Pré-Publicações do Departamento de Matemática
Universidade de Coimbra
Preprint Number 04–37

A NEW GEOMETRIC ALGORITHM TO GENERATE SMOOTH INTERPOLATING CURVES ON RIEMANNIAN MANIFOLDS

RUI C. RODRIGUES, F. SILVA LEITE AND JANUSZ JAKUBIAK

Abstract: This paper presents a new geometric algorithm to construct a C^k-smooth spline curve that interpolates a given set of data (points and velocities) on a complete Riemannian manifold. Although based on a modification of the de Casteljau procedure, our algorithm is implemented in three steps only independently of the required degree of smoothness, and therefore introduces a significant reduction in complexity. The key role is played by the choice of an appropriate smoothing function which is defined as soon as the degree of smoothness is fixed.

Keywords: Riemannian manifold, Lie group, spline functions, geometric algorithm, Casteljau algorithm.

1. Introduction

We propose a new algorithm, which has a pure geometric interpretation, in order to address the following interpolation problem.

Generate a C^k-smooth ($k \geq 1$) curve $s : [a, b] \subset \mathbb{R} \to \mathrm{M}$, on a complete Riemannian manifold M, which fulfills a set of interpolation conditions of the form

$$s(t_i) = p_i \qquad \text{and} \qquad \dot{s}(t_i) = v_i, \tag{1}$$

for a given partition $\Delta : a = t_0 < t_1 < \cdots < t_m = b$ of the time interval $[a, b]$, given points p_i on M and vectors v_i tangent to M at p_i, $i = 0, 1, \ldots, m$.

Many solutions and some efficient algorithms have been proposed to solve similar interpolation problems (where points and velocities are prescribed), which were motivated by applications in many areas of engineering, like motion planning problems in robotics or object animation which is required in computer graphics. In most cases, M is simply a Lie group or a sphere.

Received December 10, 2004.
All authors were supported by project POSI/SRI/41618/2001 and ISR - Coimbra. R.C. Rodrigues was also supported by a PRODEP grant and J. Jakubiak was also supported by Control Training Site HPMT-CT-2001-00278, during visit to ISR$_{C}oimbra$.

One development was achieved by Shoemake [18], introducing the idea of using quaternions to solve interpolation problems such as the one above, in the sphere S^3. This work was mainly motivated by applications in computer animation and the approach may be generalized to higher-dimensional spheres.

The de Casteljau algorithm [5] is a well known algorithm to generate polynomial curves in Euclidean spaces, based on recursive linear interpolation, which can be used to solve the proposed interpolation problem when the required degree of smoothness is equal to one and the manifold is $\mathbb{R}^n$. See Farin [6] for a general presentation of the de Casteljau method. Other alternatives for Euclidean spaces, proposed by Nagy and Vendel [10] and Rodrigues, Silva Leite and Rosa [15], are based on convex combinations of rather simple curves, like line segments and circular arcs.

There are a number of references on works dealing with Bezier/De Casteljau algorithms on the manifolds $SO(3), S^2, S^3$ and $SE(3)$. In addition to the work by Shoemake [18], we also mention Barr, Currin, Gabriel and Hughes [1], Chen [2], Ge and Ravani [7], Kim, Kim and Shin [9], Nielson [11] and Nielson and Heiland [12]. The objective in most of these papers is to do interpolation on $SO(3)$ using the fact that rotations in $\mathbb{R}^3$ may be represented by unit quaternions. This approach doesn't generalize to higher-dimensional manifolds. In this paper we develop a general method for m-dimensional spheres which also includes the particular case of unit quaternions.

Extension of de Casteljau algorithm to Riemannian manifolds, Lie groups and spheres can be found in the work of Crouch, Kun and Silva Leite [3, 4] and Park and Ravani [13]. The common idea to such generalizations is the replacement of linear interpolation by geodesic interpolation. In Jakubiak, Silva Leite and Rodrigues [8] linear interpolation techniques have been replaced by polynomial interpolation, in order to improve the complexity of the de Casteljau algorithm.

The algorithm we are about to present is performed in three steps only, no matter the degree of smoothness required. It is based on a modification of the de Casteljau construction of a C^1-smooth cubic spline and uses some ideas from the recent work already mentioned. We start with a review of the de Casteljau construction in section 2. Another important feature, which is also shared by the de Casteljau procedure, relies on the fact that the calculation of each spline segment depends only on the local data. This is particularly useful in applications, since any change in the data at a particular instant of time, only requires the re-calculation of two segments of the spline. This is not the case for some classical interpolating spline schemes for which a single change in the data will mean the entire re-calculation of the spline curve.

The new geometric algorithm is first presented and discussed for Euclidean spaces. This will help the visualization of its main features and will motivate its generalization to other complete Riemannian manifolds. However, the algorithm is useful as a computational device only when explicit implementation details of the algorithm are worked out. In general this objective is not reachable, but for some specific cases, like connected and compact Lie groups or spheres, we are able to calculate the interpolating curves in closed form and derive expressions for their derivatives, in order to be able to check the degree of smoothness at the interpolating points.

This paper is the natural evolution and an extended version of the paper by Rodrigues and Silva Leite [14], presented in the minisymposium "Geometric optimization with applications in numerical linear algebra, robotics, and computer vision" at the Conference MTNS2004.

2. The de Casteljau algorithm revisited

The de Casteljau algorithm is a geometric algorithm and one of the most well known algorithms used to generate polynomial spline curves in general Euclidean spaces. Its importance also follows from the simple geometric construction that is performed, which is based on the application of successive linear interpolation.

The classical de Casteljau algorithm is used to construct parameterized polynomial curves, of any given degree d, joining two points in $\mathbb{R}^n$. A sequence of $d-1$ points in $\mathbb{R}^n$ is used to implement the algorithm and for that reason they are called control points. See Farin [6] for details.

We now show how the de Casteljau algorithm can be used to generate a curve in $\mathbb{R}^n$ that fulfills all the interpolation conditions (1). For instance, from the given points p_0, p_1 and the given vectors v_0, v_1 two control points, x_0 and x_1, are uniquely determined by

$$x_0 = p_0 + \tfrac{1}{3}v_0 \qquad \text{and} \qquad x_1 = p_1 - \tfrac{1}{3}v_1.$$

In this case the classical de Casteljau algorithm is performed in three steps and the resulting curve will be a cubic polynomial. For simplicity we may consider the time interval $[0, 1]$ instead of $[t_0, t_1]$.

In the first step, the following three line segments are computed

$$\alpha_0(t) = p_0 + t\,(x_0 - p_0), \qquad \alpha_1(t) = x_0 + t\,(x_1 - x_0), \qquad \alpha_2(t) = x_1 + t\,(p_1 - x_1).$$

Then, two new curves are generated using the curves constructed in the previous step

$$\beta_0(t) = (1-t)\alpha_0(t) + t\,\alpha_1(t), \qquad \beta_1(t) = (1-t)\alpha_1(t) + t\,\alpha_2(t).$$

Remark 2.1. *We can say that β_0 and β_1 are generated from a convex combination of the curves α_0, α_1 and α_1, α_2 respectively.*

These two curves are finally combined in a similar way to generate the cubic polynomial curve given by

$$s_0(t) = (1-t)\beta_0(t) + t\,\beta_1(t) = (1-t)^2\alpha_0(t) + 2t(1-t)\alpha_1(t) + t^2\alpha_2(t).$$

If the classical de Casteljau construction is repeated for each other interval $[t_i, t_{i+1}]$, one obtains a spline curve $t \mapsto s(t)$ in $\mathbb{R}^n$ which is the result of the concatenation of all polynomial segments $t \mapsto s_i(t)$ and has the following final form

$$s(t) = s_i\left(\tfrac{t-t_i}{t_{i+1}-t_i}\right), \qquad t \in [t_i, t_{i+1}], \quad i = 0, 1 \ldots, m-1.$$

The spline curve $t \mapsto s(t)$ is locally a cubic polynomial, satisfies the interpolation conditions (1), but is only C^1-smooth (at each instant t_i).

To generate a C^k-smooth curve using the de Casteljau algorithm, one needs to prescribe k derivatives at each instant t_i. The construction of each spline segment $t \mapsto s_i(t)$ will require $2k$ control points and the computation of $(2k+1)(k+1)$ curves performed in $2k+1$ steps. It is clear that the computational cost of this algorithm increases substantially with k.

The de Casteljau algorithm has been generalized to complete Riemannian manifolds ([4], [13]) and this was mainly due to the fact that the algorithm is geometrically based. The idea is quite simple. The linear interpolation procedure in the classical case is simply replaced by geodesic interpolation. When applied to interpolation problems, the resulting curve has the same degree of smoothness as in the Euclidean case, but the implementation of the algorithm is much harder even for low dimensional cases.

3. A new geometric algorithm in $\mathbb{R}^n$

We first consider the case when M is $\mathbb{R}^n$ equipped with the Euclidean metric. The geometric algorithm proposed here is based on a modification of the de Casteljau algorithm, but has the ability of generating a spline curve in three steps only, with any required degree of smoothness. This property of our algorithm is due to the role played by a *smoothing function* ϕ.

We now show how to compute the curve $t \mapsto s_i(t) \in \mathbb{R}^n$, which connects the point p_i at $t = 0$ to p_{i+1} at $t = 1$ with initial and final velocities v_i and v_{i+1} (again, we use $[0, 1]$ instead of $[t_i, t_{i+1}]$). As in the de Casteljau algorithm we use two control points which are now

$$x_i = p_i + v_i, \qquad x_{i+1} = p_{i+1} - v_{i+1}$$

and three steps. First, we define three line segments:

$$\underline{\text{Step 1}} \quad \begin{cases} l_i(t) = p_i + t\,(x_i - p_i) = p_i + t\,v_i, \\ c_i(t) = p_i + t\,(p_{i+1} - p_i), \\ r_i(t) = x_{i+1} + t\,(p_{i+1} - x_{i+1}) = p_{i+1} + (t-1)\,v_{i+1}, \end{cases} \tag{2}$$

which will play the role of *left*, *center* and *right components* of the spline segment $t \mapsto s_i(t)$. Notice that the left and the right components satisfy

$$\begin{array}{lll} l_i(0) = p_i, & r_i(1) = p_{i+1}, & \\ \dot{l}_i(0) = v_i, & \dot{r}_i(1) = v_{i+1}, & \\ l_i^{(j)}(0) = 0, & r_i^{(j)}(1) = 0, & j \geq 2. \end{array} \tag{3}$$

Remark 3.1. *The three components are such that* $c_i(0) = l_i(0) = p_i$ *and* $c_i(1) = r_i(1) = p_{i+1}$*. This observation helps to visualize the three line segments.*

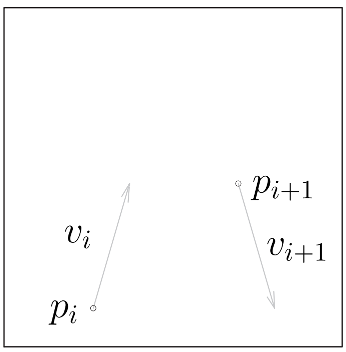

Figure 1: initial data

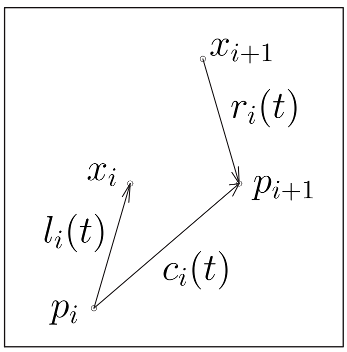

Figure 2: step 1 - components

Next, we introduce a smooth real-valued function $\phi : [0,1] \to [0,1]$ satisfying

$$\begin{array}{lll} \phi(0) = 0, & \phi(1) = 1, & \\ \phi^{(j)}(0) = 0, & \phi^{(j)}(1) = 0, & j = 1, 2, \ldots, k-1, \quad (\text{for } k > 1), \end{array} \tag{4}$$

and compute two new curves from convex combinations (using ϕ) of the ones previously constructed:

$$\underline{\text{Step 2}} \quad \begin{cases} a_i(t) = (1 - \phi(t))\, l_i(t) + \phi(t)\, c_i(t), \\ b_i(t) = (1 - \phi(t))\, c_i(t) + \phi(t)\, r_i(t). \end{cases} \tag{5}$$

Remark 3.2. *These new curves are such that $a_i(0) = b_i(0) = l_i(0) = p_i$, $a_i(1) = b_i(1) = r_i(1) = p_{i+1}$, $\dot{a}_i(0) = \dot{l}_i(0) = v_i$ and $\dot{b}_i(1) = \dot{r}_i(1) = v_{i+1}$. These boundary conditions don't depend on the choice of the function ϕ, as long as ϕ satisfies conditions (4). For the geometric constructions below, which at this point only helps to visualize the steps of the algorithm, we have chosen $\phi(t) = t$. Later, we will explain the relationship between the required degree of smoothness of the spline curve and the choice of the function ϕ.*

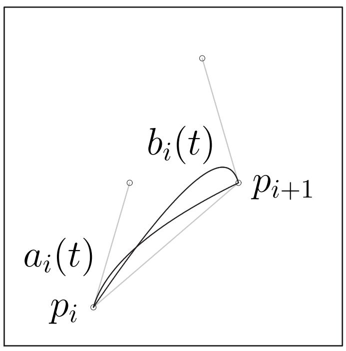

Figure 3: step 2 - curves a_i and b_i

Finally, we combine a_i and b_i in a similar way to generate the spline segment:

$$\underline{\text{Step 3}} \quad \begin{cases} s_i(t) & = & (1 - \phi(t))\, a_i(t) + \phi(t)\, b_i(t) \\ & = & (1 - \phi(t))^2\, l_i(t) + 2\, \phi(t)\, (1 - \phi(t))\, c_i(t) + \phi(t)^2\, r_i(t). \end{cases} \tag{6}$$

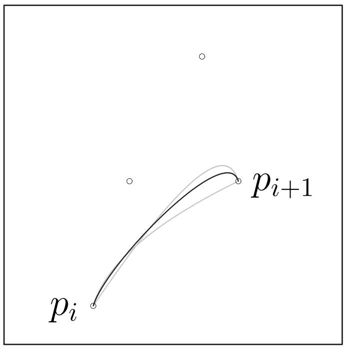

Figure 4: step 3 - curve s_i

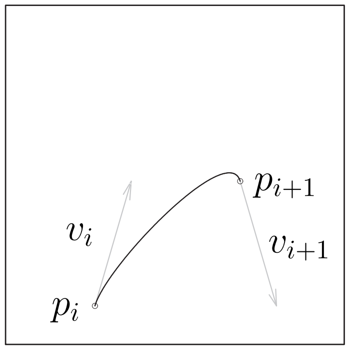

Figure 5: final segment

The next result presents the main properties of the resulting curve $t \mapsto s(t)$.

Theorem 3.1. *If $\phi : [0,1] \to [0,1]$ is a smooth function satisfying (4), then*

(1) *the spline segment $t \mapsto s_i(t)$ defined by (6), (5) and (2) is smooth and satisfies the following conditions*

$$\begin{aligned} &s_i(0) = p_i, \qquad s_i(1) = p_{i+1}, \\ &\dot{s}_i(0) = v_i, \qquad \dot{s}_i(1) = v_{i+1}, \\ &s_i^{(j)}(0) = 0, \qquad s_i^{(j)}(1) = 0, \qquad j = 2, \ldots, k; \end{aligned} \tag{7}$$

(2) *the resulting spline curve $t \mapsto s(t)$ given by*

$$s(t) = s_i\left(\tfrac{t-t_i}{t_{i+1}-t_i}\right), \qquad t \in [t_i, t_{i+1}], \quad i = 0, 1 \ldots, m-1 \tag{8}$$

is C^k-smooth and satisfies the interpolation conditions (1).

Proof: Applying Leibniz's formula for the jth derivative of a product to the formula (6) for the spline segment we get

$$s_i^{(j)}(t) = \sum_{l=0}^{j} \tbinom{j}{l} \left(1 - \phi(t)\right)^{(j-l)} a_i^{(l)}(t) + \sum_{l=0}^{j} \tbinom{j}{l} \phi^{(j-l)}(t)\, b_i^{(l)}(t).$$

To easily check all the boundary conditions we rewrite $s_i^{(j)}$ in a more convenient form. Since $(1-\phi(t))^{(j)} = -\phi^{(j)}(t)$ for $j \geq 1$ the previous formula can be written as

$$s_i^{(j)}(t) = \sum_{l=0}^{j-1} \tbinom{j}{l} \phi^{(j-l)}(t) \left(b_i(t) - a_i(t)\right)^{(l)} + (1 - \phi(t))\, a_i^{(j)}(t) + \phi(t)\, b_i^{(j)}(t).$$

Using the same argument to calculate $a_i^{(j)}(t)$ and $b_i^{(j)}(t)$ we have

$$a_i^{(j)}(t) = \sum_{l=0}^{j-1} \tbinom{j}{l}\, \phi^{(j-l)}(t)\, (c_i(t) - l_i(t))^{(l)} + (1-\phi(t))\, l_i^{(j)}(t) + \phi(t)\, c_i^{(j)}(t).$$

$$b_i^{(j)}(t) = \sum_{l=0}^{j-1} \tbinom{j}{l}\, \phi^{(j-l)}(t)\, (r_i(t) - c_i(t))^{(l)} + (1-\phi(t))\, c_i^{(j)}(t) + \phi(t)\, r_i^{(j)}(t).$$

Therefore,

$$\begin{aligned} s_i^{(j)}(t) = & \sum_{l=0}^{j-1} \tbinom{j}{l}\, \phi^{(j-l)}(t)\, (b_i(t) - a_i(t))^{(l)} \\ & + (1-\phi(t)) \sum_{l=0}^{j-1} \tbinom{j}{l}\, \phi^{(j-l)}(t)\, (c_i(t) - l_i(t))^{(l)} \\ & + \phi(t) \sum_{l=0}^{j-1} \tbinom{j}{l}\, \phi^{(j-l)}(t)\, (r_i(t) - c_i(t))^{(l)} \\ & + (1-\phi(t))^2\, l_i^{(j)}(t) + 2\,\phi(t)\,(1-\phi(t))\, c_i^{(j)}(t) + \phi(t)^2\, r_i^{(j)}(t). \end{aligned}$$

Since ϕ satisfies (4) and $c_i(0) = l_i(0)$, $a_i(0) = b_i(0)$, $c_i(1) = r_i(1)$, $a_i(1) = b_i(1)$ (see Remarks 3.1 and 3.2) we get

$$s_i^{(j)}(0) = l_i^{(j)}(0) \qquad \text{and} \qquad s_i^{(j)}(1) = r_i^{(j)}(1),$$

for all $j = 0, 1, \ldots, k$. The boundary conditions (7) follow from the properties given in (3). The second part of the theorem is a direct consequence of conditions (7). ∎

Remark 3.3 (On the complexity of the algorithm)**.** *It is clear that the complexity of our algorithm does not depend on k. Indeed, if we want a C^k-smooth curve (for $k \geq 2$), our algorithm produces a spline which also satisfies $s^{(j)}(t_i) = 0$, $i = 0, 1 \ldots, m$, for $j = 2, \ldots, k$, in three steps only.*
If these conditions were initially prescribed together with conditions (1), the de Casteljau algorithm could also be used to solve the problem. But, as already observed, the complexity of this algorithm increases substantially with k, since $2k+1$ steps are required.
If we choose $\phi(t) = t$ then the spline generated by our algorithm is C^1-smooth and coincides with the one produced by the de Casteljau algorithm. This is

not a surprise. In fact, by construction, we know that each segment will be a cubic polynomial in $\mathbb{R}^n$ *which is uniquely determined since four data points are given. Each segment of this spline is given by*

$$s_i(t) = p_i + v_i\, t + (3p_{i+1} - 3p_i - 2v_i - v_{i+1})\, t^2 + (2p_i + v_i - 2p_{i+1} + v_{i+1})\, t^3.$$

Remark 3.4 (The smoothing function)**.** *The degree of smoothness of the spline generated by the new algorithm depends only on the choice of a smooth function* $\phi : [0,1] \to [0,1]$*, satisfying (4). This is the reason why we name* ϕ *a "smoothing function" for the spline curve* s*.*
One possible choice for the function ϕ *satisfying all the conditions (4) is the following polynomial function of degree* $2k-1$

$$\phi(t) = \gamma \sum_{l=0}^{k-1} \frac{\alpha_{k+l}}{k+l}\; t^{k+l} \tag{9}$$

where

$$\alpha_{k+l} = (-1)^l \binom{k-1}{l} \qquad \textit{and} \qquad \gamma^{-1} = \sum_{l=0}^{k-1} \frac{\alpha_{k+l}}{k+l}.$$

One observation that will soon be useful is the following

$$\phi^{(k)}(0) = \gamma\, (k-1)! \qquad \textit{and} \qquad \phi^{(k)}(1) = (-1)^{k-1}\, \gamma\, (k-1)!. \tag{10}$$

More properties of this smoothing function may be found in [8]*. From now onwards we will always consider this particular smoothing function. For instance,*

- *for* $k = 1$*, we have* $\phi(t) = t$*;*
- *for* $k = 2$*, we have* $\phi(t) = t^2\,(3 - 2t)$*;*
- *and for* $k = 3$*, one gets* $\phi(t) = t^3\,(10 - 15t + 6t^2)$*.*

For the function ϕ *given by (9), each segment of the generated spline curve is a polynomial curve of degree (at most)* $4k-1$*.*
We note that the spline $t \mapsto s(t)$*, given by (8), when* ϕ *is as defined in (9), will not be* C^{k+1}*-smooth, except for some degenerate cases. To see this, it is enough to compute* $s_i^{(k+1)}(1)$ *and* $s_{i+1}^{(k+1)}(0)$*. We have to distinguish between* $k = 1$ *and* $k > 1$*, since for* $k = 1$ *we get the following formulas*

$$\ddot{s}_i(1) = 2\,(x_i - p_{i+1}) + 4\,(v_{i+1} - p_{i+1} + p_i),$$

$$\ddot{s}_{i+1}(0) = 2\,(x_{i+2} - p_{i+1}) + 4\,(p_{i+2} - p_{i+1} - v_{i+1}),$$

and for $k > 1$ we have

$$s_i^{(k+1)}(1) = 2\,(k+1)\,\phi^{(k)}(1)\,(v_{i+1} - p_{i+1} + p_i),$$
$$s_{i+1}^{(k+1)}(0) = 2\,(k+1)\,\phi^{(k)}(0)\,(p_{i+2} - p_{i+1} - v_{i+1}).$$

Now, $\ddot{s}_i(1) = \ddot{s}_{i+1}(0)$, for some $i \in \{0, 1 \ldots, m-1\}$ if and only if the following holds

$$4\,v_{i+1} = (x_i - x_{i+2}) + 2\,(p_i - p_{i+2}) \quad \Leftrightarrow \quad v_i - 4\,v_{i+1} + v_{i+2} = 3\,(p_{i+2} - p_i),$$

and for $k > 1$, we may use the equalities (10) to conclude that $s_i^{(k+1)}(1) = s_{i+1}^{(k+1)}(0)$, for some $i \in \{0, 1 \ldots, m-1\}$, if and only if

$$\begin{cases} 2\,v_{i+1} = p_{i+2} - p_i & \text{if } k \text{ is odd,} \\ p_{i+1} - p_i = p_{i+2} - p_{i+1} & \text{if } k \text{ is even.} \end{cases}$$

Remark 3.5 (Optimal properties)**.** *For $k = 1$, i.e., $\phi(t) = t$, each component of the spline function given by (8) is an L-spline (of type I) associated with the differential operator $L = \frac{d^2}{dt^2}$, the partition Δ and the incidence vector* $\mathrm{Z} = (z_1, z_2, \ldots, z_{m-1}) = (2, 2, \ldots, 2)$. *See* [17] *for the definition of L-splines and its properties.*

Consequently, if $\langle .,. \rangle$ represents the Euclidean inner product in $\mathbb{R}^n$ and Ω denotes the class of all functions $y : [a, b] \subset \mathbb{R} \to \mathbb{R}^n$ which are C^1-smooth in $[a, b]$ and fulfill the interpolation conditions (1), then the spline function given by (8) and corresponding to $\phi(t) = t$ is the solution of the following optimization problem

$$\min_{y \,\in\, \Omega} \quad \int_a^b \langle \ddot{y}(t), \ddot{y}(t) \rangle \, dt.$$

For $k > 1$ the optimal properties of the spline function produced by our algorithm are still under investigation. However, it is interesting to note that the smoothing function ϕ defined by (9) is the unique solution of the optimization problem

$$\min_{f \,\in\, C^k[0,1]} \quad \int_0^1 \langle f^{(k)}(t), f^{(k)}(t) \rangle \, dt,$$

subject to the following boundary conditions

$$f(0) = 0, \qquad f(1) = 1,$$
$$f^{(j)}(0) = 0, \quad f^{(j)}(1) = 0, \quad j = 1, 2, \ldots, k-1, \quad (\textit{for } k > 1).$$

3.1. Extension to problems with uneven conditions

The new algorithm is easily adapted to the computation of a C^k-smooth ($k \geq 1$) curve s which fulfills a more challenging set of interpolation conditions of the form

$$s(t_i) = p_i, \qquad \dot{s}(t_i) = \dot{p}_i, \qquad \ddot{s}(t_i) = \ddot{p}_i, \qquad \dots \qquad s^{(k_i)}(t_i) = p_i^{(k_i)}, \quad (11)$$

for the partition $\Delta : a = t_0 < t_1 < \cdots < t_m = b$ of the time interval $[a, b]$, points p_i in $\mathbb{R}^n$ and vectors $\dot{p}_i, \ddot{p}_i, \dots, p_i^{(k_i)}$ tangent to $\mathbb{R}^n$ at p_i, with $1 \leq k_i \leq k$ and $i = 0, 1, \dots, m$.
This extension allows uneven prescribed conditions at each instant t_i and is of particular importance in many applications. The only changes required are in the left and the right components of each segment s_i. If k_i is the number of derivatives prescribed at the initial point p_i, then the left component for the segment s_i is a polynomial of degree k_i. If k_{i+1} is the number of derivatives prescribed at the end point p_{i+1}, then the right component for the segment s_i is a polynomial of degree k_{i+1}. Besides these modifications, all remains the same, including the center component. More specifically, to compute the curve $t \mapsto s_i(t)$ that connects the point p_i at $t = 0$ to p_{i+1} at $t = 1$ with prescribed interpolation conditions (11), we use the same control points, which are now written as

$$x_i = p_i + \dot{p}_i, \qquad x_{i+1} = p_{i+1} - \dot{p}_{i+1}$$

and define the left and right components to be the Taylor polynomials

$$\begin{aligned} l_i(t) &= \sum_{j=0}^{k_i} \frac{p_i^{(j)}}{j\,!}\, t^j, \\ r_i(t) &= \sum_{j=0}^{k_{i+1}} \frac{p_{i+1}^{(j)}}{j\,!}\, (t-1)^j. \end{aligned} \quad (12)$$

Notice that l_i and r_i are such that

$$\begin{array}{ll}
l_i(0) = p_i, & r_i(1) = p_{i+1}, \\
\dot{l}_i(0) = \dot{p}_i, & \dot{r}_i(1) = \dot{p}_{i+1}, \\
\quad\vdots & \quad\vdots \\
l_i^{(k_i)}(0) = p_i^{(k_i)}, & r_i^{(k_{i+1})}(1) = p_{i+1}^{(k_{i+1})}, \\
l_i^{(j)}(0) = 0, \quad j > k_i & r_i^{(j)}(1) = 0, \quad j > k_{i+1}.
\end{array}$$

As before, we use a smoothing function ϕ satisfying (4) and compute a_i, b_i and s_i as described in (5)-(6). The next result, similar to Theorem 3.1 follows immediately.

Corollary 3.1. *If $\phi : [0,1] \to [0,1]$ is a smooth function satisfying (4), then*

(1) *the spline segment $t \mapsto s_i(t)$ defined by*

$$s_i(t) = (1-\phi(t))^2\, l_i(t) + 2\,\phi(t)\,(1-\phi(t))\,c_i(t) + \phi(t)^2\, r_i(t),$$

where l_i, r_i are given by (12) and c_i is given by (2), satisfies the following conditions

$$\begin{array}{ll}
s_i(0) = p_i, & s_i(1) = p_{i+1}, \\
\dot{s}_i(0) = \dot{p}_i, & \dot{s}_i(1) = \dot{p}_{i+1}, \\
\quad\vdots & \quad\vdots \\
s_i^{(k_i)}(0) = p_i^{(k_i)}, & s_i^{(k_{i+1})}(1) = p_{i+1}^{(k_{i+1})}, \\
s_i^{(j)}(0) = 0, & s_i^{(j)}(1) = 0, \\
j = k_i + 1, \ldots, k, & j = k_{i+1} + 1, \ldots, k.
\end{array}$$

(2) *the resulting spline curve $t \mapsto s(t)$ given by*

$$s(t) = s_i\left(\tfrac{t-t_i}{t_{i+1}-t_i}\right), \qquad t \in [t_i, t_{i+1}], \quad i = 0, 1 \ldots, m-1$$

is C^k-smooth and satisfies the set of interpolation conditions (11).

4. The new algorithm on complete Riemannian manifolds

In this section we combine the ideas just developed to implement the new algorithm in Euclidean spaces with those used to generalize the de Casteljau algorithm to complete Riemannian manifolds (see details in [3]). In this section we consider the case when only points and first derivatives are prescribed, since for higher derivatives the implementation of the algorithm requires the analogues of higher order polynomials on manifolds. Now, geodesic arcs play the role of straight line segments and, consequently, the algorithm is generally applicable as long as the computation of geodesics is tractable. When the manifold is a compact and connected Lie group G, equipped with the left and right-invariant Riemannian metric, geodesics are easily expressed in terms of one-parameter subgroups. When the manifold is a unit sphere, equipped with the Riemannian metric induced by the Euclidean metric in the embedding space, geodesics are just great circles. We next describe the new algorithm for these two special Riemannian manifolds.

4.1. The new algorithm on Lie groups

In this section G is a connected and compact Lie group, equipped with the unique right and left invariant Riemannian metric, and $\mathscr{L}$ denotes its Lie algebra. Elements in $\mathscr{L}$ will be represented by capital letters.

Similarly to the Euclidean case, the construction of the spline curve that solves the initial problem is local, so that the details will be presented only for the construction of the spline segment $t \mapsto s_i(t)$ that joins two given points in G, p_i (at $t = 0$) and p_{i+1} (at $t = 1$), with prescribed velocities $v_i = V_i p_i$ and $v_{i+1} = V_{i+1} p_{i+1}$, where V_i and V_{i+1} belong to the Lie algebra of G. The smoothing function is the same as in the Euclidean case.

We now describe the three basic steps to obtain the required spline segment $t \mapsto s_i(t)$.

<u>Step 1</u>: We first construct the geodesic segments which are the left, center and right components and are defined by:

$$
\begin{aligned}
l_i(t) &= e^{tV_i} p_i, \\
c_i(t) &= e^{tW_i} p_i, \qquad \text{where} \quad W_i = \log\,(p_{i+1}\, p_i^{-1}), \\
r_i(t) &= e^{(t-1)V_{i+1}} p_{i+1},
\end{aligned}
$$

The following boundary conditions are easily checked:

$$\begin{array}{ll}
l_i(0) = p_i, & l_i(1) = e^{V_i} p_i, \\[2ex]
\dot{l}_i(0) = V_i p_i, & \dot{l}_i(1) = V_i e^{V_i} p_i, \\[2ex]
c_i(0) = p_i, & c_i(1) = p_{i+1} \\[2ex]
\dot{c}_i(0) = W_i p_i, & \dot{c}_i(1) = W_i e^{W_i} p_i, \\[2ex]
r_i(0) = e^{-V_{i+1}} p_{i+1}, & r_i(1) = p_{i+1}, \\[2ex]
\dot{r}_i(0) = V_{i+1} e^{-V_{i+1}} p_{i+1}, & \dot{r}_i(1) = V_{i+1} p_{i+1}.
\end{array} \tag{13}$$

Step 2: Now we define $t \mapsto a_i(t)$ and $t \mapsto b_i(t)$ by:

$$\begin{array}{lll}
a_i(t) = e^{\phi(t) A_i(t)} l_i(t), & \text{where} & A_i(t) = \log\left(c_i(t)\, l_i^{-1}(t)\right), \\[1ex]
b_i(t) = e^{\phi(t) B_i(t)} c_i(t), & \text{where} & B_i(t) = \log\left(r_i(t)\, c_i^{-1}(t)\right).
\end{array} \tag{14}$$

The following alternative formulas for these two curves will simplify checking the boundary conditions of the spline curve at $t = 1$.

$$\begin{aligned}
a_i(t) &= e^{-(1-\phi(t)) A_i(t)} c_i(t), \\
b_i(t) &= e^{-(1-\phi(t)) B_i(t)} r_i(t).
\end{aligned} \tag{15}$$

Indeed,

$$\begin{aligned}
e^{-(1-\phi(t)) A_i(t)} c_i(t) &= e^{\phi(t) A_i(t)} e^{-A_i(t)} c_i(t) \\[2ex]
&= e^{\phi(t) A_i(t)} e^{-\log\left(c_i(t)\, l_i^{-1}(t)\right)} c_i(t) \\[2ex]
&= e^{\phi(t) A_i(t)} l_i(t)\, c_i^{-1}(t)\, c_i(t) \\[2ex]
&= e^{\phi(t) A_i(t)} l_i(t) \\[2ex]
&= a_i(t),
\end{aligned}$$

and similarly for b_i.

Now, using the conditions (4) for the smoothing function ϕ and the boundary conditions (13), we obtain from (14) and (15) the following:

$$\begin{aligned} &a_i(0) = b_i(0) = p_i, && a_i(1) = b_i(1) = p_{i+1}, \\ &\dot{a}_i(0) = V_i p_i, && \dot{a}_i(1) = W_i e^{W_i} p_i, \\ &\dot{b}_i(0) = W_i p_i, && \dot{b}_i(1) = V_{i+1} p_{i+1}. \end{aligned}$$

Step 3: Finally, we define the spline segment $t \mapsto s_i(t)$ by

$$s_i(t) = e^{\phi(t) S_i(t)} a_i(t), \qquad \text{where} \qquad S_i(t) = \log\left(b_i(t)\, a_i^{-1}(t)\right),$$

or, similarly,

$$s_i(t) = e^{\phi(t) S_i(t)} e^{\phi(t) A_i(t)} e^{t V_i} p_i. \tag{16}$$

Theorem 4.1. *If $\phi : [0,1] \to [0,1]$ is a smooth function satisfying (4), then the curve $t \mapsto s_i(t)$ defined by (16) satisfies the following boundary conditions*

$$\begin{aligned} &s_i(0) = p_i, && s_i(1) = p_{i+1}, \\ &\dot{s}_i(0) = V_i p_i, && \dot{s}_i(1) = V_{i+1} p_{i+1}. \end{aligned}$$

Proof: We first derive, from (16), an expression for the first derivative of s_i. Using the Campbell-Hausdorff formula

$$e^{A(t)} B(t)\, e^{-A(t)} = e^{ad\, A(t)} B(t) = \sum_{j=0}^{+\infty} ad^j A(t)(B(t)),$$

where ad denotes the adjoint operator on $\mathscr{L}$ defined by $ad\, A(B) = [A, B]$, and the following formula for the derivative of the exponencial, which may be found in Sattinger and Weaver [16]

$$\frac{d}{dt}\left(e^{A(t)}\right) = \Omega_A^L(t) e^{A(t)}, \quad \text{where} \quad \Omega_A^L(t) = \int_0^1 e^{u\, ad\, A(t)} \dot{A}(t) du,$$

we obtain

$$\begin{aligned} \dot{s}_i(t) = \Big(&\Omega_{\phi S_i}^L(t) + e^{\phi(t) ad\, S_i(t)}\, \Omega_{\phi A_i}^L(t) + \\ &+ e^{\phi(t) ad\, S_i(t)}\, e^{\phi(t) ad\, A_i(t)}\, V_i\Big)\, s_i(t). \end{aligned} \tag{17}$$

The initial conditions $s_i(0) = p_i$ and $\dot{s}_i(0) = V_i\, p_i$, follow easily from (16) and (17), if we take into consideration that $\phi(0) = 0$. To prove that the conditions at $t = 1$ are also satisfied, we rewrite the expression of s_i given in (16), similarly to what has been done for the curves obtained in step 2, to obtain

$$s_i(t) = e^{-(1-\phi(t))S_i(t)} e^{-(1-\phi(t))B_i(t)} e^{-(1-t)V_{i+1}} p_{i+1}. \tag{18}$$

Consequently, an alternative expression for the first derivative is now the following:

$$\begin{aligned}\dot{s}_i(t) = \Big(&\Omega^L_{-\psi S_i}(t) + e^{-\psi(t) ad\, S_i(t)}\, \Omega^L_{-\psi B_i}(t) + \\ &+ e^{-\psi(t) ad\, S_i(t)}\, e^{-\psi(t) ad\, B_i(t)}\, V_{i+1}\Big)\, s_i(t),\end{aligned} \tag{19}$$

where $\psi(t) = 1 - \phi(t)$, so that $\psi(0) = 0$. The final conditions $s_i(1) = p_{i+1}$ and $\dot{s}_i(1) = V_{i+1}\, p_{i+1}$, follow easily from (18) and (19). ■

To show that piecing together the spline segments, the resulting spline curve is C^k-smooth, one needs to derive higher order covariant derivatives. The covariant derivative of a vector field along a curve in G (a manifold imbedded in some high-dimensional Euclidean space $\mathbb{R}^n$) may be viewed as a new vector field along that curve, which results from differentiating as a vector field along a curve in $\mathbb{R}^n$ and then projecting it, at each point, onto the tangent space to G at that point. Details are rather technical and will be omitted in this paper. Nevertheless, when G is the Lie group of rotations $SO(n)$, with Lie algebra $so(n)$ consisting of all $n \times n$ skew-symmetric matrices, the Riemannian metric is defined by $\langle A, B\rangle = trace\,(A^T B)$, $A, B \in so(n)$ and the tangent space at a point $p \in SO(n)$ and its orthogonal complement with respect to $\langle ., . \rangle$ are respectively

$$T_p SO(n) = \{Ap :\ A \in so(n)\} \quad \text{and} \quad T_p^{\perp} SO(n) = \{Sp :\ S \in s(n)\},$$

where $s(n)$ is the set of all $n \times n$ symmetric matrices.

In this case, and after many calculations that are omitted here, we reach the final conclusion, which is a generalization to the Lie group $SO(n)$ of Theorem 3.1.

Theorem 4.2. *If $\phi : [0, 1] \to [0, 1]$ is a smooth function satisfying (4), then*

(1) *the spline segment* $t \mapsto s_i(t) \in SO(n)$ *defined by (16) satisfies the following boundary conditions*

$$s_i(0) = p_i, \qquad s_i(1) = p_{i+1},$$

$$\dot{s}_i(0) = V_i p_i, \qquad \dot{s}_i(1) = V_{i+1} p_{i+1},$$

$$\frac{D^j \dot{s}_i}{dt^j}(0) = 0, \qquad \frac{D^j \dot{s}_i}{dt^j}(1) = 0, \qquad j = 1, \ldots, k-1;$$

(2) *the resulting spline curve* $t \mapsto s(t) \in SO(n)$ *given by*

$$s(t) = s_i\left(\tfrac{t-t_i}{t_{i+1}-t_i}\right), \qquad t \in [t_i, t_{i+1}], \quad i = 0, 1 \ldots, m-1$$

is C^k*-smooth and satisfies the interpolation conditions (1).*

4.2. The new algorithm on spheres

Here we describe the geometric algorithm to construct a C^2- smooth spline curve interpolating a given set of points on the unit sphere S^n, with prescribed velocities through those points. We consider S^n equipped with the Riemannian metric induced by the Euclidean metric in the embedding space $\mathbb{R}^{n+1}$. As before, we start with the construction of a natural spline segment between two points. This means that the second covariant derivatives are zero at the boundary, which consequently guarantees the C^2- smoothness of the interpolating curve, as soon as all the spline segments are glued together. To simplify notations, we avoid the use of indexes for the construction of a generic spline segment. In order to describe the geometric algorithm that generates the natural spline segment $t \in [0, 1] \mapsto s(t) \in S^n$ joining two given points p and q, with prescribed initial and final velocities, we first recall some properties of geodesics on spheres.

Given a point $x_0 \in S^n$ and a vector v_0 tangent to the sphere at x_0, there exists a unique geodesic $t \mapsto x(t)$ that passes through x_0 at time τ, with velocity v_0:

$$x(t) = \cos\left((t-\tau)\, \| \, v_0\|\right) x_0 + \sin\left((t-\tau)\|v_0\|\right) \hat{v_0}, \tag{20}$$

where $\hat{v_0} = \frac{v_0}{\|v_0\|}$.

Also, given two (not antipodal) points $y_0, y_1 \in S^n$, the geodesic arc $t \mapsto y(t)$ which joins y_0 (at $t=0$) to y_1 (at $t=1$) is given by:

$$y(t) = \frac{\sin\left((1-t)\theta_{y_0,y_1}\right)}{\sin\theta_{y_0,y_1}}\, y_0 + \frac{\sin\left(t\theta_{y_0,y_1}\right)}{\sin\theta_{y_0,y_1}}\, y_1, \tag{21}$$

where $\theta_{y_0,y_1} = \cos^{-1}\left({y_0}^T y_1\right)$ is the angle between the vectors y_0 and y_1.

The formula (20) is used to generate the left component $t \mapsto l(t)$ and the right component $t \mapsto r(t)$ of the natural spline segment $t \mapsto s(t)$ that joins the points p (at $t=0$) to q (at $t=1$) with prescribed initial and final velocity v and w respectively. The formula (21) is used to generate the center component $t \mapsto c(t)$ and the intermediate curves in the algorithm below. Taking into consideration that the left component (respectively the right component) is a geodesic satisfying the same conditions as the spline segment at $t=0$ (respectively at $t=1$) and that the center component joins the points p (at $t=0$) and q (at $t=1$), the algorithm is performed in the following three steps:

Step 1: Construct the left, center and right components, defined by:

$$l(t) = \cos\left(t\|v\|\right) p + \sin\left(t\|v\|\right) \hat{v},$$

$$c(t) = \frac{\sin\left((1-t)\theta_{p,q}\right)}{\sin\theta_{p,q}}\, p + \frac{\sin\left(t\theta_{p,q}\right)}{\sin\theta_{p,q}}\, q,$$

$$r(t) = \cos\left((t-1)\|w\|\right) q + \sin\left((t-1)\|w\|\right) \hat{w}.$$

Step 2: Now define $t \mapsto a(t)$ and $t \mapsto b(t)$ using convex combinations (parameterized by the smoothing function ϕ) of the geodesics in the previous step:

$$a(t) = \frac{\sin\left((1-\phi(t))\theta_{l(t),c(t)}\right)}{\sin\theta_{l(t),c(t)}}\, l(t) + \frac{\sin\left(\phi(t)\theta_{l(t),c(t)}\right)}{\sin\theta_{l(t),c(t)}}\, c(t),$$

$$b(t) = \frac{\sin\left((1-\phi(t))\theta_{c(t),r(t)}\right)}{\sin\theta_{c(t),r(t)}}\, c(t) + \frac{\sin\left(\phi(t)\theta_{c(t),r(t)}\right)}{\sin\theta_{c(t),r(t)}}\, r(t).$$

Step 3: Finally, we obtain the required curve:

$$s(t) = \frac{\sin\left((1-\phi(t))\theta_{a(t),b(t)}\right)}{\sin\theta_{a(t),b(t)}}\, a(t) + \frac{\sin\left(\phi(t)\theta_{a(t),b(t)}\right)}{\sin\theta_{a(t),b(t)}}\, b(t). \tag{22}$$

In order to check that the last curve satisfies all the requirements, it is enough to compute the first and second derivatives, and evaluate them at $t = 0$ and $t = 1$. This is a tedious calculation that we omit, but can be easily checked using the following boundary conditions for the curves constructed in this algorithm and for the smoothing function ϕ:

$$\begin{aligned} &s(0) = a(0) = l(0) = p, && s(1) = b(1) = r(1) = q, \\ &\dot{s}(0) = \dot{a}(0) = \dot{l}(0) = v, && \dot{s}(1) = \dot{b}(1) = \dot{r}(1) = w, \\ &\ddot{s}(0) = \ddot{a}(0) = \ddot{l}(0) = -\|v\|^2 p, && \ddot{s}(1) = \ddot{b}(1) = \ddot{r}(1) = -\|w\|^2 q. \end{aligned} \tag{23}$$

The last two expressions imply that that $\ddot{s}(0)$ and $\ddot{s}(1)$ are orthogonal to S^n at p and q respectively, so that the second covariant acceleration vanishes at the boundary points. This observation, together with the other boundary conditions (23), are enough to conclude the following.

Theorem 4.3. *If $\phi : [0, 1] \to [0, 1]$ is any smooth function satisfying (4), then the spline segment $t \mapsto s(t)$ defined by (22) satisfies the following boundary conditions*

$$\begin{aligned} &s(0) = p, && s(1) = q, \\ &\dot{s}(0) = v, && \dot{s}(1) = w, \\ &\frac{D^2\dot{s}}{dt^2}(0) = 0, && \frac{D^2\dot{s}}{dt^2}(1) = 0. \end{aligned}$$

The next figure illustrates the result of applying the algorithm above, to generate a natural spline segment satisfying the following data:

$$\begin{aligned} p &= (\sqrt{3}/2, -\sqrt{3}/4, 1/4), \\ q &= (1/2, \sqrt{3}/4, -3/4), \\ v &= (1/2, \sqrt{3}, 3 - \sqrt{3}), \\ w &= (1/10, 9/5, (9\sqrt{3} - 1)/15). \end{aligned}$$

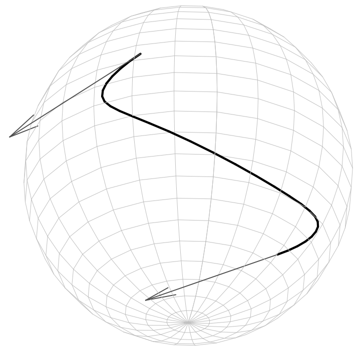

Figure 6: segment s_i

References

[1] A. Barr, B. Currin, S. Gabriel, and J. Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. In *Proc. Computer Graphics (SIGRAPH 92)*, pages 313–320, July 1992.

[2] Chao-Chi Chen. Interpolation of orientation matrices using sphere splines in computer animation. Master of science thesis, Arizona State University, 1990.

[3] P. Crouch, G. Kun, and F. Silva Leite. De Casteljau algorithm for cubic polynomials on the rotation group. In *Proceedings of the Second Portuguese Conference on Automatic Control*, pages 547–552, Porto, Portugal, September 11-13 1996.

[4] P. Crouch, G. Kun, and F. Silva Leite. The De Casteljau algorithm on Lie groups and spheres. *J. Dynam. Control Systems*, 5(3):397–429, 1999.

[5] P. De Casteljau. Outillages méthodes calcule. Technical report, Citroen, A., Paris, 1959.

[6] Gerald Farin. *Curves and surfaces for computer aided geometric design.* Computer Science and Scientific Computing. Academic Press Inc., Boston, MA, third edition, 1993.

[7] Q. J. Ge and B. Ravani. Computer aided geometric design of motion interpolants. In *Proceedings of ASME Design Automation Conf.*, pages 33–41, Miami, September 1991.

[8] J. Jakubiak, F. Silva Leite, and R. C. Rodrigues. A two-step algorithm to generate smooth interpolating splines on Riemannian manifolds. 2004. Submitted for publication.

[9] M. J. Kim, M. S. Kim, and Shin S. Y. A general construction scheme for unit quaternion curves with simple high order derivatives. In *Proc. Computer Graphics, annual conf. series, (SIGRAPH 95)*, pages 369–376, Los Angeles, 1995.

[10] M. S. Nagy and T. P. Vendel. Generating curves and swept surfaces by blended circles. *Computer Aided Geometric Design*, 17:197–206, 2000.

[11] G. Nielson. *Smooth Interpolation of Orientations*, pages 75–93. Models and Techniques in Computer Animation. Springer Verlag, Tokyo, magnenat thalmann, N. and thalmann, D. eds. edition, 1993.

[12] G. Nielson and R. Heiland. *Animated Rotations using Quaternions and Splines on a 4D sphere*, pages 17–27. Programming and Computer Software. Plemum Pub. N.Y., procgrammirovanie (Russia) springer verlag, english edition edition, 1992.

[13] F. C. Park and B. Ravani. Bézier curves on Riemannian manifolds and Lie groups with kinematic applications. *ASME Journal of Mechanical Design*, 117:36–40, 1995.

[14] R. C. Rodrigues and F. Silva Leite. A new geometric algorithm to generate spline curves. In *Proceedings of the Sixteenth International Symposium on Mathematical Theory of Networks*

and Systems (MTNS2004), Katholieke Universiteit Leuven, Belgium, July 5-9 2004. CD-ROM paper 311.PDF.

[15] R. C. Rodrigues, F. Silva Leite, and S. Rosa. On the generation of a trigonometric interpolating curve in $\mathbb{R}^3$. In *Proceedings of the 11th International Conference on Advanced Robotics, ICAR2003*, Coimbra, Portugal, 30/06-03/07 2003. CD-ROM paper N1629.PDF.

[16] D. H. Sattinger and O. L. Weaver. *Lie groups and algebras with applications to physics, geometry, and mechanics*, volume 61 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 1993. Corrected reprint of the 1986 original.

[17] M. H. Schultz and R. S. Varga. L-splines. *Numer. Math.*, 10:345–369, 1967.

[18] K. Shoemake. Animating rotation with quaternion curves. *ACM SIGGRAPH'85*, 19(3):245–254, 1985.

Rui C. Rodrigues
Departamento de Física e Matemática, Instituto Superior de Engenharia, Rua Pedro Nunes, 3030-199 Coimbra, Portugal
E-mail address: ruicr@isec.pt
URL: http://www.isec.pt/∼ruicr

F. Silva Leite
Departamento de Matemática, Universidade de Coimbra, 3001-454 Coimbra, Portugal
E-mail address: fleite@mat.uc.pt
URL: http://www.mat.uc.pt/∼fleite

Janusz Jakubiak
Institute of Engineering Cybernetics, Wroclaw University of Technology, Poland
E-mail address: Janusz.Jakubiak@pwr.wroc.pl