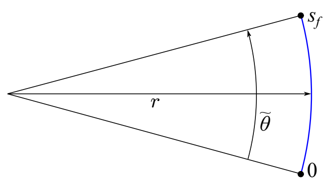


Figure 2.3: A circular arc obtained by means of the η^{3D}-splines.

where α = -0.0099417176196074, β = -0.0055734866225982, and γ = 1.00101667238653. The resulting approximation errors for the 6 test cases, obtained by still assuming (2.46) and (2.48), are shown in the third and in the fourth rows of Table 2.2. For $\widetilde{\theta} = \pi/2$, the maximum error is close 10^{-5} m for an arc whose radius is equal to $r = 1$ m. Such error is acceptable for many robotic applications and it further reduces for smaller values of r since, according to the definition of normalized error, $e(s) = r e_n(s)$.

Since α and β are very small, while $\gamma \simeq 1$, η^* is generally close to s_f. Such consideration suggested to test solutions obtained by directly assigning $\boldsymbol{\eta}$ according to (2.45) and (2.46). As shown by the fifth and the sixth rows of Table 2.2, emulation errors are still acceptable, especially for small values of $\widetilde{\theta}$.

As early anticipated, η^{3D}-splines can also emulate clothoids. Such capability has been tested by considering the same set of curves used in [55, 56] to check the emulation capabilities of Bézier curves. The set is composed by 30 clothoids, joined together so as to obtain a composite curve 6.0 m long. The composite path curvature is proportional to the path length, i.e., $\kappa(s) = s$, and each segment is 0.2 m long.

Very good results have been achieved by first imposing condition (2.46), and then by calculating the remaining 2 parameters through the following optimal problem

$$\min_{\eta_1^*, \eta_2^* \in \mathscr{S}} \max_{s \in [0, s_f]} \{|e(s)|\}.$$

The worst case error of the optimal solutions was equal to $4.655 \cdot 10^{-6}$ m. Errors were only marginally influenced by perturbations of the optimal solutions. Consequently, the worst case error only slightly increases ($5.630 \cdot 10^{-6}$ m), if the 30 optimal values of η_1^* and η_2^* are replaced by the following 2 functions obtained through a nonlinear regression

$$\eta_i(\kappa_{av}) := \lambda_{0i} + \lambda_{1i}\kappa_{av} + \lambda_{2i}\kappa_{av}^2, \quad i = 1, 2, \tag{2.50}$$

where $\kappa_{av} := (\kappa_A + \kappa_B)/2$ is the average curvature of each segment. The coefficients of (2.50) are listed in Table 2.3. The first row of Table 2.3 points out that also for clothoids, the optimal values of η_1 and η_2 are close to s_f. In facts, if $\boldsymbol{\eta}$ is chosen according to (2.45) and (2.46), the maximum error is equal to $5.060 \cdot 10^{-6}$ m for

Table 2.3: Coefficients of function (2.50)

	η_1	η_2
λ_1	0.19920352009325834053	0.20097340855985815211
λ_2	0.00067959647624348148	−0.00091915134872076114
λ_3	−0.00018934505601462691	−0.00000857636957731629

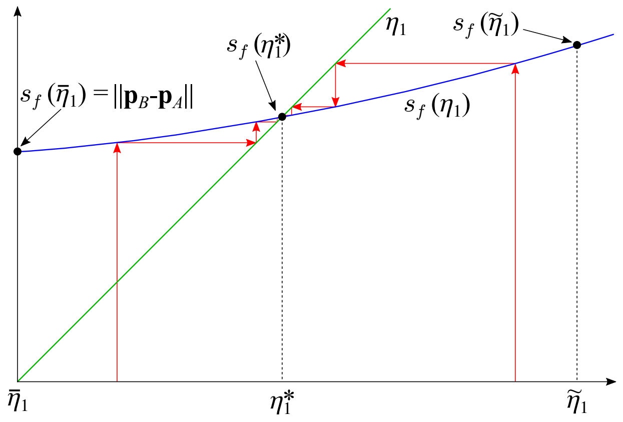
clothoids admitting $\kappa_{av} < 1$ m^{-1}, i.e., it is very close to the value obtained by solving the optimization problem.

A proper selection of $\boldsymbol{\eta}$ is also relevant when η^{3D}-splines are used to create junction curves between other primitives. According to previous discussion (2.45) and (2.46) lead to a good emulation of circular arcs and clothoids, i.e., of curves with zero or constant curvature variability $[\kappa(u) = (\kappa_B - \kappa_A)/s_f]$: apparently, such choice seems to prevent the insurgence of possible oscillatory behaviors in the curve shape. Therefore, (2.45) and (2.46) were also tested for the generation of generic profiles. The resulting curves shown, as desired, moderate and slowly variable curvatures and torsions, while oscillatory behaviors were totally absent.

The proposed selection strategy has inevitably a drawback: in case of generic interpolating conditions s_f is not known in advance, so that the imposition of (2.45) is not straightforward. In order to overcome the problem, the following iterative procedure has been used. An initial curve is generated by imposing $\eta_1 = \eta_2 = \|\mathbf{p}_A - \mathbf{p}_B\|$ and its length $\widetilde{s}_f$ is evaluated. Then, a second curve is planned by imposing $\eta_1 = \eta_2 = \widetilde{s}_f$ and its length is used for the next iteration. After few iterations – 2 or 3 are normally sufficient – the procedure converges to a value which is very close to s_f.

Evidently, such algorithm can only be used if it convergences toward $\eta_1^* = s_f(\eta_1^*)$, where η_1^* indicates the convergence point. To this purpose, function $s_f(\eta_1)$ must assume a shape similar to the one shown in Fig. 2.4: after some iterations, the algorithm would necessarily converge toward η_1^* independently from the starting point.

Proposition 3 *For sufficiently small values of $\|\mathbf{p}_B - \mathbf{p}_A\|$, the proposed algorithm converges toward $\eta_1^* = s_f(\eta_1^*)$.*



η_1
$s_f(\widetilde{\eta}_1)$
$s_f(\eta_1^*)$
$s_f(\overline{\eta}_1) = \|\mathbf{p}_B\text{-}\mathbf{p}_A\|$
$s_f(\eta_1)$
$\overline{\eta}_1$
η_1^*
$\widetilde{\eta}_1$

Figure 2.4: Convergence of the algorithm if $s_f(\eta_1)$ is monotonically increasing. The transients are indicated through red lines.

Proof — The iterative algorithm looks for the intersection point between functions $f_1 = s_f(\eta_1)$ and $f_2 = \eta_1$. As a consequence, it is first necessary to demonstrate the two functions actually intersect each other. Owing to the continuity of function $s_f(\eta_1)$, this hypothesis is verified if there exist $\overline{\eta}_1 < \widetilde{\eta}_1$ such that $s_f(\overline{\eta}_1) > \eta_1 > s_f(\widetilde{\eta}_1)$. Furthermore, in order to prove the convergence, it is necessary to verify that the slope of $s_f(\eta_1)$ is higher than -1 over the search interval and, in particular, in η_1^*. The reason of this second requirement can be understood with the aid of Figs. 2.4 and 2.5. The first one shows two typical convergence sequences occurring when $s_f(\eta_1)$ is a monotonically increasing function. Conversely, Fig. 2.5 shows two possible situations which may occur if $(ds_f)/(d\eta_1)$ is negative in η_1^*: in case (a) the derivative is greater than -1 and the convergence is achieved, while in case (b) the algorithm does not converge.

- *There exists* $\overline{\eta}_1$ *such that* $s_f(\overline{\eta}_1) > \overline{\eta}_1$ – Closed form expressions for $\mathbf{p}'(u)$ can be found by evaluating its coefficients through (2.19)-(2.26) and by assuming that (2.45) and (2.46) apply. A few algebraic manipulations make it possible to write

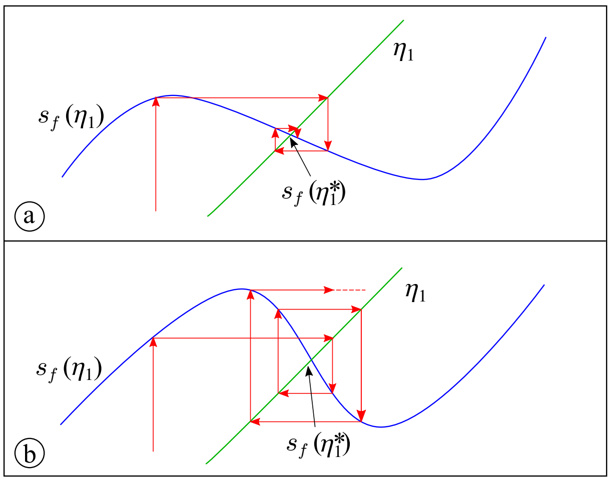


Figure 2.5: Convergence properties: (a) the algorithm converges since $(ds_f)/(d\eta_1) > -1$ for $\eta_1 = \eta_1^*$, (b) the algorithm diverges since $(ds_f)/(d\eta_1) < -1$. The transients are indicated through red lines.

$\mathbf{p}'(u)$ as follows

$$\begin{aligned}\mathbf{p}'(u;\eta_1) =& f_1(u)\eta_1^3\mathbf{b}_A + f_2(u)\eta_1^3\mathbf{b}_B + f_3(u)\eta_1^2\mathbf{n}_A + f_4(u)\eta_1^2\mathbf{n}_B \\ &+ f_5(u)\eta_1\mathbf{t}_A + f_6(u)\eta_1\mathbf{t}_B + f_7(u)(\mathbf{p}_B - \mathbf{p}_A),\end{aligned} \tag{2.51}$$

where $f_i(u), i = 1,2,\ldots,7$ are proper scalar polynomial functions which also depend on $\kappa_A, \kappa_B, \overline{\kappa}_A, \overline{\kappa}_B, \tau_A$, and τ_B.

By assuming $\eta_1 \to 0$, from (2.51) it immediately descends that

$$\lim_{\eta_1 \to 0} \left\|\mathbf{p}'(u;\eta_1)\right\| = |f_7(u)| \, \|\mathbf{p}_B - \mathbf{p}_A\| \, . \tag{2.52}$$

$s_f(\overline{\eta}_1)$ can be obtained by applying (2.52) to (2.8). The following result is obtained

$$\lim_{\eta_1 \to 0} s_f(\eta_1) = \|\mathbf{p}_B - \mathbf{p}_A\| \int_0^1 |f_7(u)| \, du = \|\mathbf{p}_B - \mathbf{p}_A\| \geq 0,$$

where $|f_7(u)| = 140(1-u)^3u^3$ is a function whose integral, evaluated over $[0,1]$, is equal to 1. Practically, by assuming $\overline{\eta}_1 = 0$ condition $s_f(\overline{\eta}_1) > \overline{\eta}_1$ is banally satisfied.

- *There exists* $\widetilde{\eta}_1$ *such that* $s_f(\widetilde{\eta}_1) < \widetilde{\eta}_1$ – By virtue of the triangular inequality, (2.51) allows one writing the following expression

$$\begin{aligned}\left\|\mathbf{p}'(u)\right\| \leq & |f_1(u)|\,\eta_1^3\,\|\mathbf{b}_A\| + |f_2(u)|\,\eta_1^3\,\|\mathbf{b}_B\| + |f_3(u)|\,\eta_1^2\,\|\mathbf{n}_A\| + |f_4(u)|\,\eta_1^2\,\|\mathbf{n}_B\| \\ & + |f_5(u)|\,\eta_1\,\|\mathbf{t}_A\| + |f_6(u)|\,\eta_1\,\|\mathbf{t}_B\| + |f_7(u)|\,\|\mathbf{p}_B - \mathbf{p}_A\|\,, \\ = & |f_1(u)|\,\eta_1^3 + |f_2(u)|\,\eta_1^3 + |f_3(u)|\,\eta_1^2 + |f_4(u)|\,\eta_1^2 \\ & + |f_5(u)|\,\eta_1 + |f_6(u)|\,\eta_1 + |f_7(u)|\,\|\mathbf{p}_B - \mathbf{p}_A\|\,, \end{aligned} \tag{2.53}$$

According to (2.8), s_f can be obtained by integrating $\|\mathbf{p}'(u)\|$. Consequently, the integrals of both sides of (2.53) lead, after some algebraic manipulations, to the following inequality

$$s_f(\eta_1) \leq K_1\eta_1^3 + K_2\eta_1^2 + 0.9074\eta_1 + \|(\mathbf{p}_B - \mathbf{p}_A)\|\,, \tag{2.54}$$

where

$$\begin{aligned} K_1 :=& 0.2798\cdot 10^{-2}(|\kappa_A\tau_A| + |\kappa_B\tau_B|) \geq 0, \\ K_2 :=& 0.8988\cdot 10^{-2}(|\kappa_A| + |\kappa_B|) + 0.4663\cdot 10^{-3}(|\bar{\kappa}_A| + |\bar{\kappa}_B|) \geq 0. \end{aligned}$$

Bearing in mind (2.54), condition $s_f(\widetilde{\eta}_1) < \widetilde{\eta}_1$ is satisfied if, in turn, the following inequality holds

$$(K_1\widetilde{\eta}_1^2 + K_2\widetilde{\eta}_1 + 0.9074)\widetilde{\eta}_1 \leq \widetilde{\eta}_1 - \|(\mathbf{p}_B - \mathbf{p}_A)\| \tag{2.55}$$

or, equivalently, if

$$(0{,}0926 - K_1\widetilde{\eta}_1^2 - K_2\widetilde{\eta}_1)\widetilde{\eta}_1 \geq \|(\mathbf{p}_B - \mathbf{p}_A)\| \tag{2.56}$$

By recalling that $K_1, K_2, \widetilde{\eta}_1 \geq 0$, two conclusion can be drawn. Firstly, the following inequality must apply in order to satisfy (2.55)

$$\widetilde{\eta}_1 \geq \|(\mathbf{p}_B - \mathbf{p}_A)\|\,. \tag{2.57}$$

Secondly, depending on the interpolating conditions (2.56) may not admit feasible solutions. In that case, the original planning problem can be split into sub-problems, so as to reduce $\|\mathbf{p}_B - \mathbf{p}_A\|$. Owing to the structure of (2.56), it will always be possible to find reasonably small values for $||\mathbf{p}_B - \mathbf{p}_A||$ and $\widetilde{\eta}_1$ such that (2.56) and (2.57) are simultaneously satisfied.

It is important to remark that conditions (2.56) is actually very restrictive, being obtained from a triangular inequality. Normally, condition $s_f(\widetilde{\eta}_1) < \widetilde{\eta}_1$ is satisfied by simply selecting a sufficiently large value of $\widetilde{\eta}_1$ fulfilling (2.57).

- *Condition* $(ds_f)/(d\eta_1) > -1$ *is satisfied* $\forall \eta_1 \in \mathbb{R}^+$ – Equation (2.51) can also be written as follows

$$\mathbf{p}'(u;\eta_1) = \underbrace{[\mathbf{t}_A\ \mathbf{n}_A\ \mathbf{b}_A]}_{{}^0_A\mathbf{R}} \begin{bmatrix} f_5(u)\eta_1 \\ f_3(u)\eta_1^2 \\ f_1(u)\eta_1^3 \end{bmatrix} \underbrace{[\mathbf{t}_B\ \mathbf{n}_B\ \mathbf{b}_B]}_{{}^0_B\mathbf{R}} \begin{bmatrix} f_6(u)\eta_1 \\ f_4(u)\eta_1^2 \\ f_2(u)\eta_1^3 \end{bmatrix} + f_7(u)(\mathbf{p}_B - \mathbf{p}_A), \tag{2.58}$$

where ${}^0_A\mathbf{R}$ and ${}^0_B\mathbf{R}$ are rotation matrices which describe the orientation of the Frenet frame at the beginning and at the end of the curve. Bearing in mind (2.8) and (2.58), the derivative of s_f w.r.t. η_1 can be written as follows

$$\frac{\partial s_f}{\partial \eta_1} = \int_0^1 \frac{\partial \mathbf{p}'(u;\eta_1)}{\partial \eta_1} \mathbf{t}(u;\eta_1) du \tag{2.59}$$

where

$$\frac{\partial \mathbf{p}'(u;\eta_1)}{\partial \eta_1} = {}^0_A\mathbf{R} \begin{bmatrix} f_5(u) \\ 2f_3(u)\eta_1 \\ 3f_1(u)\eta_1^2 \end{bmatrix} + {}^0_B\mathbf{R} \begin{bmatrix} f_6(u) \\ 2f_4(u)\eta_1 \\ 3f_2(u)\eta_1^2 \end{bmatrix}.$$

Equation (2.59) does not admit a closed form representation, so that its minimum value can be found by solving the following optimization problem

$$\min_{\gamma \in \Gamma} \left\{ \frac{\partial s_f(\eta_1)}{\partial \eta_1} \right\}$$

where $\gamma = \{{}^0_A\mathbf{R}, \kappa_A, \overline{\kappa}_A, \tau_A, {}^0_B\mathbf{R}, \kappa_B, \overline{\kappa}_B, \tau_B, \eta_1\}$ and Γ is a proper search space. In particular, ${}^0_A\mathbf{R}, {}^0_B\mathbf{R} \in SO(3)$, $\kappa_A, \kappa_B, \eta_1 \in \mathbb{R}^+$, and $\overline{\kappa}_A, \overline{\kappa}_B, \tau_A, \tau_B \in \mathbb{R}$. If the optimization problem returns a value greater than -1, then the algorithm converges independently from the interpolating conditions. The result does not depend on the direction and on the norm of $\mathbf{p}_B - \mathbf{p}_A$, so that such vector was assumed constant and equal to $\mathbf{p}_B - \mathbf{p}_A = [1\ 0\ 0]^T$.

The optimization and the subsequent analysis of the results evidenced some interesting properties. The problem is nonlinear and multimodal, so that it has been repeatedly solved by starting from randomly chosen points. The minimum cost index obtained over all the runs was equal to $-1.666 \cdot 10^{-2}$, i.e., it was much higher than -1. Similar cost indexes were found in other runs of the algorithm for alternative minimizers. A deeper analysis of the solutions revealed that the derivative of $s_f(\eta_1)$ can be negative for very particular configurations of the interpolating conditions and for values of η_1 close to $||\mathbf{p}_B - \mathbf{p}_A||$ (for the specific case, for η_1 close to 1). Furthermore, the derivative is generally positive over $\mathbb{R}^+$: when negative solutions are detected, they span over very narrow intervals of η_1. ■

Remark 1 *The most common application for η^{3D}-splines concerns the creation of smooth junctions between linear segments. For example, this is typical planning case occurring for CNC machines. In such a framework, terms K_1 and K_2 are identically zero, so that (2.56) is banally satisfied if the following condition holds*

$$\widetilde{\eta}_1 \geq \frac{||\mathbf{p}_B - \mathbf{p}_A||}{0{,}0926},$$

so that the iterative algorithm certainly converges.

As previously asserted, the iterative procedure proposed for the selection of $\boldsymbol{\eta}$ typically converges in 2-3 iterations, so that computational times are compatible with the real-time requirement. In order to prove such assertion, the iterative procedure has been tested by considering the generation of junction curves between circular arcs. A set of 2250 test cases has been generated. The following constant interpolating conditions for the starting point of the $\boldsymbol{\eta}^{3D}$-splines have been assumed:

$$\mathbf{p}_A = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \ \mathbf{R}_A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \ \kappa_A = 1, \ \dot{\kappa}_A = \tau_A = 0, \tag{2.60}$$

which are relative to a circular arc whose radius is equal to 1 m. Conversely, variable interpolating conditions have be assumed for the end-point. They have been generated as follows

$$\mathbf{p}_B = \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix}; \begin{cases} x_B & \in \{-0.3,\ 0,\ 0.3\} \\ y_B & \in \{0.3,\ 0.6,\ 0.9\} \\ z_B & \in \{0,\ 0.3\} \end{cases} \tag{2.61}$$

$$\mathbf{R}_B = R_x(\theta_2)R_z(\theta_1)\mathbf{R}_A;\ \theta_1, \theta_2 \in \left\{0,\ \frac{\pi}{4},\ \frac{\pi}{2},\ \frac{3\pi}{4},\ \pi\right\} \tag{2.62}$$

$$\kappa_B \in \{0.1,\ 0.5,\ 1,\ 2,\ 10\}, \tag{2.63}$$

$$\bar{\kappa}_B = \tau_B = 0, \tag{2.64}$$

where $R_k(\theta) \in SO(3)$ indicates a rotation around the k axis [in (2.62) $k \in \{x, z\}$]. As usual $\eta_3 = \eta_4 = \eta_5 = \eta_6 = 0$.
The algorithm converged in all the test cases. Table 2.4 shows some statistics concerning the obtained results. They are expressed as function on i, where i is the number of iterations considered. The statistics are relative the following benchmarks:

- Percentage difference e between η and s_f at the ith iteration $\left(e = \frac{|\eta_1 - s_f|}{s_f}\right)$;
- Computational time t at the ith iteration expressed in seconds;
- Percentage difference m between the initial and the final value of η_1 at the ith iteration $\left(m = \frac{\eta_1 - ||\mathbf{p}_A - \mathbf{p}_B||}{\eta_1}\right)$.

Table 2.4 makes it possible to draw some conclusions. The algorithm can be reasonably stopped at the 3rd iteration, since the average values of e is close to 1% and the m terms are very similar to the ones achieved at the 4th iteration, i.e., the last cycle only marginally modifies the path shape. The same table further shows that, in many practical cases, two iterations are actually sufficient. The planning algorithm – executed on a single core of an Intel i7-1165G7 processor running at @2.80GHz – converges, on average, in $[8\ 12] \cdot 10^{-6}$ s, i.e., the execution time is compatible with a wide range of real-time applications. Computational times associated to any i show a very low standard deviation of the associated, i.e., they are predictable: this is another important characteristic in real-time contexts.

Table 2.4: Statistics concerning the computation of η_1 after i iterations. e and m are expressed in %, t is expressed in seconds.

i		avg	dev	min	max
	e	0.095	0.058	$4.90 \cdot 10^{-5}$	0.288
1	t	$4.07 \cdot 10^{-6}$	$2.68 \cdot 10^{-7}$	$4.00 \cdot 10^{-6}$	$5.00 \cdot 10^{-6}$
	m	0.473	0.171	0.099	0.940
	e	0.032	0.028	$3.60 \cdot 10^{-8}$	0.152
2	t	$8.54 \cdot 10^{-6}$	$5.08 \cdot 10^{-7}$	$8.00 \cdot 10^{-6}$	$1.00 \cdot 10^{-5}$
	m	0.507	0.188	0.099	0.940
	e	0.012	0.014	$2.60 \cdot 10^{-11}$	0.084
3	t	$1.30 \cdot 10^{-5}$	$2.54 \cdot 10^{-7}$	$1.20 \cdot 10^{-5}$	$1.50 \cdot 10^{-5}$
	m	0.516	0.193	0.099	0.940
	e	0.005	0.007	$1.80 \cdot 10^{-14}$	0.047
4	t	$1.72 \cdot 10^{-5}$	$4.58 \cdot 10^{-7}$	$1.70 \cdot 10^{-5}$	$2.00 \cdot 10^{-5}$
	m	0.519	0.195	0.099	0.940
	e	0.002	0.003	0	0.027
5	t	$2.14 \cdot 10^{-5}$	$5.79 \cdot 10^{-7}$	$2.10 \cdot 10^{-5}$	$2.40 \cdot 10^{-5}$
	m	0.521	0.196	0.099	0.940

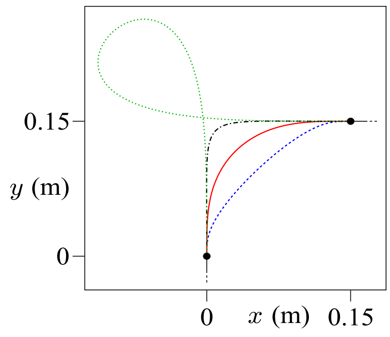


Figure 2.6: Curve shapes obtained for (blue dotted) $\eta_1 = \eta_2 = \bar{s}_f/2$; (red solid) $\eta_1 = \eta_2 = \bar{s}_f$; (black dash-dotted) $\eta_1 = \eta_2 = 1.5\bar{s}_f$; (green dashed) $\eta_1 = \eta_2 = 4\bar{s}_f$.

An application of the proposed iterative procedure is shown in Fig. 2.6, in which the η^{3D}-splines have been used for the generation of smooth junctions between two linear segments. The following interpolating conditions have been used, directly derived from the endpoints of the linear segments: $\mathbf{p}_A = [0\,0\,0]^T$, $\mathbf{p}_B = [0.15\,0.15\,0]^T$, $\mathbf{t}_A = [0\,1\,0]^T$, $\mathbf{n}_A = [1\,0\,0]^T$, $\mathbf{b}_A = [0\,0\,-1]^T$, $\mathbf{t}_B = [0\,0\,-1]^T$, $\mathbf{n}_B = [0\,-1\,0]^T$, $\mathbf{b}_B = [0\,0\,-1]^T$, $\kappa_A = \kappa_B = \overline{\kappa}_A = \overline{\kappa}_B = \tau_A = \tau_B = 0$.

The 4 curves shown in Fig. 2.6 have been obtained by assuming that (2.46) applies. Furthermore, named $\bar{s}_f$ the path length obtained through the iterative procedure, η_1 and η_2 have been chosen as follows: (a) $\eta_1 = \eta_2 = \bar{s}_f/2$; (b) $\eta_1 = \eta_2 = \bar{s}_f$; (c) $\eta_1 = \eta_2 = 1.5\bar{s}_f$; (d) $\eta_1 = \eta_2 = 4\bar{s}_f$. Fig. 2.7 shows the corresponding curvatures and highlights that solution (b) returns the smallest values: $\kappa(u)$ smoothly increases from 0 up to an almost constant value – the central part of the curve is, approximately, a circular arc – and, then, it newly decreases to 0.

Another example is proposed in Fig. 2.8. It concerns 2 straight segments which are not coplanar. Apart from $\mathbf{p}_B := [0.15\,0.15\,0.15]^T$, the remaining interpolating conditions are the same used for the previous example and, similarly, the same 4 selection rules have been assumed for η_1 and η_2. As shown in Fig. 2.9, solution (b) is still the one with the smallest curvatures.

No further examples are presented to avoid a tedious dissertation, but additional tests have shown that the proposed selection strategy generally returns curves with

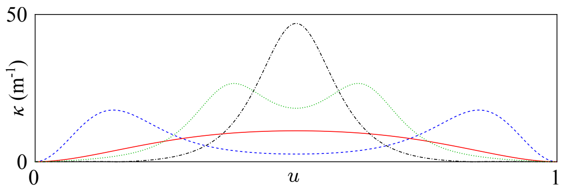


Figure 2.7: Curvature profiles corresponding to the 4 curves shown in Fig. 2.6: dashed blue line $\eta_1 = \eta_2 = \bar{s}_f/2$; solid red line $\eta_1 = \eta_2 = \bar{s}_f$; dash-dotted black line $\eta_1 = \eta_2 = 1.5\bar{s}_f$; dotted green line $\eta_1 = \eta_2 = 4\bar{s}_f$.

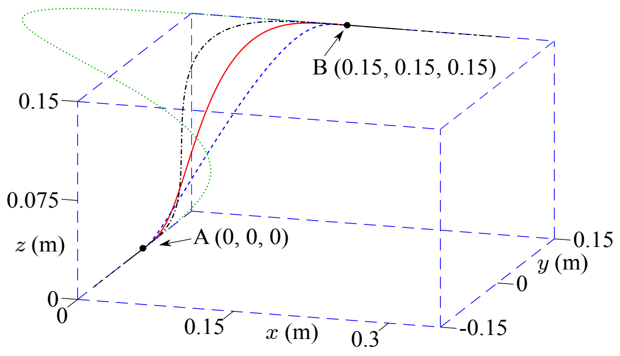


Figure 2.8: Curve shapes obtained for (a) $\eta_1 = \eta_2 = \bar{s}_f/2$; (b) $\eta_1 = \eta_2 = \bar{s}_f$; (c) $\eta_1 = \eta_2 = 1.5\bar{s}_f$; (d) $\eta_1 = \eta_2 = 4\bar{s}_f$.

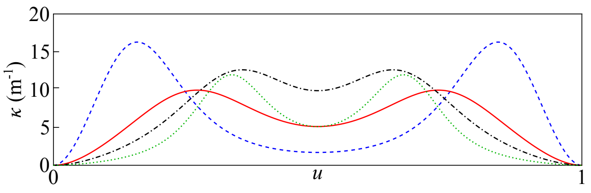


Figure 2.9: Curvature profiles corresponding to the 4 curves shown in Fig. 2.8: dashed blue line $\eta_1 = \eta_2 = \bar{s}_f/2$; solid red line $\eta_1 = \eta_2 = \bar{s}_f$; dash-dotted black line $\eta_1 = \eta_2 = 1.5\bar{s}_f$; dotted green line $\eta_1 = \eta_2 = 4\bar{s}_f$.

limited lengths and curvatures, and which avoid oscillatory behaviors. In general, for sufficiently high values of $\eta_1 = \eta_2$ – higher then the ones considered in the examples – maximum curvatures start decreasing, but curve lengths become excessive. The above mentioned characteristics allow one concluding that the strategy proposed for the selection of $\boldsymbol{\eta}$, while not yielding to optimal solutions, returns smooth $\mathscr{G}^3$ curves – curvatures and curvature derivatives are limited – of reasonable length – s_f is generally comparable with $\|\mathbf{p}_B - \mathbf{p}_A\|$. This result is achieved at a negligible computational cost, since the η^{3D}-splines coefficients are immediately obtained from (2.19)–(2.26). As already mentioned, if specific optimal conditions were to be satisfied, $\boldsymbol{\eta}$ should be selected through nonlinear programming algorithms.

In next Section 2.4 the η^{3D}-splines are experimentally tested with the aid of an industrial manipulator.

2.4 Experimental validation

η^{3D}-splines have been embedded in the path planner of an industrial manipulator and, subsequently, they have been experimentally tested by generating some $\mathscr{G}^3$ composite paths. To this purpose, a Comau Smart SiX 6-1.4 manipulator was used. The manipulator can be remotely controlled by means of an external Linux PC whose kernel was patched with the Real Time Application Interface (RTAI) software [57]. The communication between PC and robot controller exploits a real-time Ethernet connection.

As early mentioned, η^{3D}-splines can also emulate, exactly or with a good approximation, many path primitives commonly used by conventional planners. For this reason, a Cartesian planner, entirely based on the η^{3D}-splines, was implemented and exploited for the generation of a composite $\mathscr{G}^3$ path.

The trajectory shown in Fig. 2.10 has been specifically synthesized to this purpose. It is made of a set of curves whose interpolating conditions are assigned so as to generate some common path primitives. In particular, the $\mathscr{G}^3$ path contains 3 straight lines (see red segments 1, 3 and 130), 3 adjacent circular arcs (see the green segments from 5 to 7), a helical curve (see the black segments from 9 to 67), and a conic spiral

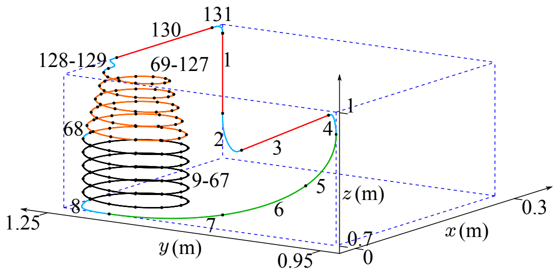


Figure 2.10: The composite $\mathscr{G}^3$ path used for Experiment 2.

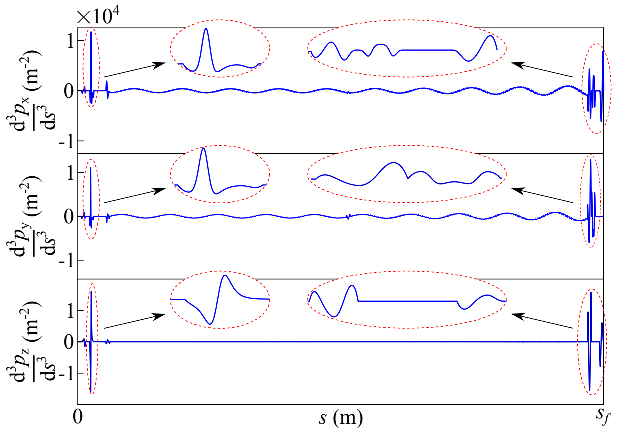


Figure 2.11: The 3 components of $(d^3\mathbf{p})/(ds^3)$ are continuous functions.

(see the orange segments from 69 to 127). Such curves are joined by means of generic η^{3D}-spline profiles so as to guarantee the overall $\mathscr{G}^3$ continuity of the composite path (see cyan segments 2, 4, 8, 68, 128, 129, 131). Black dots highlight the segments end-points. For all the curves, vector $\boldsymbol{\eta}$ was always selected according to (2.45) and (2.46).

The maximum emulation error ($2.7 \cdot 10^{-5}$ m) was detected for curves from 5 to 7: for many actual robotic applications such value is acceptable and, according to the discussion in Section 2.3, it can be further reduced by shortening the arcs lengths. Fig. 2.11 shows that, as desired, $(d^3\mathbf{p})/(ds^3)$ is continuous over the entire composite path and, consequently, the joint jerks shown in Fig. 2.12 are continuous as well.

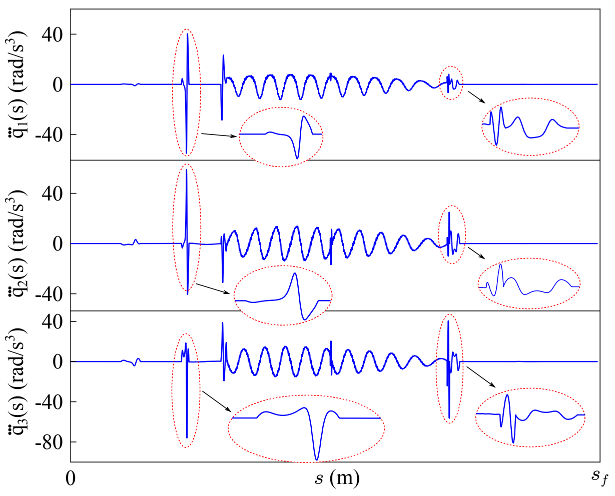


Figure 2.12: For the composite $\mathscr{G}^3$ path, the resulting jerk profiles of the first 3 joints are continuous.

Chapter 3

Corner Smoothing

The techniques mentioned in 1.2 are conceived to handle problems involving paths made of linear segments. In order to overcome such limitation, an interesting planning primitive for the flexible generation of junction segments has been recently proposed in [24]. It is named 3D general clothoids and guarantees the $\mathscr{G}^3$ continuity of the resulting composite path. According to its name, the junction curve is obtained from a set of 4 clothoids properly placed in the three-dimensional (3D) space. Similarly to other planners, it generates smooth junctions between linear segments but, additionally, it also handles cases in which circular arcs are involved. The sole drawback of the technique is represented by the planning method, which requires solving a system of equations involving the numerical computation of Fresnel integrals.

The planning primitive devised in 2 owns characteristics which make it suited for the smart generation of $\mathscr{G}^3$ smoothing curves. As before seen, the η^{3D}-splines coefficients are directly and efficiently obtained, through closed form expressions, from the interpolating conditions. Additionally, curvature, sharpness, and torsion of the η^{3D}-splines can be easily and inexpensively kept small, so that the novel planning primitive represents an ideal candidate for the online generation of smooth profiles.

This work will show how the path primitives typically assumed by the GCode of CNC machines, i.e., linear segments and circular arcs, can be efficiently and smoothly joined by means of the η^{3D}-splines, so as to obtain $\mathscr{G}^3$ composite paths. Problems related to chips thickness, contact surface and tool temperatures are supposed to be solved in advance by proper CAM programs: η^{3D}-splines are used for the final refinement of the path. The maximum tolerance between the resulting composite path and the original one is kept below an user defined threshold.

It is worth to point out that, while several efficient solutions to the corner smoothing problem have been proposed for paths made of linear segments [36, 47], the generation of $\mathscr{G}^3$ composite paths involving circular primitives is only considered in [24], where it is handled by means of the 3D general clothoids. For this reason, this work proposes direct comparisons between the results achievable with the η^{3D}-splines and the ones deriving from the use of the 3D general clothoids. In particular, it will be shown that η^{3D}-splines can accomplish the task by admitting a slightly better smoothness level through a decidedly simpler planning method: a single curve is required for the generation of each junction segment – instead of 4 – and its coefficients are found by means of closed form equations. Consequently, while the 3D general clothoids are mainly conceived for offline planning strategies, the η^{3D}-splines can also be used in online contexts.

The chapter is organized as follows. Section 3.1 explains how η^{3D}-splines can be used to join two linear segments or to create arc-to-arc junctions. Comparisons between the planning primitive proposed in [24] and the η^{3D}-splines are given in Section 3.2. Final conclusions are drawn in Section 5.3.

3.1 The Generation of $\mathscr{G}^3$ paths

A composite path for CNC machines is typically given by a sequence of linear segments and circular arcs. As a consequence, in the best case, it admits a $\mathscr{G}^1$ geometric continuity but, more frequently, it is simply $\mathscr{G}^0$. This section η^{3D}-splines will be exploited to smoothly join the original primitives so as to generate composite $\mathscr{G}^3$-paths.

Two classic cases, respectively shown in Figs. 3.1 and 3.2, are here addressed: the creation of junction curves between two linear segments and between two circular arcs. The extension to mixed cases is straightforward.

In the case of two linear segments, by indicating their reciprocal lengths as $s_{f_1} :=$

$\|\mathbf{p}_1 - \mathbf{p}_0\|$ and $s_{f_2} := \|\mathbf{p}_2 - \mathbf{p}_0\|$, respectively, the corresponding primitives can be analytically expressed as follows

$$\mathbf{p}_p(\bar{s}) := \begin{cases} \mathbf{p}_0 - \hat{\mathbf{t}}_A\,(s_{f_1} - \bar{s}) & \bar{s} \in [0, s_{f_1}] \\ \mathbf{p}_0 - \hat{\mathbf{t}}_B\,(s_{f_1} - \bar{s}) & \bar{s} \in (s_{f_1}, s_{f_1} + s_{f_2}]. \end{cases} \tag{3.1}$$

Analogously, for the two arcs in Fig. 3.2 the following equations apply

$$\mathbf{p}_p(\bar{s}) = \begin{cases} \mathbf{p}_{c_1} - r\cos\left(\frac{s_{f_1}-\bar{s}}{r_1}\right)\hat{\mathbf{n}}_{01} + r\sin\left(\frac{s_{f_1}-\bar{s}}{r_1}\right)\hat{\mathbf{t}}_{01}\,, \\ \qquad\qquad\qquad \bar{s} \in [0, s_{f_1}] \\ \mathbf{p}_{c_2} - r\cos\left(\frac{s_{f_2}-\bar{s}}{r_2}\right)\hat{\mathbf{n}}_{02} + r\sin\left(\frac{s_{f_2}-\bar{s}}{r_2}\right)\hat{\mathbf{t}}_{02}\,, \\ \qquad\qquad\qquad \bar{s} \in [s_{f_1}, s_{f_1} + s_{f_2}]. \end{cases} \tag{3.2}$$

Where $\hat{\mathbf{n}}_{ij}$ denotes the unit normal vector associated to the jth curve in $\mathbf{p}_i$. The same notation is used for tangent vectors.

The smoothing problem is banally solved by means of the η^{3D}-spline once the interpolating conditions are known. To this purpose, it is first necessary to select the start and the end points of the junction curve, namely $\mathbf{p}_A$ and $\mathbf{p}_B$, along the original segments (see also Figs. 3.1 and 3.2). Each of them is located at the same distance l from $\mathbf{p}_0$. For the same points, correct interpolating conditions must be assigned. In particular, for a junction curve between two linear segments, $\hat{\mathbf{t}}_A$ and $\hat{\mathbf{t}}_B$ can be obtained as follows

$$\hat{\mathbf{t}}_A = \frac{\mathbf{p}_0 - \mathbf{p}_1}{\|\mathbf{p}_0 - \mathbf{p}_1\|}; \quad \hat{\mathbf{t}}_B = \frac{\mathbf{p}_2 - \mathbf{p}_0}{\|\mathbf{p}_2 - \mathbf{p}_0\|}\,, \tag{3.3}$$

while $\mathbf{p}_A$ and $\mathbf{p}_B$ admit the following representation

$$\mathbf{p}_A = \mathbf{p}_0 - l\hat{\mathbf{t}}_A; \quad \mathbf{p}_B = \mathbf{p}_0 + l\hat{\mathbf{t}}_B\,. \tag{3.4}$$

Normal ($\hat{\mathbf{n}}_A$ and $\hat{\mathbf{n}}_B$) and binormal ($\mathbf{h}_A$ and $\mathbf{h}_B$) vectors do not need to be specified because the curves to be interpolated are linear segments. The $\mathscr{G}^3$ continuity is preserved by also assigning curvature, sharpness, and torsion at the beginning and at the end of the curve. Since the original paths are straight segments, they must be assigned as follows

$$\kappa_A = \kappa_B = \bar{\kappa}_A = \bar{\kappa}_B = \tau_A = \tau_B = 0\,. \tag{3.5}$$

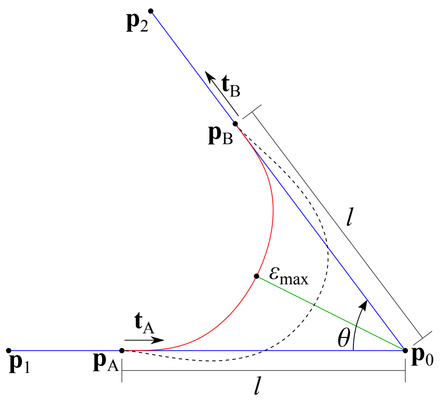


Figure 3.1: A $\mathscr{G}^3$ junction between two linear segments (solid blue lines). An η^{3D}-spline (solid red line) is used to join points A and B. ε_{max} is the maximum path approximation error. Situations like the one pointed out by the black dashed line can not occur with η^{3D}-splines.

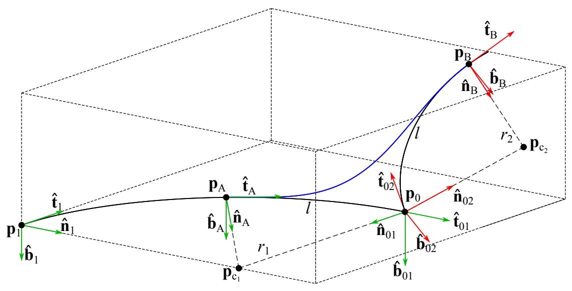


Figure 3.2: A $\mathscr{G}^3$ junction between two circular arcs (solid black lines). The green Frenet frames refer to the first arc, while the red ones refer to the second arc. An η^{3D}-spline (solid blue line) is used to join points A and B.

Similar considerations apply in case of junctions between linear segments and circular arcs or between arcs. Fig. 3.2 shows a situation concerning the generation of a junction curve between two arcs. The units vectors of the Frenet frames for the two arcs, computed in $\mathbf{p}_0$, can be expressed as follows

$$\hat{\mathbf{n}}_{01} = \frac{\mathbf{p}_{c_1} - \mathbf{p}_0}{r_1}, \quad \hat{\mathbf{n}}_1 = \frac{\mathbf{p}_{c_1} - \mathbf{p}_1}{r_1}, \tag{3.6}$$

$$\mathbf{h}_1 = \frac{\hat{\mathbf{n}}_1 \times \hat{\mathbf{n}}_{01}}{\|\hat{\mathbf{n}}_1 \times \hat{\mathbf{n}}_{01}\|}, \quad \hat{\mathbf{t}}_{01} = \hat{\mathbf{n}}_{01} \times \mathbf{h}_1, \tag{3.7}$$

$$\hat{\mathbf{n}}_{02} = \frac{\mathbf{p}_{c_2} - \mathbf{p}_0}{r_2}, \quad \hat{\mathbf{n}}_2 = \frac{\mathbf{p}_{c_2} - \mathbf{p}_2}{r_2} \tag{3.8}$$

$$\mathbf{h}_2 = \frac{\hat{\mathbf{n}}_{02} \times \hat{\mathbf{n}}_2}{\|\hat{\mathbf{n}}_{02} \times \hat{\mathbf{n}}_2\|}, \quad \hat{\mathbf{t}}_{02} = \hat{\mathbf{n}}_{02} \times \mathbf{h}_2, \tag{3.9}$$

where $r_1 := \|\mathbf{p}_{c_1} - \mathbf{p}_0\| = \|\mathbf{p}_{c_1} - \mathbf{p}_1\|$ and $r_2 := \|\mathbf{p}_{c_2} - \mathbf{p}_0\| = \|\mathbf{p}_{c_2} - \mathbf{p}_2\|$ are the radii of the two arcs. By assuming again that the η^{3D}-spline curve begins and ends at a distance l from $\mathbf{p}_0$, the following interpolating conditions can be assumed in A and B respectively

$$\mathbf{p}_A = \mathbf{p}_{c_1} - r_1 \cos(l/r_1)\, \hat{\mathbf{n}}_{01} + r_1 \sin(l/r_1)\, \hat{\mathbf{t}}_{01}, \tag{3.10}$$

$$\mathbf{p}_B = \mathbf{p}_{c_2} - r_2 \cos(l/r_2)\, \hat{\mathbf{n}}_{02} + r_2 \sin(l/r_2)\, \hat{\mathbf{t}}_{02}, \tag{3.11}$$

$$\hat{\mathbf{n}}_A = \frac{\mathbf{p}_{c_1} - \mathbf{p}_A}{r_1}, \quad \hat{\mathbf{t}}_A = \hat{\mathbf{n}}_A \times \mathbf{h}_1, \quad \mathbf{h}_A = \mathbf{h}_1, \tag{3.12}$$

$$\hat{\mathbf{n}}_B = \frac{\mathbf{p}_{c_2} - \mathbf{p}_B}{r_1}, \quad \hat{\mathbf{t}}_B = \hat{\mathbf{n}}_B \times \mathbf{h}_2, \quad \mathbf{h}_B = \mathbf{h}_2, \tag{3.13}$$

$$\kappa_A = 1/r_1, \quad \kappa_B = 1/r_2. \tag{3.14}$$

Initial and final torsion and sharpness are still equal to zero

$$\bar{\kappa}_A = \bar{\kappa}_B = \tau_A = \tau_B = 0. \tag{3.15}$$

Clearly, the resulting composite path is an approximation of the original one, whose accuracy depends on the choice of $\mathbf{p}_A$ and $\mathbf{p}_B$ – and consequently on l – and on the characteristics of the η^{3D}-spline curve. Concerning the last aspect, in case of junction curves between linear segments, shapes like the dashed one shown in Fig. 3.1 are strictly avoided by assuming (2.45)–(2.46). The resulting curve is contained in the same plane of $\hat{\mathbf{t}}_A$ and $\hat{\mathbf{t}}_B$ and admits very small curvatures and sharpnesses (see also chapter 2).

The approximation introduced by the smoothing process is typically quantified by means of the maximum Euclidean distance between the original path and the modified one. To this purpose, the following expression is used

$$e_{max} = \max_{\bar{s}\in\left[s_{f_1}-l, s_{f_1}+l\right]} \min_{u\in[0,1]} \left\{\|\mathbf{p}_p(\bar{s}) - \mathbf{p}(u)\|\right\}, \tag{3.16}$$

where $\mathbf{p}_p(\bar{s})$ indicates the original path, i.e., it is given by (3.1) or (3.2), while $\mathbf{p}(u)$ is the η^{3D}-spline junction curve. According to (3.16), the hypothesis is that $\mathbf{p}_p(\bar{s})$ is approximated by the η^{3D}-splines for a length $2l$.

All the terms required for the synthesis of the η^{3D}-spline curve can be obtained – through (3.3)–(3.5) or, alternatively, through (3.6)–(3.15) – from the knowledge of l. Consequently, e_{max} only depends on l, which can be chosen so as to guarantee that $e_{max} \leq e_{lim}$, where e_{lim} is the desired maximum approximation error. Such result is obtained by initially imposing $l = \min\left\{s_{f_1}/2, s_{f_2}/2\right\}$ and by computing e_{max} for the resulting η^{3D}-spline. If condition $e_{max} \leq e_{lim}$ is not satisfied, e_{max} can be reduced with the aid of a search method applied to l (bisection, Newton-Raphson, etc.).

In case of generic paths, e_{max} is found by solving the minimax problem (3.16). Conversely, if the smoothing problem involves two linear segments, a closed form equation can be provided, thus reducing the overall computational burden of the planning algorithm. Such equation can be found according to the procedure proposed in the remainder of this section.

The symmetry properties of the η^{3D}-splines make it possible to assert that, for the problem at hand, the maximum error is obtained in the midpoint of the curve, i.e., for $u = 1/2$, so that it can be expressed as follows

$$e_{max} = \left\|\mathbf{p}\left(\frac{1}{2}\right) - \mathbf{p}_0\right\| . \tag{3.17}$$

A few algebraic manipulations of the η^{3D}-splines equations return the following result

$$\mathbf{p}\left(\frac{1}{2}\right) - \mathbf{p}_0 = \frac{11\eta - 32l}{64}(\hat{\mathbf{t}}_A - \hat{\mathbf{t}}_B) \tag{3.18}$$

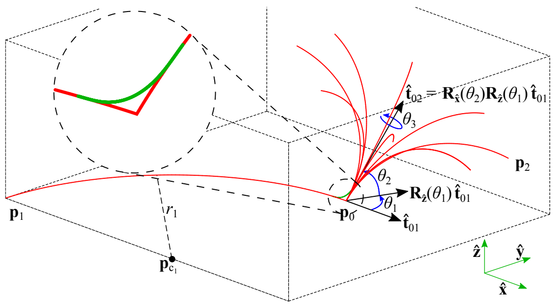

$\hat{\mathbf{t}}_{02} = \mathbf{R}_{\hat{\mathbf{x}}}(\theta_2)\mathbf{R}_{\hat{\mathbf{z}}}(\theta_1)\,\hat{\mathbf{t}}_{01}$
θ_3
$\mathbf{p}_2$
θ_2
$\mathbf{R}_{\hat{\mathbf{z}}}(\theta_1)\,\hat{\mathbf{t}}_{01}$
θ_1
$\mathbf{p}_0$
$\hat{\mathbf{t}}_{01}$
$\mathbf{p}_1$
r_1
$\mathbf{p}_{c_1}$
$\hat{\mathbf{z}}$
$\hat{\mathbf{y}}$
$\hat{\mathbf{x}}$

Figure 3.3: An interpolation test set, obtained by assuming $r_2 = 0.5$, $\theta_1 = \pi/4$, $\theta_2 = \pi/2$, and $\theta_3 \in \{-\pi, -\pi/4, -\pi/2, \ldots, 3\pi/4\}$. The green frame shows the orientation of the reference frame. The close-up circle shows an example of smoothing curve for the case $\theta_3 = 0$.

As previously asserted, η^{3D}-splines are planned by imposing $\eta = s_f$. From Fig. 3.1 it immediately descends that, necessarily, $\eta = s_f \leq 2l$ and that limit condition $\eta = 2l$ is reached only when $\hat{\mathbf{t}}_A$ and $\hat{\mathbf{t}}_B$ are perfectly aligned. As a consequence, $(32\,l - 11\,\eta)/64$ is certainly greater than 0, e_{max} can be expressed as follows

$$e_{max} = \frac{32\,l - 11\,\eta}{64} \left\| \hat{\mathbf{t}}_A - \hat{\mathbf{t}}_B \right\| . \tag{3.19}$$

By defining $\theta \in [0, \pi]$ the unsigned angle between the two segments to be joined, and by recalling the Carnot theorem, a few manipulations make it possible to write

$$e_{max} = \left(\frac{1}{2} l - \frac{11}{64} \eta \right) \sqrt{2(1 + \cos\theta)} . \tag{3.20}$$

By recalling that $\eta = s_f$ depends on the choice of l, it can be concluded that, as expected, e_{max} is only function of l.

3.2 Comparative test cases

The proposed smoothing method has been compared with the 3D general clothoids recently devised in [24]. Such planning primitive admits profiles whose smoothness level is comparable or even better than the one achievable with well known primitives like, for example, the B-splines. Furthermore, up to now it is the sole primitive which has been used for corner smoothing problems involving circular arcs.

Tests were carried out by considering the most general case, i.e., an arc-to-arc smoothing problem. All quantities in the following are expressed in centimeters. The first of the two arcs is the same for the whole test set and it is characterized by the following parameters: $r_1 = 1$, $\mathbf{p}_{c_1} = [0,0,0]^T$, $\mathbf{p}_1 = [-1,0,0]^T$, and $\mathbf{p}_0 = [0,1,0]^T$. The second arc joins the first one in $\mathbf{p}_0$ and admits $r_2 \in \{0.25, 0.5, 1, 2, 4\}$. $\hat{\mathbf{t}}_{02}$ is obtained by rotating $\hat{\mathbf{t}}_{01}$ according to the following equation

$$\hat{\mathbf{t}}_{02} = \mathbf{R}_x(\theta_2)\mathbf{R}_z(\theta_1)\hat{\mathbf{t}}_{01}\,,$$

where $\mathbf{R}_x(\theta_2)$ and $\mathbf{R}_z(\theta_1)$ indicate rotation matrices around the x and the z axes, respectively, while $\theta_1, \theta_2 \in \{0, \pi/4, \pi/2, 3\pi/4, \pi\}$. The normal and binormal unit vectors at the beginning of the second arc are obtained as follows

$$\begin{aligned}
\hat{\mathbf{n}}_{02} &= \mathbf{R}_t(\theta_3)\mathbf{R}_x(\theta_2)\mathbf{R}_z(\theta_1)\hat{\mathbf{n}}_{01}\,,\\
\mathbf{h}_{02} &= \mathbf{R}_t(\theta_3)\mathbf{R}_x(\theta_2)\mathbf{R}_z(\theta_1)\mathbf{h}_{01}\,,
\end{aligned}$$

where $\mathbf{R}_t(\theta_3)$ is a rotation matrix around the $\hat{\mathbf{t}}_{02}$ axis and $\theta_3 \in \{-\pi, -\pi/4, -\pi/2, \ldots, 3\pi/4\}$; therefore the test set contains a total of 1000 configurations (5 different values for r_2, θ_1, θ_2 and 8 values for θ_3, $5 \times 5 \times 5 \times 8 = 1000$). Fig. 3.3 shows some test curves obtained for $r_2 = 0.5$, $\theta_1 = \theta_2 = \pi/4$, and for all the values of θ_3 in the test set.

In order to propose fair comparisons, the maximum error for both primitives was kept below the same threshold $e_{lim} = 0.2$ by means of the same iterative procedure acting on l.

For each curve of the test set, the worst case curvature, sharpness, and torsion can be defined as follows

$$\begin{aligned}
\kappa_\eta^* &= \max_{u\in[0,1]}\{\kappa_\eta(u)\}, \quad & \kappa_{clot}^* &= \max_{s\in[0,s_f]}\{\kappa_{clot}(s)\}\\
\bar{\kappa}_\eta^* &= \max_{u\in[0,1]}\{|\bar{\kappa}_\eta(u)|\}, \quad & \bar{\kappa}_{clot}^* &= \max_{s\in[0,s_f]}\{|\bar{\kappa}_{clot}(s)|\},\\
\tau_\eta^* &= \max_{u\in[0,1]}\{|\tau_\eta(u)|\}, \quad & \tau_{clot}^* &= \max_{s\in[0,s_f]}\{|\tau_{clot}(s)|\},
\end{aligned}$$

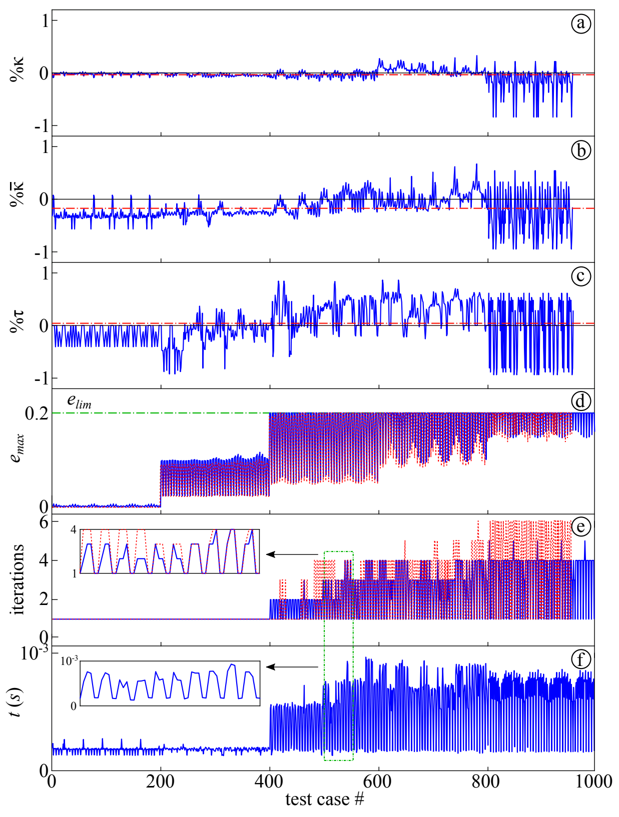


Figure 3.4: Experimental results for the 1000 test cases. Curvature, sharpness and torsion indexes are shown in (a), (b) and (c), respectively. Dash-dotted red lines indicate the average values of the three indexes. (d) compares the e_{max} values obtained with the η^{3D}-splines (solid blue line) with the ones returned by the 3D general clothoids (dashed red line). Still using the same colors, (e) shows, for each of the two methods, the number of iterations required to impose $e_{max} \leq e_{lim}$, while (f) reports the total computational time for the synthesis of each curve. The value of θ_1 changes every 200 samples: due to its influence on the η^{3D}-spline characteristics, its switches cause evident change of patterns.

where subscripts η and $clot$ indicate η^{3D}-splines and 3D general clothoids, respectively. Homologous curves of the test set can then be compared by means of the following indexes

$$\%\kappa = \frac{\kappa^*_\eta - \kappa^*_{clot}}{\max\{\kappa^*_\eta, \kappa^*_{clot}\}}, \tag{3.21}$$

$$\%\bar{\kappa} = \frac{\bar{\kappa}^*_\eta - \bar{\kappa}^*_{clot}}{\max\{\bar{\kappa}^*_\eta, \bar{\kappa}^*_{clot}\}}, \tag{3.22}$$

$$\%\tau = \frac{\tau^*_\eta - \tau^*_{clot}}{\max\{\tau^*_\eta, \tau^*_{clot}\}}, \tag{3.23}$$

Indexes denote, for each quantity, the percentage difference w.r.t. the worst case value: negative outcomes indicate that η-splines are smoother than 3D general clothoids and vice versa.

In 48 cases of the test set it was not possible to achieve convergence for the 3D general clothoids. In particular, numerical problems arose for some of the configurations admitting $\hat{\mathbf{t}}_A = -\hat{\mathbf{t}}_B$, i.e., for $\theta_1 = \pi$. Consequently, comparisons are limited to 952 test cases.

Figure 3.4 summarizes the results. More in details, Figs. 3.4a, 3.4b, and 3.4c show the trends of $\%\kappa$, $\%\bar{\kappa}$ and $\%\tau$. Both planning methods return curves with a similar smoothness level, despite slightly better profiles are generally achieved by means of the η^{3D}-splines, especially in terms of sharpness. Such conclusion is also confirmed by Table 3.1, which reports average, standard deviation, maximum, and minimum values of all indexes. Fig. 3.4d shows that the two planners provide similar performances in terms of e_{max} and that, in any case, upper bound e_{lim} is never exceeded. Necessarily, data related to the last 48 tests are missing for the 3D general clothoids.

As early stated, both algorithms fulfill condition $e_{max} \leq e_{lim}$ through an iterative procedure which acts on l. For each test case, the number of iterations required to achieve such condition is shown in Fig. 3.4e. Such figure can be evidently subdivided into clusters of 200 cases, each of them associated to a particular value of θ_1. The first cluster corresponds to $\theta_1 = 0$ and, consequently, $\hat{\mathbf{t}}_A = \hat{\mathbf{t}}_B$: the original path is, at least, $\mathscr{G}^1$ and l is found by means of a single iteration. The same happens for $\theta = \pi/4$, while higher values of θ_1 impose to iteratively select l. As pointed out

Table 3.1: Comparisons between η^{3D}-splines and 3D general clothoids in term of curvature, sharpness, and torsion

	avg	dev	min	max
$\%\kappa$	-0.0341067	0.124334	-0.840426	0.33095
$\%\bar{\kappa}$	-0.174315	0.244473	-0.951469	0.672681
$\%\tau$	0.041203	0.372961	-0.93996	0.865881

by the detail in Fig. 3.4e, the number of iterations changes even within the same cluster of tests and, in general, less iterations are required for the η^{3D}-splines. Finally, Fig. 3.4f shows the computational times required for the synthesis of the η^{3D}-splines, obtained by means of an Intel i7-1165G7 processor running at @2.80GHz. They evidently depend on the number of iterations but, even in the worst case, the solution is achieved in less than 1 millisecond. The computational times for the 3D general clothoids are not shown since they are more than two order of magnitude higher. In fact, beyond the iterations required to achieve condition $e_{max} \leq e_{lim}$, 3D general clothoids also require additional recursions for the computation of the coefficients of each curve. Indeed, they are obtained by numerically solving, through an iterative procedure, a system of equations involving some Fresnel integrals: computational times depend on the interpolating conditions and, sometimes, the algorithm may not converge. Furthermore, the iterative nature of the planning algorithm is the reason of very variable computational times, so that it can be asserted that 3D general clothoids are actually suited to off-line planning strategies.

On the contrary, computational efficiency and robustness represent the most evident characteristics of the smoothing method proposed in this work. Concerning robustness, the coefficients of the η^{3D}-splines are directly obtained from the interpolating conditions through closed form expressions (2.19)–(2.26) and, consequently, feasible solutions are always available. For the same reason, the computational burden of the novel primitive is small and predictable: a single curve is planned on average in $1.81 \cdot 10^{-4}$ s with a standard deviation equal to $3.07 \cdot 10^{-5}$ s: even considering the iterations required for the proper selection of l, the total computational time for

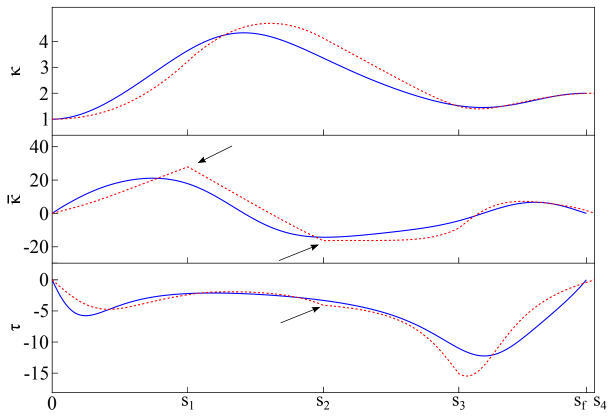


Figure 3.5: Curvature, sharpness and torsion for the η^{3D}-splines (solid blue curves) and the 3D general clothoids (dashed red curves), obtained assuming $\theta_1 = \pi/2$, $\theta_2 = 3\pi/4$, $\theta_3 = \pi/4$, and $\kappa_2 = 2$

each curve never exceeds 10^{-3} s, as proven by Fig. 3.4f.

Figure 3.5 shows curvature, sharpness, and torsion for one of the test cases and proves that both primitives are $\mathscr{G}^3$-continuous. The same figure highlights that, additionally, the inner points of the η^{3D}-splines admit an even higher continuity level, while for the 3D general clothoids the derivatives of sharpness and torsion are discontinuous in correspondence of the clothoid-to-clothoid junctions (see the black arrows).

Part II

Orientation Planner

Chapter 4

State of the Art

The target of automated manufacturing is the production of accurate objects starting from a their CAD representation. In order to pursue higher precision and flexibility, manufacturing machines evolved along the years, by gaining degrees of freedom: nowadays, it is very common to encounter CNC machines equipped with five axis. The increased number of axis imposes to devise new planning primitives, able to simultaneously manage position and orientation of the working tool. This work proposes a novel planner, particularly suited for the generation of smooth multi-axis trajectories. In this second part, the path generation problem is supposed solved by means of the η^{3D}-splines presented in the first part, so that the emphasis is mainly posed on the orientation planner.

The orientation planning problem is common to many automation contexts and can be summarized as follows: given a sequence of through points, a trajectory must be planned so that the end-effector could exactly cross all of them with the desired orientation. Typically, generated motions should simultaneously satisfy some additional features, and smoothness is certainly one of them. To this purpose, reduced solicitation can be achieved by guaranteeing that the actuators reference signals are jerk-continuous. Orientations can be expressed in multiple ways and, consequently, different planning techniques can be conceived. The most commonly used notations were compared in [5] in terms of effectiveness and conciseness. Rotation matrices have an immediate physical meaning but, being based on 9 terms, are certainly redundant. Consequently, they are not immediately suited to planning purposes. Euler angles return a compact orientation description, but they lack of effectiveness in some particular configurations. Angle-axis representation and quaternions are the most effective orientation notations: they both use four terms (one of them is clearly redundant) and are not subject to singular configurations.

One of the first planning strategies specifically conceived for quaternions was proposed by Shoemake in 1985 [58]: the Spherical Linear Interpolation (SLERP). Such strategy is well suited for the management of point-to-point trajectories, but it is not immediately usable in multi-point contexts, which, conversely, are very common in robotics applications. When the motion involves multiple via-points, the SLERP approach causes “orientation corners", which induce discontinuities in the actuators speeds, accelerations and jerks. The “orientation corner” problem is equivalent to the well known “path corner” problem arising in CNC machines: in both cases discontinuities can be avoided by admitting trajectory approximations. In order clarify the concept, let us consider the path generation problem for a CNC machine schematically shown in Fig. 4.1 (the orientation planning problem poses similar problems). Such figure shows a situation in which a generic curve, represented by the black solid line, should be followed by a milling tool. As known, in CNC applications the original path is preliminarily converted into a sequence of via-points: at run-time the machining tool executes a series of linear segments joining such points (see the blue dotted lines), thus introducing a first approximation with respect to the desired profile. In order to eliminate discontinuity issues caused by the adoption of straight line primitives, corner smoothing methods (see the green dash-dotted lines) have been proposed [59–62]. However, such strategies, as shown in Fig. 4.1, induce further approximations w.r.t the nominal path.

An equivalent problem arises for orientations trajectories when using the above mentioned SLERP primitive. In order to partially overcome the continuity issues, an alternative primitive, named SQUAD, was proposed in [63]. However, as shown in [64], such orientation primitive only guarantees the $\mathscr{C}^1$ continuity of the reference signals at the via-points.

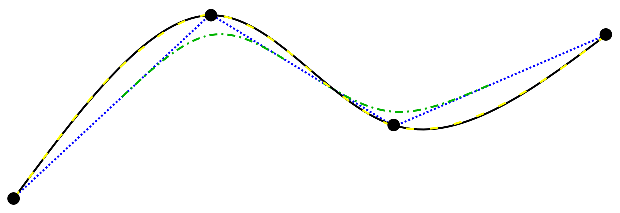


Figure 4.1: Comparison between the desired CAD path (black solid line), the GCODE converted path (blue dotted line), the corner smoothed path (green dash-dotted line) and η^{3D}-splines path (yellow dashed line).

Previous considerations make it possible to assert that, for an accurate generation of the surface profile, particular care must be dedicated to the generation of both the path and the orientation trajectories. More in details, path approximations can be strongly reduced, for example, by adopting a primitive proposed in the first part of this work: a curve which exactly crosses the assigned via-points can be generated by interpolating them through the η^{3D}-splines. As indicatively shown by the yellow dashed lines in Fig. 4.1, such primitive allows path errors which are smaller than the ones that can be achieved with the corner smoothing strategy. However, in order to implement a fully working planner in the operational space, such path primitive needs to be combined to an orientation planner. This part is devoted to the design of such planner, so as to associate a proper orientation to each point of the η^{3D}-splines path. The proposed primitive is characterized by a $\mathscr{C}^3$ continuity level.

As previously stated, continuity problems can be overcome by means of orientation corner smoothing techniques, like the ones proposed in [59–62]. Alternatively, new primitives have been designed for the direct smooth interpolation of the quaternions assigned at the via-points. To this purpose, the literature proposes several approaches, which differ each other for the adopted primitive: to obtain the $\mathscr{C}^1$ continuity Shoemake [63] and Dam et al [64] use a spherical interpolation, while to obtain the $\mathscr{C}^2$ continuity Legnani et al [65] use a repeated combination of SLERP curves, Kim et al [66], Ge et al [67] and Pu et al [68]use B-splines, Nielson [69] uses ν-splines, Liu et al [70] use quartic polynomials while Tan et al [71] use quintic polynomials. It is important to point out that all the aforementioned strategies [63–71] only guarantee the $\mathscr{C}^1$ or the $\mathscr{C}^2$ continuity of the orientation profiles. Consequently, the corresponding actuators jerks are discontinuous. The sole alternative strategies proposed in the literature, which generates position and orientation trajectories in the operational space and also guarantees jerk-continuous signals, are the ones proposed in [61, 62].

Chapter 5

Orientation

In this second part, in order to avoid jerk discontinuities and to obtain very smooth trajectories, a novel orientation planning approach, able to guarantee the $\mathscr{C}^3$ continuity of the reference signals, is designed. The new primitive is specifically conceived to be used together with the η^{3D}-splines path planner. When used together, the two primitives allow the generation of jerk-continuous trajectories. A common denominator of both primitives is represented by their low computational burden. The orientation planner proposed in this work could also be combined with alternative $\mathscr{G}^3$ path primitives, like the one given in [24] or, if jerk continuity is not mandatory, with $\mathscr{G}^2$ primitives [25, 43].

The novel planner differs from the ones proposed in [61,62] for a relevant aspect. While all the three approaches guarantee the $\mathscr{C}^3$ continuity – actually [61] allows the $\mathscr{C}^4$ continuity – the planners designed in [61, 62] are explicitly conceived to handle corner smoothing problems, so that trajectories are made of linear segments properly joined at the via-points by means of blending curves. As early seen, this implies that the assigned positions and orientations at the via-points are satisfied with a given tolerance. Conversely, the trajectory planner proposed in this work allows a strict fulfillment of the assigned positions and orientations. This is possible because the η^{3D}-splines can generate a wide variety of curves – like, for example, linear segments, circular arcs, clothoids, conic spirals, etc. – directly passing through the assigned via-points (see part 2 for more details), so that corner smoothing techniques, which necessarily cause tolerances, are not required at all.

The chapter is organized as follows. Section 5.1 introduces some preliminary considerations on the continuity characteristics which must be owned by the orientation planner in order to achieve continuous jerk signals. In the same section, some basic concepts on the quaternion theory are recalled. Section 5.2 proposes the orientation planner. Two alternative primitives are considered. Experimental tests have been executed with a Comau Smart SiX 6.14 manipulator. The corresponding outcomes are discussed in Section 5.3 and are visually shown by means of two multimedia attachments. Final conclusions are drawn in Section 5.3.

5.1 Preliminary considerations on the generation of jerk continuous signals

The main purpose of this work is to devise a new orientation planning primitive for trajectories in the operational space, so as to guarantee very smooth movements of the actuators. More precisely, the generated Cartesian trajectories must guarantee that the third order derivatives of the corresponding joint trajectories, i.e., $\dddot{\mathbf{q}}$, are continuous.

Given a manipulator with n degrees of freedom, a direct relationship exists between a point in the configuration space, i.e., $\mathbf{q} \in \mathbb{R}^n$, and the corresponding point in the operational space. In this work, points in the operational space are represented through a position $\mathbf{p} \in \mathbb{R}^3$ and an orientation $\boldsymbol{\varepsilon} \in \mathbb{R}^4$, the latter expressed through the unit quaternion representation. As a consequence, the continuity on $\mathbf{q}$ implies, in turn, the continuity on both $\mathbf{p}$ and $\boldsymbol{\varepsilon}$.

Analogously, velocities in the two spaces are correlated through the Jacobian matrix $\mathbf{J}(\mathbf{q})$, according to the following expression

$$\mathbf{v}_N = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}, \tag{5.1}$$

where $\mathbf{v}_N := [\dot{\mathbf{p}}^T \boldsymbol{\omega}^T]^T \in \mathbb{R}^6$ is the generalized velocity vector, which contains the linear and the angular velocities of the end-effector. The applications considered in this work involve non-redundant systems, so that their Jacobian matrices, far from singularities, are invertible. Consequently, (5.1) can be rewritten as follows

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\,\mathbf{v}_N. \tag{5.2}$$

Equation (5.2) makes it possible to assert that the continuity of $\dot{\mathbf{q}}$ is achieved by guaranteeing the continuity on $\mathbf{v}_N$.

Similar considerations hold for the higher order derivatives. $\mathbf{J}(\mathbf{q})$ is continuously differentiable, so that some algebraic manipulations on the derivatives of (5.1) w.r.t. the time, lead to the following expressions

$$\begin{aligned}
\ddot{\mathbf{q}} &= \mathbf{J}^{-1}(\mathbf{q})\left(\mathbf{a}_N - \dot{\mathbf{J}}\,\dot{\mathbf{q}}\right)\\
\dddot{\mathbf{q}} &= \mathbf{J}^{-1}(\mathbf{q})\left(\mathbf{j}_N - \ddot{\mathbf{J}}(\mathbf{q})\,\dot{\mathbf{q}} - 2\dot{\mathbf{J}}(\mathbf{q})\,\ddot{\mathbf{q}}\right)
\end{aligned}$$

where $\mathbf{a}_N := [\ddot{\mathbf{p}}^T\ \boldsymbol{\alpha}^T]^T$ and $\mathbf{j}_N := [\,\dddot{\mathbf{p}}^T\ \boldsymbol{\iota}^T]^T$ are the first and second time derivatives of $\mathbf{v}_N$ which, necessarily, need to be continuous in order to guarantee the continuity of $\ddot{\mathbf{q}}$ and of $\dddot{\mathbf{q}}$.

The aforementioned considerations indicate that $\mathscr{C}^3$ trajectories in the joint space can be achieved by means of trajectories in the Cartesian space which fulfill some specific path and orientation properties. A complete analysis on the path characteristics can be found in chapter 2 and it is here omitted for conciseness. Conversely, in this paper the attention is focused on the orientation planning problem: specific conditions for the generation of smooth profiles are provided in the following.

The end-effector orientation can be expressed through the Euler parameters $\boldsymbol{\varepsilon} := [\varepsilon_0\,\varepsilon_1\,\varepsilon_2\,\varepsilon_3]^T$. As known, the following condition applies

$$||\boldsymbol{\varepsilon}|| = 1.$$

A direct relationship between $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\varepsilon}}$ can be found according to the following reasoning. The end-effector orientation can be represented through a rotation matrix $\mathbf{R}$ or, alternatively, through $\boldsymbol{\varepsilon}$. The two representations are each other correlated by the following expression

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$
$$= \left[\begin{array}{c|c|c} 1-2\varepsilon_2^2-2\varepsilon_3^2 & 2(\varepsilon_1\varepsilon_2-\varepsilon_3\varepsilon_0) & 2(\varepsilon_1\varepsilon_3+\varepsilon_2\varepsilon_0) \\ 2(\varepsilon_1\varepsilon_2+\varepsilon_3\varepsilon_0) & 1-2\varepsilon_1^2-2\varepsilon_3^2 & 2(\varepsilon_2\varepsilon_3-\varepsilon_1\varepsilon_0) \\ 2(\varepsilon_1\varepsilon_3-\varepsilon_2\varepsilon_0) & 2(\varepsilon_2\varepsilon_3+\varepsilon_1\varepsilon_0) & 1-2\varepsilon_1^2-2\varepsilon_2^2 \end{array}\right]. \quad (5.3)$$

As known, the following relation applies

$$\mathbf{S}(\boldsymbol{\omega}) = \dot{\mathbf{R}}\mathbf{R}^T = \begin{bmatrix} \dot{r}_{11} & \dot{r}_{12} & \dot{r}_{13} \\ \dot{r}_{21} & \dot{r}_{22} & \dot{r}_{23} \\ \dot{r}_{31} & \dot{r}_{32} & \dot{r}_{33} \end{bmatrix} \begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix} \quad (5.4)$$

where

$$\mathbf{S}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

is a skew-symmetric matrix which depends on the angular velocity of the end-effector $\boldsymbol{\omega} := [\omega_x\, \omega_y\, \omega_z]^T$.

The components of $\boldsymbol{\omega}$ can be extracted from (5.4). In particular, a few algebraic manipulations make it possible to write

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} r_{21}\dot{r}_{31} + r_{22}\dot{r}_{32} + r_{23}\dot{r}_{33} \\ r_{31}\dot{r}_{11} + r_{32}\dot{r}_{12} + r_{33}\dot{r}_{13} \\ r_{11}\dot{r}_{21} + r_{12}\dot{r}_{22} + r_{13}\dot{r}_{23} \end{bmatrix}. \quad (5.5)$$

By substituting the terms of (5.3) into (5.5), after a few algebraic manipulations the following relationship between $\dot{\boldsymbol{\varepsilon}}$ and $\boldsymbol{\omega}$ is finally obtained

$$\boldsymbol{\omega} = 2\mathbf{M}(\boldsymbol{\varepsilon})\dot{\boldsymbol{\varepsilon}}, \quad (5.6)$$

where

$$\mathbf{M}(\boldsymbol{\varepsilon}) := \begin{bmatrix} -\varepsilon_1 & \varepsilon_0 & -\varepsilon_3 & \varepsilon_2 \\ -\varepsilon_2 & \varepsilon_3 & \varepsilon_0 & -\varepsilon_1 \\ -\varepsilon_3 & -\varepsilon_2 & \varepsilon_1 & \varepsilon_0 \end{bmatrix}. \quad (5.7)$$

The inverse relationship between $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\varepsilon}}$ can be obtained according to the following reasoning. Let us add an additional row to (5.7), so as to obtain the following matrix

$$\mathbf{M}^*(\boldsymbol{\varepsilon}) := \begin{bmatrix} \varepsilon_0 & \varepsilon_1 & \varepsilon_2 & \varepsilon_3 \\ -\varepsilon_1 & \varepsilon_0 & -\varepsilon_3 & \varepsilon_2 \\ -\varepsilon_2 & \varepsilon_3 & \varepsilon_0 & -\varepsilon_1 \\ -\varepsilon_3 & -\varepsilon_2 & \varepsilon_1 & \varepsilon_0 \end{bmatrix}.$$

It can be easily proved that $\boldsymbol{\varepsilon}$ and $\dot{\boldsymbol{\varepsilon}}$ are each other orthogonal, so that $\boldsymbol{\varepsilon}^T\dot{\boldsymbol{\varepsilon}} = 0$. Consequently, (5.6) can be rewritten as follows

$$\begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} = 2\mathbf{M}^*(\boldsymbol{\varepsilon})\,\dot{\boldsymbol{\varepsilon}}. \tag{5.8}$$

Matrix $\mathbf{M}^*(\boldsymbol{\varepsilon})$ is non-singular since it can be verified that

$$det(\mathbf{M}^*(\boldsymbol{\varepsilon})) = \left(\varepsilon_0^2 + \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2\right)^2 = ||\boldsymbol{\varepsilon}||^4 = 1,$$

so that $\mathbf{M}^*(\boldsymbol{\varepsilon})^{-1}$ always exists. Additionally, $\mathbf{M}^*(\boldsymbol{\varepsilon})$ is orthonormal, i.e. $\mathbf{M}^*(\boldsymbol{\varepsilon})^{-1} = \mathbf{M}^*(\boldsymbol{\varepsilon})^T$. Indeed, the following condition applies

$$\mathbf{M}^*(\boldsymbol{\varepsilon})^T\mathbf{M}^*(\boldsymbol{\varepsilon}) = \begin{bmatrix} ||\boldsymbol{\varepsilon}||^2 & 0 & 0 & 0 \\ 0 & ||\boldsymbol{\varepsilon}||^2 & 0 & 0 \\ 0 & 0 & ||\boldsymbol{\varepsilon}||^2 & 0 \\ 0 & 0 & 0 & ||\boldsymbol{\varepsilon}||^2 \end{bmatrix} = I^{4\times 4}.$$

As a consequence, (5.8) can be inverted as follows

$$\dot{\boldsymbol{\varepsilon}} = \frac{1}{2}\mathbf{M}^*(\boldsymbol{\varepsilon})^T \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix},$$

or, more compactly, as follows

$$\dot{\boldsymbol{\varepsilon}} = \frac{1}{2}\mathbf{M}(\boldsymbol{\varepsilon})^T\,\boldsymbol{\omega}, \tag{5.9}$$

where $\mathbf{M}(\boldsymbol{\varepsilon})$ is given by (5.7).

Due to (5.6) and (5.9), it can be asserted that the continuity of $\boldsymbol{\omega}$ is achieved through the continuity of $\dot{\boldsymbol{\varepsilon}}$.

By differentiating (5.6) the following equations are further obtained

$$
\begin{aligned}
\boldsymbol{\alpha} =& 2\left[\mathbf{M}(\boldsymbol{\varepsilon})\ddot{\boldsymbol{\varepsilon}} + \mathbf{M}(\dot{\boldsymbol{\varepsilon}})\dot{\boldsymbol{\varepsilon}}\right], \\
\boldsymbol{\iota} =& 2\left[\mathbf{M}(\boldsymbol{\varepsilon})\dddot{\boldsymbol{\varepsilon}} + 2\mathbf{M}(\dot{\boldsymbol{\varepsilon}})\ddot{\boldsymbol{\varepsilon}} + \mathbf{M}(\ddot{\boldsymbol{\varepsilon}})\dot{\boldsymbol{\varepsilon}}\right],
\end{aligned}
$$

where

$$
\mathbf{M}(\dot{\boldsymbol{\varepsilon}}) := \begin{bmatrix} -\dot{\varepsilon}_1 & \dot{\varepsilon}_0 & -\dot{\varepsilon}_3 & \dot{\varepsilon}_2 \\ -\dot{\varepsilon}_2 & \dot{\varepsilon}_3 & \dot{\varepsilon}_0 & -\dot{\varepsilon}_1 \\ -\dot{\varepsilon}_3 & -\dot{\varepsilon}_2 & \dot{\varepsilon}_1 & \dot{\varepsilon}_0 \end{bmatrix},
$$

$$
\mathbf{M}(\ddot{\boldsymbol{\varepsilon}}) := \begin{bmatrix} -\ddot{\varepsilon}_1 & \ddot{\varepsilon}_0 & -\ddot{\varepsilon}_3 & \ddot{\varepsilon}_2 \\ -\ddot{\varepsilon}_2 & \ddot{\varepsilon}_3 & \ddot{\varepsilon}_0 & -\ddot{\varepsilon}_1 \\ -\ddot{\varepsilon}_3 & -\ddot{\varepsilon}_2 & \ddot{\varepsilon}_1 & \ddot{\varepsilon}_0 \end{bmatrix},
$$

i.e., the continuity on $\boldsymbol{\alpha}$ and $\boldsymbol{\iota}$ is mapped into equivalent continuity conditions on $\ddot{\boldsymbol{\varepsilon}}$ and $\dddot{\boldsymbol{\varepsilon}}$.

5.2 The $\mathscr{C}^3$ continuous orientation planner

According to the discussion in Section 5.1, the $\mathscr{C}^3$ continuity of the trajectory in the configuration space imposes, in turn, that also the planning primitive adopted for $\boldsymbol{\varepsilon}$ must be $\mathscr{C}^3$. In order to guarantee a strict relationship between orientation and curvilinear coordinate, $\boldsymbol{\varepsilon}$ is planned as a function of the curvilinear coordinate s. Consequently, a trajectory for $\boldsymbol{\varepsilon}$ is obtained by combining $\boldsymbol{\varepsilon}(s)$ with a timing law $s(t)$. The time derivatives of the orientation trajectory can be expressed as follows

$$
\begin{aligned}
\dot{\boldsymbol{\varepsilon}} &= \frac{d\boldsymbol{\varepsilon}}{dt} = \frac{d\boldsymbol{\varepsilon}}{ds}\frac{ds}{dt} = \dot{s}\boldsymbol{\varepsilon}', \\
\ddot{\boldsymbol{\varepsilon}} &= \dot{s}^2\boldsymbol{\varepsilon}'' + \ddot{s}\boldsymbol{\varepsilon}', \\
\dddot{\boldsymbol{\varepsilon}} &= \dot{s}^3\boldsymbol{\varepsilon}''' + 3\dot{s}\ddot{s}\boldsymbol{\varepsilon}'' + \dddot{s}\boldsymbol{\varepsilon}',
\end{aligned}
$$

where $\boldsymbol{\varepsilon}' := (d\boldsymbol{\varepsilon})/(ds)$, $\boldsymbol{\varepsilon}'' := (d^2\boldsymbol{\varepsilon})/(ds^2)$, and $\boldsymbol{\varepsilon}''' := (d^3\boldsymbol{\varepsilon})/(ds^3)$. Clearly, the required $\mathscr{C}^3$ continuity on $\dddot{\boldsymbol{\varepsilon}}$ is achieved if $\boldsymbol{\varepsilon}(s)$ and $s(t)$ are both $\mathscr{C}^3$ continuous. For conciseness, the dependency of $\boldsymbol{\varepsilon}$ on s is dropped in the following.

The orientation primitive could be potentially planned by only considering components ε_1, ε_2, and ε_3. The fourth component, i.e. ε_0 could be subsequently obtained by imposing condition $\|\boldsymbol{\varepsilon}\| = 1$. However, such planning strategy would lead to complex expressions for $\boldsymbol{\varepsilon}'$, $\boldsymbol{\varepsilon}''$, and $\boldsymbol{\varepsilon}'''$. For such reason, the four components of the quaternion are independently planned. The resulting vector, i.e., $\bar{\boldsymbol{\varepsilon}}$, is later normalized in order to fulfill condition $\|\boldsymbol{\varepsilon}\| = 1$ by means of the following equation

$$\boldsymbol{\varepsilon} = \frac{\bar{\boldsymbol{\varepsilon}}}{|\bar{\boldsymbol{\varepsilon}}|}.$$

The derivatives of $\boldsymbol{\varepsilon}$ w.r.t. s can be easily obtained through some algebraic manipulations. More precisely, it can be easily proved that they can be represented as follows

$$\boldsymbol{\varepsilon}' = \frac{\bar{\boldsymbol{\varepsilon}}'}{|\bar{\boldsymbol{\varepsilon}}|} + \alpha\bar{\boldsymbol{\varepsilon}}, \tag{5.10}$$

$$\boldsymbol{\varepsilon}'' = \frac{\bar{\boldsymbol{\varepsilon}}''}{|\bar{\boldsymbol{\varepsilon}}|} + 2\alpha\bar{\boldsymbol{\varepsilon}}' + \beta\bar{\boldsymbol{\varepsilon}}, \tag{5.11}$$

$$\boldsymbol{\varepsilon}''' = \frac{\bar{\boldsymbol{\varepsilon}}'''}{|\bar{\boldsymbol{\varepsilon}}|} + 3\alpha\bar{\boldsymbol{\varepsilon}}'' + 3\beta\bar{\boldsymbol{\varepsilon}}' + \gamma\bar{\boldsymbol{\varepsilon}}, \tag{5.12}$$

where

$$\alpha = \frac{d}{ds}\left(\frac{1}{|\bar{\boldsymbol{\varepsilon}}|}\right) = -\frac{\bar{\boldsymbol{\varepsilon}}^T\bar{\boldsymbol{\varepsilon}}'}{|\bar{\boldsymbol{\varepsilon}}|^3}, \tag{5.13}$$

$$\beta = \frac{d^2}{ds^2}\left(\frac{1}{|\bar{\boldsymbol{\varepsilon}}|}\right) = -\frac{|\bar{\boldsymbol{\varepsilon}}'|^2}{|\bar{\boldsymbol{\varepsilon}}|^3} - \frac{\bar{\boldsymbol{\varepsilon}}^T\bar{\boldsymbol{\varepsilon}}''}{|\bar{\boldsymbol{\varepsilon}}|^3} + 3\frac{(\bar{\boldsymbol{\varepsilon}}^T\bar{\boldsymbol{\varepsilon}}')^2}{|\bar{\boldsymbol{\varepsilon}}|^5}, \tag{5.14}$$

$$\begin{aligned}\gamma = \frac{d^3}{ds^3}\left(\frac{1}{|\bar{\boldsymbol{\varepsilon}}|}\right) &= -3\frac{\bar{\boldsymbol{\varepsilon}}'^T\bar{\boldsymbol{\varepsilon}}''}{|\bar{\boldsymbol{\varepsilon}}|^3} - \frac{\bar{\boldsymbol{\varepsilon}}^T\bar{\boldsymbol{\varepsilon}}'''}{|\bar{\boldsymbol{\varepsilon}}|^3} \\ &\quad + 9\frac{(\bar{\boldsymbol{\varepsilon}}^T\bar{\boldsymbol{\varepsilon}}'' + |\bar{\boldsymbol{\varepsilon}}'|^2)(\bar{\boldsymbol{\varepsilon}}^T\bar{\boldsymbol{\varepsilon}}')}{|\bar{\boldsymbol{\varepsilon}}|^5} - 15\frac{(\bar{\boldsymbol{\varepsilon}}^T\bar{\boldsymbol{\varepsilon}}')^3}{|\bar{\boldsymbol{\varepsilon}}|^7}.\end{aligned} \tag{5.15}$$

Evidently, by virtue of (5.10)–(5.15), the continuity properties of $\bar{\boldsymbol{\varepsilon}}$ are inherited by $\boldsymbol{\varepsilon}$, so that $\boldsymbol{\varepsilon}$ is certainly $\mathscr{C}^3$ if $\bar{\boldsymbol{\varepsilon}}$ is $\mathscr{C}^3$.

According to the premises, the components of $\bar{\boldsymbol{\varepsilon}}$ are planned independently. Thus, a scalar primitive is adopted for each of them: it is indicated in the reminder of this section as $f(s)$.

Given a set of $n+1$ via-points points, whose displacement along the curve is given through the corresponding curvilinear coordinates $\mathbf{s} = \{0, s_1, \ldots, s_n\}$, function $f(s)$ can be partitioned into a set of n sub-functions defined as follows

$$f(s) := \begin{cases} f_1(s) & \text{if } 0 \leq s \leq s_1 \\ f_2(s - s_1) & \text{if } s_1 < s \leq s_2 \\ \ldots & \\ f_n(s - s_{n-1}) & \text{if } s_{n-1} < s \leq s_n \end{cases} . \tag{5.16}$$

By defining $\hat{s} := s - s_{k-1}$, functions $f_k(\cdot)$ in (5.16) can be rewritten as follows

$$f_k(\hat{s}), \quad 0 < \hat{s} \leq \hat{s}_k,$$

where $\hat{s}_k := s_k - s_{k-1}$.

Many alternative primitives can be adopted for $f_k(\hat{s}), k = 1, 2, \ldots, n$. A nice survey is proposed in [72] for the achievement of composite $\mathscr{C}^2$ function. In this work, a possible strategy is devised in order to achieve the required $\mathscr{C}^3$ continuity. For both of them $f_k(\hat{s})$ is defined as follows

$$f_k(\hat{s}) = a_{0k} + a_{1k}\hat{s} + a_{2k}\hat{s}^2 + a_{3k}\hat{s}^3 + a_{4k}\hat{s}^4 + a_{5k}\hat{s}^5.$$

The polynomials coefficients can be evaluated by solving a linear system which can be set up by considering a proper set of interpolating conditions. For example, since the orientations at the via-points are assigned, $\hat{f}_0 := f_1(0)$, $\hat{f}_k := f_k(\hat{s}_k), k = 1, 2, \ldots, n$ are known and must be satisfied by the polynomials. In the two strategies next proposed, jerks at the beginning and at the end of each segment of the composite trajectory are posed equal to zero in order to limit the signal variability.

5.2.1 Strategy A

In the first method, two additional free-displacement points are added in the middle of the first and of the last intervals of the composite function (see also [72]) in order to balance the number of equations and unknowns. This implies that $\bar{n} = n+2$ polynomial functions, namely quintic splines, must be generated.

The following conditions must be satisfied in order to generate a $\mathscr{C}^3$ composite function

- imposition of the initial and final values for each segment of the composite curve ($2\bar{n}-4$ conditions)

$$\begin{aligned} &f_1(0) = \hat{f}_0 \\ &f_k(\hat{s}_k) = f_{k+1}(0) = \hat{f}_k, \quad k = 2,3,\ldots,\bar{n}-2, \\ &f_n(\hat{s}_{\bar{n}}) = \hat{f}_{\bar{n}}. \end{aligned}$$

  It is worth noticing that the positions of the free-displacement points have not been assigned;

- imposition of the composite function continuity at the free-displacement points (2 conditions)

$$f_1(\hat{s}_1) = f_2(0), \quad f_{\bar{n}-1}(\hat{s}_{\bar{n}-1}) = f_{\bar{n}}(0);$$

- imposition of the speed and the acceleration continuity [$2(\bar{n}-1)$ conditions]

$$\begin{aligned} f'_k(\hat{s}_k) = f'_{k+1}(0), \quad k = 1,2,\ldots,\bar{n}-1, \\ f''_k(\hat{s}_k) = f''_{k+1}(0), \quad k = 1,2,\ldots,\bar{n}-1; \end{aligned}$$

- imposition of null jerks at the beginning and at the end of each segment ($2\bar{n}$ conditions)

$$f'''_k(0) = f'''_k(\hat{s}_k) = 0, \quad k = 1,2,\ldots,\bar{n};$$

- imposition of the speeds and the accelerations at the beginning and end of the composite curve (4 conditions)

$$\begin{aligned} f'_1(0) &= \hat{f}'_0, & f''_1(0) &= \hat{f}''_0, \\ f'_{\bar{n}}(\hat{s}_{\bar{n}}) &= \hat{f}'_{\bar{n}}, & f''_{\bar{n}}(\hat{s}_{\bar{n}}) &= \hat{f}''_{\bar{n}}. \end{aligned}$$

Summarizing, the planning problem requires solving the following linear system made of $6\bar{n}$ equations in $6\bar{n}$ unknowns

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{5.17}$$

where $\mathbf{x} := [a_{01}\, a_{11}\, a_{21}\, a_{31}\, a_{02}\, a_{12}\, \dots\, a_{\bar{n}\bar{n}}]^T \in \mathbb{R}^{6\bar{n}}$ is the vector of the unknowns, $\mathbf{A} \in \mathbb{R}^{6\bar{n}\times 6\bar{n}}$, and $\mathbf{b} \in \mathbb{R}^{6\bar{n}}$. Some algebraic manipulations make it possible to reduce the order of (5.17), so that $\mathbf{A}$ is converted into the following three-diagonal matrix

$$\mathbf{A} = \begin{bmatrix} 20\hat{s}_1 & 3\hat{s}_1 & & & & \\ 20\hat{s}_1 & 10\hat{s}_1 + 7\hat{s}_3 & 3\hat{s}_3 & & \mathbf{0} & \\ & \ddots & \ddots & \ddots & & \\ & 3\hat{s}_k & 7(\hat{s}_k + \hat{s}_{k+1}) & 3\hat{s}_{k+1} & & \\ & & \ddots & \ddots & \ddots & \\ & \mathbf{0} & & 3\hat{s}_{\bar{n}-2} & 7\hat{s}_{\bar{n}-2} + 10\hat{s}_{\bar{n}} & 20\hat{s}_{\bar{n}} \\ & & & & 3\hat{s}_{\bar{n}} & 20\hat{s}_{\bar{n}} \end{bmatrix}$$

with $\mathbf{A} \in \mathbb{R}^{(\bar{n}-1)\times(\bar{n}-1)}$. Analogously, the structure of $\mathbf{b} \in \mathbb{R}^{\bar{n}-1}$ changes as follows

$$\mathbf{b} = \begin{bmatrix} 10\frac{\hat{f}_2-\hat{f}_0}{\hat{s}_1} - 20\hat{f}'_0 - \frac{17}{2}\hat{f}''_0\hat{s}_1 \\ 10\frac{\hat{f}_3-\hat{f}_2}{\hat{s}_3} - 10\hat{f}'_0 - 5\hat{f}''_0\hat{s}_1 \\ \vdots \\ 10\frac{\hat{f}_{k+1}-\hat{f}_k}{\hat{s}_{k+1}} - 10\frac{\hat{f}_k-\hat{f}_{k-1}}{\hat{s}_k} \\ \vdots \\ -10\frac{\hat{f}_{\bar{n}-2}-\hat{f}_{\bar{n}-3}}{\hat{s}_{\bar{n}-2}} + 10\hat{f}'_{\bar{n}} - 5\hat{f}''_{\bar{n}}\hat{s}_{\bar{n}} \\ -10\frac{\hat{f}_{\bar{n}}-\hat{f}_{\bar{n}-2}}{\hat{s}_{\bar{n}}} + 20\hat{f}'_{\bar{n}} - \frac{17}{2}\hat{f}''_{\bar{n}}\hat{s}_{\bar{n}} \end{bmatrix} \in \mathbb{R}^{\bar{n}-1}.$$

and $\mathbf{x} = [a_{22}\, a_{23}\, \dots\, a_{2\bar{n}}]^T \in \mathbb{R}^{\bar{n}-1}$ becomes the vector of the unknowns.

Efficient algorithms exist for the inversion of three-diagonal matrices [73], so that large values of $\bar{n}$ can be easily handled. Once the reduced order system has been solved, the remaining coefficients of the polynomials can be obtained through the following closed form equations

$$a_{01} = \hat{f}_0, \tag{5.18}$$

$$a_{02} = \frac{6a_{22} + 7\hat{f}_0''}{20}\hat{s}_1^2 + \hat{f}_0'\hat{s}_1 + \hat{f}_0, \tag{5.19}$$

$$a_{0j} = \hat{f}_{j-1}, \quad j = 3, 4, \ldots, \bar{n} - 1, \tag{5.20}$$

$$a_{0\bar{n}} = \frac{6a_{2\bar{n}} + 7\hat{f}_{\bar{n}}''}{20}\hat{s}_{\bar{n}}^2 + \hat{f}_{\bar{n}}'\hat{s}_{\bar{n}} + \hat{f}_{\bar{n}}, \tag{5.21}$$

$$a_{11} = \hat{f}_0', \tag{5.22}$$

$$a_{12} = -\frac{20a_{22} + 6a_{23} + 7\hat{f}_0''}{20}\hat{s}_1 - \hat{f}_0' - \frac{\hat{f}_0 - \hat{f}_2}{\hat{s}_1}, \tag{5.23}$$

$$a_{1j} = -\frac{7a_{2j} + 3a_{2(j+1)}}{10}\hat{s}_j + \frac{\hat{f}_j - \hat{f}_{j-1}}{\hat{s}_j}, \qquad j = 3, 4, \ldots, \bar{n} - 2, \tag{5.24}$$

$$a_{1(\bar{n}-1)} = -7\frac{2a_{2(\bar{n}-1)} - \hat{f}_{\bar{n}}''}{20}\hat{s}_{\bar{n}} - \hat{f}_{\bar{n}}' + \frac{\hat{f}_{\bar{n}} - \hat{f}_{\bar{n}-2}}{\hat{s}_{\bar{n}}}, \tag{5.25}$$

$$a_{1\bar{n}} = -\frac{2a_{2\bar{n}} + \hat{f}_{\bar{n}}''}{2}\hat{s}_{\bar{n}} + \hat{f}_{\bar{n}}', \tag{5.26}$$

$$a_{21} = \frac{1}{2}\hat{f}_0'', \tag{5.27}$$

$$a_{3j} = 0, \quad j = 1, 2, \ldots, \bar{n}, \tag{5.28}$$

$$a_{41} = \frac{2a_{22} - \hat{f}_0''}{4\hat{s}_1^2}, \tag{5.29}$$

$$a_{4j} = \frac{a_{2(j+1)} - a_{2j}}{2\hat{s}_j^2}, \quad j = 2, 3, \ldots, \bar{n} - 1, \tag{5.30}$$

$$a_{4\bar{n}} = \frac{\hat{f}_{\bar{n}}'' - 2a_{2\bar{n}}}{4\hat{s}_{\bar{n}}^2} \tag{5.31}$$

$$a_{51} = \frac{\hat{f}_0'' - 2a_{22}}{10\hat{s}_1^3}, \tag{5.32}$$

$$a_{5j} = \frac{a_{2j} - a_{2j+1}}{5\hat{s}_j^3}, \quad j = 2, 3, \ldots, \bar{n} - 1, \tag{5.33}$$

$$a_{5\bar{n}} = \frac{2a_{2\bar{n}} - \hat{f}_{\bar{n}}''}{10\hat{s}_{\bar{n}}^3}. \tag{5.34}$$

The aforementioned solution applies for $\bar{n} > 4$, i.e., when the problem admits more than 3 planning points. If $\bar{n} = 4$, matrices **A** and **b** of (5.17) change as follows

$$\mathbf{b} = \begin{bmatrix} 10\frac{\hat{f}_2 - \hat{f}_0}{\hat{s}_1} - 20\hat{f}_0' - \frac{17}{2}\hat{f}_0''\hat{s}_1 \\ 2(\hat{f}_4' - \hat{f}_0') - \hat{f}_0''\hat{s}_1 - \hat{f}_4''\hat{s}_4 \\ -10\frac{\hat{f}_4 - \hat{f}_2}{\hat{s}_4} + 20\hat{f}_4' - \frac{17}{2}\hat{f}_4''\hat{s}_4 \end{bmatrix},$$

$$\mathbf{A} = \begin{bmatrix} 20\hat{s}_1 & 3\hat{s}_1 & 0 \\ 4\hat{s}_1 & 2(\hat{s}_1 + \hat{s}_4) & 4\hat{s}_4 \\ 0 & 3\hat{s}_4 & 20\hat{s}_4 \end{bmatrix},$$

while $\mathbf{x} = [a_{22}\, a_{23}\, a_{24}]^T \in \mathbb{R}^3$. The remaining coefficients are still obtained through (5.18)–(5.34) by assuming $\bar{n} = 4$.

5.2.2 Strategy B

Alternatively, the orientation functions can be planned by still assuming 5th order polynomials, but by increasing to 6 the order of the first and of the last polynomials in order to avoid adding the two free-displacement points.

The linear system matrices $\mathbf{A} \in \mathbb{R}^{(n-1)\times(n-1)}$ and $\mathbf{b} \in \mathbb{R}^{n-1}$ assume the following expressions

$$\mathbf{A} = \begin{bmatrix} 4\hat{s}_1 + 7\hat{s}_2 & 3\hat{s}_2 & & & & \mathbf{0} \\ 3\hat{s}_2 & 7(\hat{s}_2 + \hat{s}_3) & 3\hat{s}_3 & & & \\ \ddots & \ddots & \ddots & & & \\ & 3\hat{s}_k & 7(\hat{s}_k + \hat{s}_{k+1}) & 3\hat{s}_{k+1} & & \\ & & \ddots & \ddots & \ddots & \\ \mathbf{0} & & & 3\hat{s}_{n-2} & 7(\hat{s}_{n-2} + \hat{s}_{n-1}) & 3\hat{s}_{n-1} \\ & & & & 3\hat{s}_{n-1} & 7\hat{s}_{n-1} + 4\hat{s}_n \end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix} 20\frac{\hat{f}_0-\hat{f}_1}{\hat{s}_1} - 10\frac{\hat{f}_1-\hat{f}_2}{\hat{s}_2} + 10\hat{f}'_0 + 2\hat{f}''_0\hat{s}_1 \\ \vdots \\ 10\frac{\hat{f}_{k+1}-\hat{f}_k}{\hat{s}_{k+1}} - 10\frac{\hat{f}_k-\hat{f}_{k-1}}{\hat{s}_k} \\ \vdots \\ 20\frac{\hat{f}_n-\hat{f}_{n-1}}{\hat{s}_n} - 10\frac{\hat{f}_{n-1}-\hat{f}_{n-2}}{\hat{s}_{n-1}} - 10\hat{f}'_n + 2\hat{f}''_n\hat{s}_n \end{bmatrix},$$

while $\mathbf{x} = [a_{22}\, a_{23}\, \dots\, a_{2n}]^T \in \mathbb{R}^{n-1}$. The remaining coefficients can be computed through the following closed form equations

$$a_{0j} = \hat{f}_{j-1}, \quad j = 1,2,...,n, \tag{5.35}$$

$$a_{11} = \hat{f}'_0, \tag{5.36}$$

$$a_{1j} = -\frac{7a_{2j} + 3a_{2(j+1)}}{10}\hat{s}_j + \frac{\hat{f}_j - \hat{f}_{j-1}}{\hat{s}_j}, j = 2,3,...,n-1, \tag{5.37}$$

$$a_{1n} = -\frac{2a_{2n} - \hat{f}''_n}{5}\hat{s}_n - \hat{f}'_n + 2\frac{\hat{f}_n - \hat{f}_{n-1}}{\hat{s}_n}, \tag{5.38}$$

$$a_{21} = \frac{1}{2}\hat{f}''_0, \tag{5.39}$$

$$a_{3j} = 0, \quad j = 1,2,\dots,n, \tag{5.40}$$

$$a_{41} = -\frac{a_{22} + 2\hat{f}''_0}{\hat{s}_1^2} - 5\frac{\hat{f}'_0}{\hat{s}_1^3} + 5\frac{\hat{f}_1 - \hat{f}_0}{\hat{s}_1^4}, \tag{5.41}$$

$$a_{4j} = \frac{a_{2(j+1)} - a_{2j}}{2\hat{s}_j^2}, \quad j = 2,3,\dots,n-1, \tag{5.42}$$

$$a_{4n} = -\frac{4a_{2n} + 3\hat{f}''_n}{2\hat{s}_n^2} - 5\frac{\hat{f}'_n}{\hat{s}_n^3} - 5\frac{\hat{f}_n - \hat{f}_{n-1}}{\hat{s}_n^4}, \tag{5.43}$$

$$a_{51} = \frac{(8a_{22} + 11\hat{f}''_0)}{5\hat{s}_1^3} + 6\frac{\hat{f}'_0}{\hat{s}_1^4} - 6\frac{\hat{f}_1 - \hat{f}_0}{\hat{s}_1^5}, \tag{5.44}$$

$$a_{5j} = \frac{a_{2j} - a_{2j+1}}{5\hat{s}_j^3}, \quad j = 2, 3, \dots, n-1, \tag{5.45}$$

$$a_{5n} = 2\frac{a_{2n} + \hat{f}_n''}{\hat{s}_n^3} - 6\frac{\hat{f}_n'}{\hat{s}_n^4} + 6\frac{\hat{f}_n - \hat{f}_{n-1}}{\hat{s}_n^5}, \tag{5.46}$$

$$a_{61} = -\frac{6a_{22} + 7\hat{f}_0''}{10\hat{s}_1^4} - 2\frac{\hat{f}_0'}{\hat{s}_1^5} + 2\frac{\hat{f}_1 - \hat{f}_0}{\hat{s}_1^6}, \tag{5.47}$$

$$a_{6n} = -\frac{6a_{2n} + 7\hat{f}_n''}{10\hat{s}_n^4} + 2\frac{\hat{f}_n'}{\hat{s}_n^5} - 2\frac{\hat{f}_n - \hat{f}_{n-1}}{\hat{s}_n^6}. \tag{5.48}$$

A particular solution is required when only three points are given (n=2). The composite curve is made of 2 polynomials. Coefficient a_{22} is obtained by means of the following equation

$$a_{22} = \frac{1}{2(\hat{s}_1 + \hat{s}_2)} \left[\hat{f}_0'' \hat{s}_1 + \hat{f}_2'' \hat{s}_2 + 5(\hat{f}_0' - \hat{f}_2') - 10\frac{\hat{f}_1 - \hat{f}_0}{\hat{s}_1} + 10\frac{\hat{f}_2 - \hat{f}_1}{\hat{s}_2} \right],$$

while the remaining coefficients are calculated through (5.35), (5.36), (5.38)–(5.41), (5.43), (5.44) and (5.46)–(5.48) by assuming $n = 2$.

5.3 Experimental Results

In order to obtain a $\mathscr{C}^3$ timing law, a jerk-continuous $s(t)$ profile is needed. Typically, Cartesian trajectories are used for applications which require a constant speed of the tool frame (machining, gluing, soldering, laser cutting, etc.). For this reason, a simple timing law, like the one shown in Fig. 5.1, was adopted for the experiments. More complex shapes can be assumed for alternative tasks, provided that the resulting function is $\mathscr{C}^3$: for example, in order to satisfy a set of given velocity or acceleration limits, a trajectory scaling strategy like the one proposed in [74] can be used. The timing law planning problem is not considered in this work, so that no further details are provided.

The orientation planner has been tested by means of simulations and through the aid of a Comau Smart SiX 6.14 manipulator, a six link anthropomorphic robot. The end-effector path is generated by means of the η^{3D}-spline planner proposed in chapter 2, while the orientation is provided through the first primitive proposed here

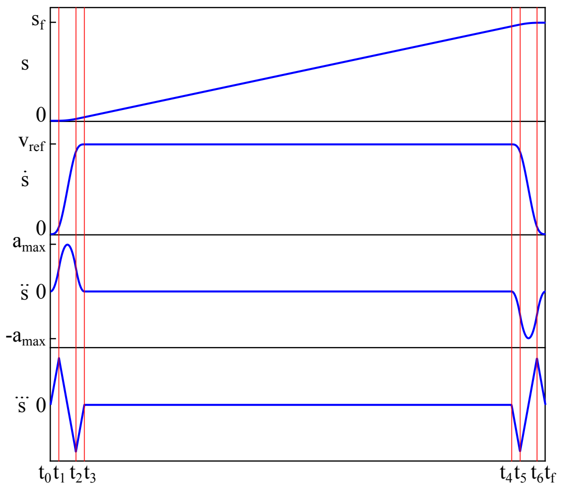


Figure 5.1: The timing law used for the experiments.

(Strategy A). A sphere milling test has been simulated in order to show the effectiveness of the planner for machining purposes. Additionally, two test trajectories have been generated and, then, executed with a real manipulator. In the first experiment, a milling trajectory is planned for a spherical surface. Fig. 5.2 shows its shape. When finishing an object, the perpendicularity – or keeping a given angle – between the milling cutter and the surface is critically important in order to guarantee both efficiency (in terms of chip rate, bit temperature, etc.) and surface quality. The proposed example assumes that the tool tip axis is always aligned with the radius of the sphere, so that the planner effectiveness is checked by considering the error angle between the axis of the milling tool and the surface normal. It was verified that the larger errors occur when the path direction suddenly changes, so that the milling precision can be simply improved by increasing the density of via-points at the turning points. A maximum error equal to $6.5 \cdot 10^{-4}$ rad was achieved along the whole path by assuming the via-points shown in Fig. 5.2: apart from the changes of direction, a very limited number of via-points are required to mill the entire surface.

As early anticipated, the second set of experiments involved a real industrial manipulator. Two different trajectories were considered. As shown in Fig. 5.3, the first

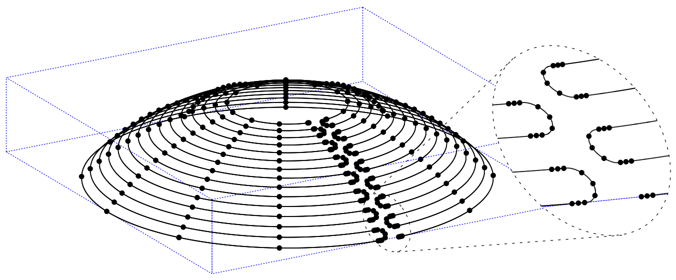


Figure 5.2: The trajectory adopted for the CNC example. Black dots indicate the assigned via-points, the end-effector longitudinal axis (which is not shown in the figure) is aligned to the sphere radius.

trajectory is composed by a circular arc immediately followed by a straight segment. The same figure also shows the assigned orientations of the end-effector frame at the via-points. The two paths are joined through a Cartesian curve still obtained by means of an η^{3D}-spline curve. The whole path is $\mathscr{G}^3$. The jerk reference signals for the six joints, obtained by means of the proposed planner, are shown in Fig. 5.4: as expected, they are continuous. The second manipulator experiment exploits a composite path made of circular arcs, linear segments, conic spirals, and helical curves generated by means of the η^{3D}-splines. As shown in Fig. 5.5, proper orientations have been assigned in each via-point of the path. In order to show the flexibility of the proposed planner, a constant orientation has been assumed for the first segments of the composite trajectory, while the tool orientation changes during the execution of the conic spirals and of the helical curves. More precisely, the tool-frame rotates around its own $\hat{\mathbf{y}}$ axis, so that its $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ axes remain confined inside a vertical plane.

The acquired orientation errors, i.e. the angular differences between the unit vectors of the trajectory frame and the ones of the tool frame, these latter obtained from the encoders readings, are shown in Fig. 5.6. Thanks to the jerk continuity (the corresponding figure has been omitted for conciseness), the absolute values of the errors do not show evident peaks along the whole path but, conversely, they remain confined within a constant strip whose amplitude is equal to $4.2 \cdot 10^{-3}$ rad.

As stated in the introduction, the orientation planner was conceived to be used

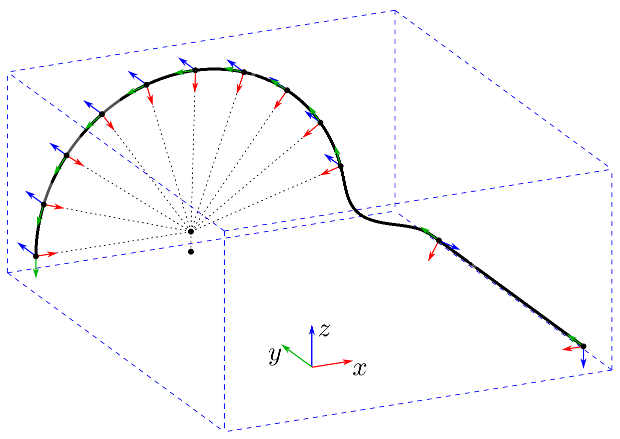


Figure 5.3: The trajectory adopted for the first experiment on the manipulator. Black dots indicate the assigned via points, while the reference frames highlight the desired end-effector orientations.

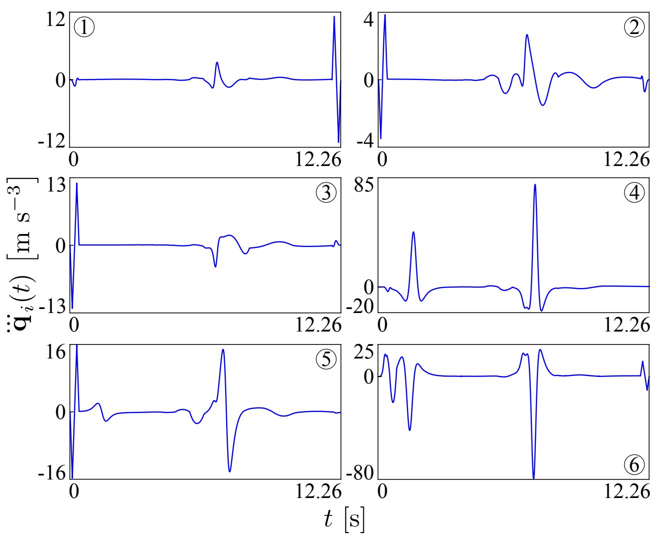


Figure 5.4: Time profiles of the jerks associated to the six joints. The continuity condition is satisfied for all of them.

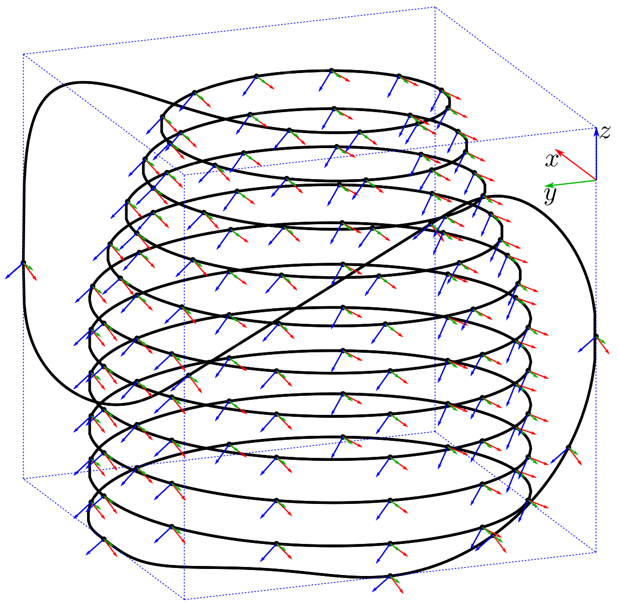


Figure 5.5: The trajectory adopted for the second experiment on the manipulator. Black dots indicate the assigned via-points, while the reference frames highlight the desired end-effector orientations.

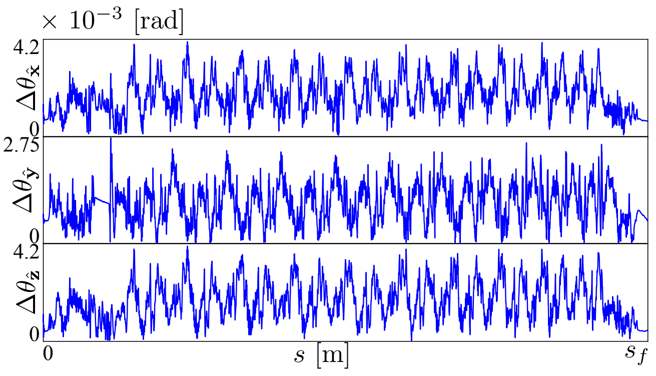


Figure 5.6: Orientation errors of the tool-frame.

in rapidly changing scenarios. Consequently, its computing times have been checked in order to quantify the computational burden. The planning algorithm was run by considering 998 test sets. More in detail, the *n*th set, with $n = 3, 4, \ldots, 1000$, was made of n randomly chosen via-points. Tests were executed on one single core of an Intel i7-1165G7 processor running at @2.80GHz. The investigation made it possible to establish that the computational burden depends linearly on the number of via points. More precisely, the computational time is equal to $n \times 2.19 \cdot 10^{-7}$ s, so that the trajectory shown in Fig. 2.10, which is composed by 127 segments, is planned in $2.781 \cdot 10^{-5}$ s. Consequently, a trajectory made of 4500 points could be processed within 1 millisecond.

Conclusions

This thesis has investigated new methods for the complete trajectory planning within a generic robotic system. Proposed primitives can solve the real-time trajectory planning as they can be used as stand alone planners or, as seen in this thesis, as different parts of the same planner.

η^{3D}-splines are an extremely flexible path planning primitive. As shown in the first part, it can easily emulate other planning primitives and its parameters can be obtained, at a negligible computational cost, directly from the assigned interpolating conditions. These properties, combined with the $\mathscr{G}^3$ geometric continuity that the η^{3D}-splines can guarantee, make it possible to smartly create in real time smooth paths of considerable complexity, by only using a single planning primitive. The research activity has next focused on two open problems. On the one hand, the effort has been posed on the investigation of solutions which exploit the available degrees of freedom for the generation of smoother path, so as to further reduce the errors in the joint space; on the other, η^{3D}-splines have been integrated with an additional primitive, so as to allow the generation of motions which also account for the tool orientations.

A novel method for the management of the corner smoothing problem has been proposed in this thesis. The devised strategy, based on the η^{3D}-splines primitive, allows one generating smooth junctions between straight lines and circular arcs, i.e., the primitives which are typically used in CNC machines, so as to convert the original profiles into $\mathscr{G}^3$ composite paths. The results of test cases highlighted the strengths of the novel smoothing strategy. They can be summarized as follows: simplicity, η^{3D}-

splines coefficients are computed, in a straightforward manner, through closed form expressions; robustness, a solution is always found independently from the interpolating conditions; efficiency, the computational time is compatible with the typical sample times of CNC machines; smoothness, the smoothing curves admit moderate curvatures, sharpnesses, and torsions. The last research on this topic has focused on the simultaneous management of the tool-tip orientation, so as to achieve an integrated planner for the generation of smooth primitives in the operational space.

The orientation planner proposed in this work makes it possible to generate, in a straightforward way, composite trajectories in the operational space. Its combination with the η^{3D}-splines guarantees that the corresponding joint trajectories are $\mathscr{C}^3$. Consequently, smooth joint movements can be achieved and mechanical solicitations can be reduced. The computational burden of the novel primitive is particularly light, so that trajectories with many via-points can be easily handled even in real-time contexts. It is worth to remark that, despite in this thesis the orientation planner has been experimentally tested on an industrial manipulator, it could also be profitably used for the management of CNC machines with more than 3 axes. The approach is evidently scalable, so that a future extension of the work could consider the generation of continuous-snap signals obtained by combining $\mathscr{G}^4$ path curves with $\mathscr{C}^4$ orientation primitives. Another potential future improvement of this work could concern the achievement of trajectories with a specified degree of smoothness, obtained by bounding joint velocities, accelerations, and jerks. Such result can be achieved, for example, by means of scaling techniques acting on the timing law [74].

Summarizing, a whole new trajectory planning strategy has been theorized, developed and tested, and its results are outstanding. The combination between η^{3D}-splines and the orientation planner allows modern robotic systems to reach higher speeds while maintaining the same precision. Another huge result is to have the complete control over the trajectory while the system is running: at any time one could modify or re-plan any of the upcoming tracks without bothering about computational burden or path continuity.

Future works may lead to the integration of this planner in a collision avoidance system in order to modify robots trajectory to avoid any kind of dangerous situation.

Another future work could lead to a study on η^{3D}-splines by running a thorough analysis of the influence that all free parameters have on the final shape of the curve.

Bibliography

[1] L. Dubins, “On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents,” *Amer. J. of Mathematics*, vol. 79, pp. 497–517, 1957.

[2] P. Souères and J.-P. Laumond, “Shortest paths synthesis for a car-like robot,” *IEEE Trans. Auto. Control*, vol. 41, no. 5, pp. 672–688, May 1996.

[3] W. Nelson, “Continuous-curvature paths for autonomous vehicles,” in *IEEE Int. Conf. Robot. and Autom., ICRA89*, vol. 3, Scottsdale, AZ, May 1989, pp. 1260–1264.

[4] Y. Kanayama and B. Hartman, “Smooth local path planning for autonomous vehicles,” *Int. J. of Robot. Research*, vol. 16, no. 3, pp. 263–284, 1997.

[5] R. Campa and H. de la Torre, “Pose control of robot manipulators using different orientation representations: A comparative review,” in *2009 American Contr. Conf.*, 2009, pp. 2855–2860.

[6] J. Reeds and R. Shepp, “Optimal paths for a car that goes both forward and backward,” *Pac. J. of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[7] J.-D. Boissonnat, A. Cérézo, and J. Leblond, “Shortest paths of bounded curvature in the plane,” in *Proc. of the 1992 IEEE Int. Conf. Robot. and Autom.*, Nice, France, May 1992, pp. 2315–2320.

[8] W. Yao, N. Qi, C. Yue, and N. Wan, "Curvature-Bounded Lengthening and Shortening for Restricted Vehicle Path Planning," *IEEE Trans. Autom. Sci. and Eng.*, vol. 17, no. 1, pp. 15–28, Jan 2020.

[9] W. Nelson, "Continuous steering-function control of robot carts," *IEEE Trans. Ind. Electron.*, vol. 36, no. 3, pp. 330–337, Aug 1989.

[10] S. Fleury, P. Souères, J. Laumond, and R. Chatila, "Primitives for smoothing paths of mobile robots," in *Proc. IEEE Int. Conf. Robot. and Autom.*, Atlanta, GA, September 1993, pp. 832–839.

[11] J.-D. Boissonnat, A. Cérézo, and J. Leblond, "A note on shortest paths in the plane subject to a constraint on the derivative of the curvature," INRIA, Rocquencourt, France, Tech. Rep. 2160, January 1994.

[12] J. Reuter, "Mobile robots trajectories with continuously differentiable curvature: an optimal control approach," in *Proc. 1998 IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, vol. 1, Victoria, B.C., Canada, Oct. 1998, pp. 38–43.

[13] G. Yang and B. Choi, "Smooth trajectory planning along Bezier curve for mobile robots with velocity constraints," *Int. J. Control and Autom.*, vol. 6, pp. 225–234, Jan 2013.

[14] L. Zhang, L. Sun, S. Zhang, and J. Liu, "Trajectory planning for an indoor mobile robot using quintic Bezier curves," in *2015 IEEE Int. Conf. on Robot. and Biomim. (ROBIO)*, Dec 2015.

[15] J. Kim, "Trajectory Generation of a Two-Wheeled Mobile Robot in an Uncertain Environment," *IEEE Trans. on Ind. Electron.*, vol. 67, no. 7, pp. 5586–5594, 2020.

[16] T. Berglund, A. Brodnik, H. Jonsson, M. Staffanson, and I. Soderkvist, "Planning Smooth and Obstacle-Avoiding B-Spline Paths for Autonomous Mining Vehicles," *IEEE Trans. Autom. Sci. and Eng.*, vol. 7, no. 1, pp. 167–172, Jan 2010.

[17] A. Piazzi and C. Guarino Lo Bianco, “Quintic G^2-splines for trajectory planning of autonomous vehicles,” in *Procs IEEE Int. Veh. Symp.*, Dearborn (MI), USA, Oct 2000, pp. 198–203.

[18] C. Guarino Lo Bianco, A. Piazzi, and M. Romano, “Smooth motion generation for unicycle mobile robots via dynamic path inversion,” *IEEE Trans. Robot.*, vol. 20, no. 5, pp. 884–891, Oct. 2004.

[19] A. Piazzi, M. Romano, and C. Guarino Lo Bianco, “G^3-Splines for the Path Planning of Wheeled Mobile Robots.” in *Proc. 2003 Eur. Control Conf., ECC 2003*, Cambridge, UK, September 2003.

[20] A. Piazzi, C. Guarino Lo Bianco, and M. Romano, “$\boldsymbol{\eta}^3$-splines for the smooth path generation of wheeled mobile robots,” *IEEE Trans. Robot.*, vol. 23, NO. 5, pp. 1089–1095, 2007.

[21] F. Ghilardelli, G. Lini, and A. Piazzi, “Path Generation Using $\boldsymbol{\eta}^4$-Splines for a Truck and Trailer Vehicle,” *IEEE Trans. Autom. Sci. and Eng.*, vol. 11, no. 1, pp. 187–203, Jan 2014.

[22] W. Ding, W. Gao, K. Wang, and S. Shen, “An Efficient B-Spline-Based Kinodynamic Replanning Framework for Quadrotors,” *IEEE Trans. on Rob.*, vol. 35, no. 6, pp. 1287–1306, 2019.

[23] S. Zhang, L. Sun, Z. Chen, X. Lu, and J. Liu, “Smooth path planning for a home service robot using η^3-splines,” in *IEEE Int. Conf. on Rob. and Biom. (ROBIO)*, 2014, pp. 1910–1915.

[24] Q.-B. Xiao, M. Wan, Y. Liu, X.-B. Qin, and W.-H. Zhang, “Space corner smoothing of CNC machine tools through developing 3D general clothoid,” *Rob. and Comp.-Int. Manuf.*, vol. 64, p. 101949, 2020.

[25] X. Huang, F. Zhao, T. Tao, and X. Mei, “A newly developed corner smoothing methodology based on clothoid splines for high speed machine tools,” *Rob. and Comp.-Int. Manuf.*, vol. 70, p. 102106, 2021.

[26] W. Wang, C. Hu, K. Zhou, S. He, and L. Zhu, “Local asymmetrical corner trajectory smoothing with bidirectional planning and adjusting algorithm for CNC machining,” *Rob. and Comp.-Int. Manuf.*, vol. 68, p. 102058, 2021.

[27] D. Simon and C. Isik, “Optimal trigonometric robot joint trajectories,” *Robotica*, vol. 9, no. 4, pp. 379–386, 1991.

[28] A. Gasparetto and V. Zanotto, “A new method for smooth trajectory planning of robot manipulators,” *Mech. and Mach. Theory*, vol. 42, pp. 455–471, 2007.

[29] S. Kucuk, “Maximal dexterous trajectory generation and cubic spline optimization for fully planar parallel manipulators,” *Computers & Electrical Engineering*, vol. 56, pp. 634–647, 2016.

[30] ——, “Optimal trajectory generation algorithm for serial and parallel manipulators,” *Rob. and Comp.-Int. Manuf.*, vol. 48, pp. 219–232, 2017.

[31] S. Patil, J. Pan, P. Abbeel, and K. Goldberg, “Planning Curvature and Torsion Constrained Ribbons in 3D With Application to Intracavitary Brachytherapy,” *IEEE Trans. Autom. Sci. and Eng.*, vol. 12, no. 4, pp. 1332–1345, Oct 2015.

[32] C. Dai, S. Lefebvre, K. Yu, J. M. P. Geraedts, and C. C. L. Wang, “Planning Jerk-Optimized Trajectory With Discrete Time Constraints for Redundant Robots,” *IEEE Trans. Autom. Sci. and Eng.*, pp. 1–14, Mar 2020.

[33] Y.-A. Lu, K. Tang, and C.-Y. Wang, “Collision-free and smooth joint motion planning for six-axis industrial robots by redundancy optimization,” *Rob. and Comp.-Int. Manuf.*, vol. 68, p. 102091, 2021.

[34] Z. Xin, H. Zhao, S. Wan, l. Xiangfei, and H. Ding, “An Analytical Decoupled Corner Smoothing Method for Five-Axis Linear Tool Paths,” *IEEE Access*, vol. 7, pp. 22 763–22 772, Feb 2019.

[35] Q. Bi, J. Shi, Y. Wang, L. Zhu, and H. Ding, “Analytical curvature-continuous dual-Bezier corner transition for five-axis linear tool path,” *Int. J. Mach. Tools and Manuf.*, vol. 91, pp. 96–108, 2015.

[36] J. Yang and A. Yuen, "An analytical local corner smoothing algorithm for five-axis CNC machining," *Int. J. Mach. Tools and Manuf.*, vol. 123, pp. 22–35, Jul 2017.

[37] J. Huang, Y. Lu, and L.-M. Zhu, "Real-time feedrate scheduling for five-axis machining by simultaneously planning linear and angular trajectories," *Int. J. Mach. Tools and Manuf.*, vol. 135, pp. 78–96, 2018.

[38] A. Gasparetto, A. Lanzutti, R. Vidoni, and V. Zanotto, "Experimental validation and comparative analysis of optimal time-jerk algorithms for trajectory planning," *Rob. and Comp.-Int. Manuf.*, vol. 28, no. 2, pp. 164–181, 2012.

[39] S. Tajima and B. Sencer, "Global tool-path smoothing for CNC machine tools with uninterrupted acceleration," *Int. J. Mach. Tools and Manuf.*, vol. 121, pp. 81–95, 2017.

[40] X. Shen, F. Xie, X.-J. Liu, and Z. Xie, "A smooth and undistorted toolpath interpolation method for 5-DoF parallel kinematic machines," *Rob. and Comp.-Int. Manuf.*, vol. 57, pp. 347–356, 2019.

[41] Z. C. Chen and M. A. Khan, "Piecewise B-Spline Tool Paths With the Arc-Length Parameter and Their Application on High Feed, Accurate CNC Milling of Free-Form Profiles," *J. Manuf. Sci. Eng.*, vol. 134, no. 3, 05 2012.

[42] M. Chen, W.-S. Zhao, and X.-C. Xi, "Augmented Taylor's expansion method for B-spline curve interpolation for CNC machine tools," *Int. J. Mach. Tools and Manuf.*, vol. 94, pp. 109–119, 2015.

[43] S. He, C. Yan, Y. Deng, C.-H. Lee, and X. Zhou, "A tolerance constrained G^2 continuous path smoothing and interpolation method for industrial SCARA robots," *Rob. and Comp.-Int. Manuf.*, vol. 63, p. 101907, 2020.

[44] W. Fan, C.-H. Lee, and J.-H. Chen, "A real-time curvature-smooth interpolation scheme and motion planning for CNC machining of short line segments," *Int. J. Mach. Tools and Manuf.*, vol. 96, pp. 27–46, 2015.

[45] H. Zhao, L. Zhu, and H. Ding, "A parametric interpolator with minimal feed fluctuation for CNC machine tools using arc-length compensation and feedback correction," *Int. J. Mach. Tools and Manuf.*, vol. 75, pp. 1–8, 2013.

[46] J. Shi, Q. Bi, L. Zhu, and Y. Wang, "Corner rounding of linear five-axis tool path by dual PH curves blending," *Int. J. Mach. Tools and Manuf.*, vol. 88, pp. 223–236, 2015.

[47] Q. Hu, Y. Chen, X. Jin, and J. Yang, "A Real-Time C^3 Continuous Local Corner Smoothing and Interpolation Algorithm for CNC Machine Tools," *J. Manuf. Sci. Eng.*, vol. 141, no. 4, 02 2019.

[48] A. Pekarovskiy, T. Nierhoff, S. Hirche, and M. Buss, "Dynamically Consistent Online Adaptation of Fast Motions for Robotic Manipulators," *IEEE Trans. on Rob.*, vol. 34, no. 1, pp. 166–182, 2018.

[49] J. Kim and E. A. Croft, "Online near time-optimal trajectory planning for industrial robots," *Rob. and Comp.-Int. Manuf.*, vol. 58, pp. 158–171, 2019.

[50] T. Hsu and J. Liu, "Design of smooth path based on the conversion between η^3 spline and Bezier curve," in *2020 American Contr. Conf. (ACC)*, 2020, pp. 3230–3235.

[51] K. Kant and S. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, 1986.

[52] E. Kreyszig, *Differential Geometry*. New York: Dover Publications, 1991.

[53] Y. Fang, J. Hu, W. Liu, Q. Shao, J. Qi, and Y. Peng, "Smooth and time-optimal S-curve trajectory planning for automated robots and machines," *Mechanism and Machine Theory*, vol. 137, pp. 127–153, 2019.

[54] C. Guarino Lo Bianco and O. Gerelli, "Generation of paths with minimum curvature derivative with η^3-splines," *IEEE Trans. on Autom. Sci. and Eng.*, vol. 7, no. 2, pp. 249–256, Apr. 2010.

[55] M. Brezak and I. Petrović, “Real-time Approximation of Clothoids With Bounded Error for Path Planning Applications,” *IEEE Trans. on Rob.*, vol. 30, no. 2, pp. 507–515, 2014.

[56] Y. Chen, Y. Cai, J. Zheng, and D. Thalmann, “Accurate and Efficient Approximation of Clothoids Using Bézier Curves for Path Planning,” *IEEE Trans. on Rob.*, vol. 33, no. 5, pp. 1242–1247, 2017.

[57] L. Dozio and P. Mantegazza, “Linux Real Time Application Interface (RTAI) in Low Cost High Performance Motion Control,” in *Motion Control 2003, a conference of ANIPLA National Italian Association for Automation*, Milano, Italy, March 2003.

[58] K. Shoemake, “Animating Rotation with Quaternion Curves,” in *Proc. 12th Conf. on Comput. Graphics and Interactive Techn.*, 1985, pp. 245–254.

[59] M.-C. Ho, Y.-R. Hwang, and C.-H. Hu, “Five-axis tool orientation smoothing using quaternion interpolation algorithm,” *Int. J. of Mach. Tools and Manuf.*, vol. 43, pp. 1259–1267, Sep 2003.

[60] R. Xu, X. Cheng, G. Zheng, and Z. Chen, “A tool orientation smoothing method based on machine rotary axes for five-axis machining with ball end cutters,” *Int. J. of Adv. Manuf. Technol.*, vol. 92, pp. 3615–3625, Oct 2017.

[61] R. M. Grassmann and J. Burgner-Kahrs, “Quaternion-Based Smooth Trajectory Generator for Via Poses in $\boldsymbol{SE(3)}$ Considering Kinematic Limits in Cartesian Space,” *IEEE Rob. and Autom. Letters*, vol. 4, no. 4, pp. 4192–4199, 2019.

[62] Y. Jixiang, L. Dingwei, Y. Congcong, and D. Han, “An analytical C^3 continuous tool path corner smoothing algorithm for 6R robot manipulator,” *Rob. and Comp.-Int. Manuf.*, vol. 64, p. 101947, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584519305976

[63] K. Shoemake, “Quaternion calculus and fast animation, computer animation: 3-d motion specification and control.” Siggraph, 1987.

[64] E. B. Dam, M. Koch, and M. Lillholm, "Quaternions, interpolation and animation," 1998.

[65] G. Legnani, I. Fassi, A. Tasora, and D. Fusai, "A practical algorithm for smooth interpolation between different angular positions," *Mechanism and Machine Theory*, vol. 162, p. 104341, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X21000999

[66] M.-J. Kim, M.-S. Kim, and S. Shin, "A C^2-continous B-spline Quaternion Curve Interpolating a Given Sequence of Solid Orientations," in *Computer Animation*, May 1995, pp. 72–81.

[67] W. Ge, Z. Huang, and G. Wang, "Interpolating Solid Orientations with a C^2-Continuous B-Spline Quaternion Curve," in *Proc. Int. Conf. on Technol. E-learning and Digit. Entertainment*, 2007, pp. 606–615.

[68] Y. Pu, Y. Shi, X. Lin, Y. Hu, and Z. Li, "C^2-Continuous Orientation Planning for Robot End-Effector with B-Spline Curve Based on Logarithmic Quaternion," *Mathematical Problems in Engineering*, vol. 2020, p. 2543824, Jul 2020.

[69] G. Nielson, "v-Quaternion splines for the smooth interpolation of orientations," *IEEE Trans. visualization and comput. graph.*, vol. 10, pp. 224–9, 04 2004.

[70] Y. Liu, Z. Xie, Y. Gu, C. Fan, X. Zhao, and H. Liu, "Trajectory planning of robot manipulators based on unit quaternion," in *2017 IEEE Int. Conf. on Advanced Intelligent Mechatronics (AIM)*, 2017, pp. 1249–1254.

[71] J. Tan, Y. Xing, W. Fan, and P. Hong, "Smooth orientation interpolation using parametric quintic-polynomial-based quaternion spline curve," *Journal of Computational and Applied Mathematics*, vol. 329, pp. 256–267, 2018.

[72] L. Biagiotti and C. Melchiorri, *Trajectory planning for automatic machines and robots*. Heidelberg, Germany: Springer, Berlin, 2008.

[73] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, UK: Cambridge University Press, 1992.

[74] C. Guarino Lo Bianco, M. Faroni, M. Beschi, and A. Visioli, “A predictive technique for the real-time trajectory scaling under high-order constraints,” *IEEE/ASME Trans. on Mechatronics*, vol. 27, no. 1, pp. 315–326, 2022.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Corrado Guarino Lo Bianco for his trust and help during the last three years and for being always open to dialogue for both research and private subjects.

Secondly I would love to dedicate a special thank to Marina Raineri, for being my point of reference since my bachelor thesis, six years ago, and for transmitting her passion everyday.

I also wish to thank my friends and colleagues Davide, Mattia, Irene and Stefano who made work days funnier and less boring but also who could help me when I had need. A thank goes also to Shabnam and Giammarco, who joined our group in the last period.

Out of University there is a bunch of people who deserves my gratitude: starting with my parents who supported me during my whole adventure, continuing to Fabio who stands beside me whatever happened, concluding with my whole family without which I would not be the same person I am today.

Finally, the biggest thank goes to my wife Elena, who always supports and loves me even in the toughest days.