JOURNAL OF DYNAMICAL AND CONTROL SYSTEMS, Vol. 5, No. 3, 1999, 397-429

THE DE CASTELJAU ALGORITHM ON LIE GROUPS AND SPHERES

P. CROUCH, G. KUN, F. SILVA LEITE

ABSTRACT. We examine the De Casteljau algorithm in the context of Riemannian symmetric spaces. This algorithm, whose classical form is used to generate interpolating polynomials in $\mathbb{R}^n$, was also generalized to arbitrary Riemannian manifolds by others. However, the implementation of the generalized algorithm is difficult since detailed structure, such as boundary value expressions, has not been available. Lie groups are the most simple symmetric spaces, and for these spaces we develop expressions for the first and second order derivatives of curves of arbitrary order obtained from the algorithm. As an application of this theory we consider the problem of implementing the generalized De Casteljau algorithm on an m-dimensional sphere. We are able to fully develop the algorithm for cubic splines with Hermite boundary conditions and more general boundary conditions for arbitrary m.

1. INTRODUCTION

The problem of synthesizing a smooth motion of a rigid body or groups of rigid bodies, such as robots, that interpolates a set of configurations in space has considerable importance in many engineering applications. Other direct applications include path planning for aerospace vehicles. In this context, Crouch and Jackson [14], [15], and [16], and Jackson [17] developed a concept of dynamic interpolation, in which the usual interpolation concept is generalized to include the case where the interpolating curves are generated by dynamical systems. Such problems turn out to be very much more

1991 *Mathematics Subject Classification.* 22E15, 53E35.

Key words and phrases. De Casteljau algorithm, symmetric spaces, Lie groups, spheres, polynomial curves, cubic splines, covariant derivatives.

The first author was supported in part by NATO, Grant 4/C/94/PO.

The second author was supported by Acção Integrada Luso-Alemã, No. 314-AI-p-dr.

The third author was supported in part by ISR, research network contract ERB FMRXCT-970137 and Gulbenkian Foundation grant while the author was visiting the Systems Science and Engineering Research Center at the Arizona State University.

1079-2724/99/0700-0397$16.00/0

computationally complex than the same problem in Euclidean polynomial interpolation, where computationally efficient algorithms are available to compute the interpolating polynomials. One of these algorithms is the De Casteljau algorithm. In essence this algorithm is a geometric construction, whereby two points in $\mathbb{R}^m$ are joined by a polynomial via an iterative linear interpolation process. It is indeed remarkable that this successive linear interpolation does indeed yield a polynomial.

The power of this algorithm lies in the fact that, since it is geometrically based, it can be easily generalized from $\mathbb{R}^m$ to other spaces, as long as the linear interpolation process is suitably redefined.

The goal of this paper is to develop details of the De Casteljau algorithm in the special cases of connected and compact Lie groups and spheres. That one could generalize the concept of the De Casteljau algorithm to arbitrary Riemannian manifolds was first pointed out by Park and Ravani [31], where usual straight line segments are replaced by geodesic segments. However, the algorithm is useful as a computational device only when explicit implementation details of the algorithm are worked out. This is our overall objective in the paper, for some specific cases where we are able to calculate the geodesic flows, and derive expressions for the derivatives of the generalized polynomial curves obtained from the algorithm. This does not seem to be done elsewhere in the literature. While we would like to be able to work out explicit details for general Riemannian manifolds, the problem is already hard for general compact Lie groups and spheres S^m, $m > 2$ The class of spaces known as the Riemannian symmetric spaces includes these examples and seems to be the right class of spaces for a general theory pertaining to the curves produced by the De Casteljau algorithm, although we do not pursue this degree of generality here.

The first objective of our paper is to work out a general expression for the first two derivatives of generalized polynomial curves defined by the generalized De Casteljau algorithm, for the special case of compact Lie groups. Thus, this result holds specifically for all of the orthogonal groups $\mathrm{SO}(m)$, $m \geq 3$, and, in particular, the rotation group $\mathrm{SO}(3)$ in $\mathbb{R}^3$.

While most of our analysis holds for more general Lie groups where the logarithm is defined, such as $\mathrm{SE}(3)$, the group of rigid motions in $\mathbb{R}^3$, there is no natural invariant means of differentiating, and, therefore, any construction will be dependent upon specific choices such as the metric.

The second objective of our paper is to work out the details of the generalized De Casteljau algorithm on m-dimensional spheres $S^m, m \geq 2$. The Lie group $\mathrm{SO}(m+1)$ acts transitively on S^m, and geodesics on S^m correspond to certain geodesics on $\mathrm{SO}(m+1)$, which are one-parameter subgroups in $\mathrm{SO}(m+1)$. Thus the algorithm for S^m can be based upon the somewhat simpler algorithm for $\mathrm{SO}(m+1)$. Computing the geodesics, or

one-parameter subgroups, on $SO(m+1)$ for the algorithm in general requires the computation of matrix exponentials and logarithms, which for $m > 2$ is computationally intensive. However for SO(3) both computations reduce to evaluating analytic expressions. For S^m, $m \geq 2$, it turns out that only matrix exponentials and logarithms in SO(3) need to be computed, and hence the computation of geodesics is again tractable. The more interesting feature of computing generalized polynomials on S^m, is in dealing with the boundary conditions. One must carefully adjust the boundary conditions of the corresponding generalized polynomials on $SO(m+1)$ so that the projected curves on S^m meet the required boundary constraints. For Hermite boundary values, where initial and final velocities are prescribed in addition to initial and final positions, we are able to completely solve the problem of generating third order generalized polynomial curves on S^m, $m \geq 2$, which meet the desired boundary conditions. In particular, this is accomplished for S^2 and SO(3), recovering the algorithm outlined in Chen [8] for S^2. However, for general interpolation problems, where one is required to piece together many segments of third order generalized polynomials, the Hermite boundary value problem does not have a recursive solution. Thus boundary value problems where one specifies initial position, velocity, and acceleration, together with final position are more convenient. In [13] we were able to solve for third order generalized polynomials, with these boundary values on S^2 and SO(3). In the present article we generalize the results of [13] for general S^m, $m \geq 3$. The principal practical interest in such algorithms is to develop interpolation techniques for $SO(3), S^2$, and SE(3) although much interest has been demonstrated in developing the technique for S^3 viewed as the space of unit quarternions and a convenient parameterization of SO(3). Indeed, dual quarternions have been employed by Jüttler [24] to accommodate SE(3). However, it is clear that the current literature fails to develop a satisfactory means of dealing with C^k smoothness, $k \geq 1$, because of the difficulty in dealing with closed form solutions for the derivatives. This has been achieved in a limited sense in Ge and Ravini [22], but not in a general framework applicable in a wide variety of problems. There are a number of references on works dealing with Bezier/De Casteljau algorithms on manifolds, but usually for the spaces $SO(3), S^2, S^3$, and SE(3). In addition to the work by Park and Ravani [31] and others cited above, we also mention Shoemake [33], Ge and Ravani [20], Nielson [27] and [28], Kim, Kim, and Shin [25], Barr, Currin, Gabriel, and Hughes [1], and Chen [8]. The objective in most of these papers is to do interpolation on SO(3) using the fact that rotations in $\mathbb{R}^3$ can be represented by unit quaternions. However, the De Casteljau algorithm in S^3 is developed in a way that cannot be generalized to higher order dimensional spheres. The work of Shoemake [33] is however an exception. He uses an alternative formula for the geodesic arc

on S^3 which can be generalized to higher-dimensional spheres. In this paper we develop a general method for m-dimensional spheres which also includes the particular case of unit quaternions.

These references also fail to tackle the smoothness of the interpolants in a completely satisfactory manner. Clearly the smoothness of a curve in general depends upon its parameterization. For curves on general Riemannian manifolds, one can develop the notion of arc length, induced by the Riemannian structure. By also developing a means to differentiate which is compatible with this metric structure, the so-called Riemannian covariant derivative, one can consider the C^k, $k \geq 1$, smoothness of curves relative to the arc length parameterization. This can be considered as a measure of the intrinsic smoothness of the curve, as measured by the Riemannian structure. Generalizing the De Casteljau algorithm using geodesics parameterized by arc length, one can develop, in theory, generalized splines of arbitrary intrinsic smoothness. Other authors, such as Kim, Kim, and Shin [25], simply avoid these issues by reparametrizing "simple" interpolating curves, so that the required degree of smoothness be obtained, without affecting the intrinsic smoothness. While the detail of the analysis presented here may seem overly complicated, it is generally applicable and may find use outside of the spaces S^2, S^3, SO(3), and SE(3). In particular, it provides a framework in which derivatives of arbitrary order can be derived if necessary.

Interpolating curves on Riemannian manifolds can be achieved in more ways than as prescribed by the De Casteljau algorithm described here. For example, C^2 cubic splines in $\mathbb{R}^n$ obey a variational principle, see Farin [19]. In Crouch and Silva Leite [9] and Camarinha, Crouch, and Silva Leite [5] and [6] this idea is generalized to Riemannian manifolds to define another potential class of generalized C^2 cubic splines. We show here that in the case of compact abelian groups, this class of curves coincides with the class of curves produced by the generalized De Casteljau algorithm. However, this is certainly not the case for Lie groups such as SO(3). Moreover, the class of curves defined by the variational problem does not seem so computationally tractable as those developed by the De Casteljau algorithm. The solution curves to the variational problems have their own intrinsic quality. Indeed, in application areas such as robotics and path planning, such curves describe natural motions of the mechanical system, as derived from Newton's laws. While we do not discuss our work on these interpolating curves in this paper, we do compare the curves derived from the De Casteljau algorithm with those obtained via the variational principle and make some interesting observations.

2. De Casteljau algorithm and polynomial interpolation

In this section we review the classical De Casteljau algorithm introduced by De Casteljau [18] and Bezier [2] for the generalization of polynomial curves in Euclidean space and recall the reader a generalization to Riemannian manifolds as proposed by Park and Ravani [31]. We start with the general construction on a Riemannian manifold M, from which the classical case is obtained by making $M = \mathbb{R}^n$, and then on a Lie group G. We use the standard text of Farin [19] to serve as a background in the case of splines on $\mathbb{R}^n$. Following the classical case, and the work of Park and Ravani [31], the basic definition of the De Casteljau algorithm is as follows. If we are given a set $\{x_0, \cdots, x_n\}$ of distinct points in a Riemannian manifold M, a smooth curve $t \to \beta_n(t, x_0, \cdots, x_n) := \beta_n(t)$ and M, joining x_0 (at $t = 0$) and x_n (at $t = 1$), can be constructed by successive geodesic interpolation as follows. We assume that M is geodesically comlete. If $\beta_1(t, x_i, x_{i+1})$ is the geodesic arc joining x_i (at $t = 0$) and x_{i+1} (at $t = 1$), then we set

$$\begin{gathered} \beta_k(t, x_i, \cdots, x_{i+k}) = \\ = \beta_1\left(t, \beta_{k-1}(t, x_i, \cdots, x_{i+k-1}), \beta_{k-1}(t, x_{i+1}, \cdots, x_{i+k})\right), \\ k = 1, \cdots, n; \quad i = 0, \cdots, n-k. \end{gathered} \tag{1}$$

We refer to β_n as a generalized polynomial of degree n in M.
If $M = \mathbb{R}^m$, then

$$\begin{gathered} \beta_0(t, x_i) = x_i \ \ \forall i = 0, \cdots, n, \\ \beta_k(t, x_i, \cdots, x_{i+k}) = \\ = (1-t)\beta_{k-1}(t, x_i, \cdots, x_{i+k-1}) + t\beta_{k-1}(t, x_{i+1}, \cdots, x_{i+k}), \\ k = 1, \cdots, n; \quad i = 0, \cdots, n-k. \end{gathered}$$

This curve satisfies the following boundary conditions:

$$\begin{gathered} \beta_n(0) = x_0, \quad \beta_n(1) = x_n, \\ \frac{d^r}{dt^r}\beta_n(0) = \frac{n!}{(n-r)!}\Delta^r x_0, \quad \frac{d^r}{dt^r}\beta_n(1) = \frac{n!}{(n-r)!}\Delta^r x_{n-r}, \end{gathered} \tag{2}$$

where $\Delta^r x_i = \sum_{j=0}^{r} \binom{cr}{j} (-1)^{r-j} x_{i+j}$. However, it does not pass through the other points $x_1, \cdots x_{n-1}$. These points will be called hereafter *control points.*

Cubic polynomials, in particular, can be defined through the De Casteljau algorithm, by setting $n = 3$. In this case, from the last formulas one obtains

$$\frac{d}{dt}\beta_3(0) = 3(x_1 - x_0), \quad \frac{d^2}{dt^2}\beta_3(0) = 6\{(x_2 - x_1) - (x_1 - x_0)\}.$$

These two identities give a relation between the first and second derivatives at $t = 0$ and the points x_0, x_1, x_2, and x_3 in the following way. Given points x_0, x_3 and vectors v, w in $\mathbb{R}^m$, if one wants to construct a cubic polynomial $t \to p(t)$ satisfying

$$p(0) = x_0, \quad p(1) = x_3, \quad \dot{p}(0) = v, \quad \ddot{p}(0) = w, \tag{3}$$

simply choose the control points needed to apply the De Casteljau algorithm as $x_1 = \frac{1}{3}v + x_0$ and $x_2 = \frac{1}{6}w + \frac{2}{3}v + x_0$. Then $\beta_3(t, x_0, x_1, x_2, x_3)$ is the required $p(t)$. This analysis also works for higher degree polynomials (of odd degree $2k-1$), as long as the initial and final positions and the first k derivatives of $p(t)$ at $t = 0$ are prescribed. For polynomials of even degree $2k$ the same procedure applies, except that in this case, to guarantee existence, the first k derivatives at $t = 0$ are no longer independent. For instance, for $k = 1$, a quadratic polynomial satisfying (3) only exists if $v = -2w$. We refer to Farin [19] for more detail concerning the theory of Bézier curves.

The interval $[0, 1]$ can be replaced by any other interval $[T_1, T_2]$, $T_1 < T_2$, just by choosing the reparametrization $(t \to s)$ defined by $t = (s - T_1)/(T_2 - T_1)$. Also, if instead of prescribing x_0, x_3 and the first and second derivatives at $t = 0$, we prescribe the initial and final points and the first derivatives at $t = 0$ and $t = 1$, then the control points which produce the unique spline satisfying $p(0) = x_0$, $p(1) = x_3$, $\dot{p}(0) = v$, and $\dot{p}(1) = w$, are given by $x_1 = \frac{1}{3}v + x_0$ and $x_2 = -\frac{1}{3}w + x_3$. This analysis was considered in our previous paper [12]. However, the previous viewpoint considered above has computational advantages whenever instead of polynomial curves one is interested in spline curves in which the boundary data is not symmetrically specified, as we now explain.

The C^2 cubic spline curve p on $[0, T]$, in $\mathbb{R}^n$, is usually defined as the curve satisfying the following interpolation problem. Given a partition, $0 = T_0 < T_1 < \cdots < T_k = T$, distinct points $s_0, s_1, \ldots, s_k$ and vectors v_0 and v_1,

(a) $p(T_i) = s_i$, $0 \le i \le k$,

(b) $\dot{p}(0) = v_0$, $\dot{p}(T) = v_1$,

(c) $\dot{p}(T_i^-) = \dot{p}(T_i^+)$, $\ddot{p}(T_i^-) = \ddot{p}(T_i^+)$, $1 \le i \le k-1$,

(d) $p|_{[T_i, T_{i+1}]} = P_{i+1}$, $0 \le i \le k-1$, is a cubic polynomial.

Conditions (a), (b), and (c) supply respectively $2kn$, $2n$ and $2(k-1)n$ boundary values to the n-cubic polynomials specified in (d), which is the total of $4kn$ boundary conditions required to completely specify the problem.

Note that the symmetrically specified data (b) implies that the polynomial curves P_i cannot be generated recursively in the order $P_1, P_2, \ldots, P_k$. The problem is truly a two-point boundary value problem. In the Euclidean

case this is not an immediate computational problem, see Farin [19]. However, in the non-Euclidean case this issue becomes much more involved. By replacing the boundary data (b) by conditions similar to (3),

$$(b') \quad \dot{p}(0) = v_0, \quad \ddot{p}(0) = w_0,$$

the polynomial curves $P_1, P_2, \cdots P_k$, can indeed be calculated recursively using the formulas above. One can use a "shooting" method to solve the problem specified by the boundary data (b') and varying the value of the vector w_0.

Blindly using the schemes above does however lead to interpolating curves which sometimes display wild departures from the set of interpolating data, as explained in Farin [19]. There are many techniques one can employ to compensate for this problem. These techniques are equally applicable to the algorithms generated for Riemannian manifolds.

A few remarks should be made concerning the general applicability of the De Casteljau construction. Although the geometry of a Riemannian manifold possesses enough structure to formulate the construction, the basic ingredients used, the geodesic arcs, are implicitly defined by a set of nonlinear differential equations. Thus the basic algorithm can be only practically implemented when we can reduce the calculation of these geodesics to a manageable form.

In the case of compact Lie groups, the geodesics are just one-parameter subgroups and hence for matrix compact Lie groups the computation of a geodesic is just exponentiation of a matrix. We now restrict ourselves in this section to the Lie group case and compute the first two derivatives at the boundary values, to generalize the expressions (2) in the case $M = \mathbb{R}^m$.

Given $x_0, x_1, \cdots, x_n$, distinct points in G, let V_k^1, $k = 0, \cdots, n-1$ be the infinitesimal generators of the geodesic curves on G joining the points x_k at time $t = 0$ and x_{k+1} at $t = 1$, that is,

$$x_{k+1} = \exp(V_k^1)x_k, \quad k = 0, \cdots, n-1, \tag{4}$$

and for every $t \in [0,1]$ and $k = 0, \cdots, n-1$, define Lie algebra elements $V_k^j(t)$, by:

$$\exp(V_k^j(t)) = \exp(tV_{k+1}^{j-1}(t)) \exp((1-t)V_k^{j-1}(t)), \quad j \geq 2. \tag{5}$$

Now, for every $t \in [0,1]$, define a sequence of points in G recursively by:

$$p_0(t) = x_0,$$
$$p_k(t) = \exp(tV_0^k(t))p_{k-1}(t), \quad k \geq 1.$$

In particular, for $k = n$, one obtains

$$p_n(t) = \exp(tV_0^n(t)) \exp(tV_0^{n-1}(t)) \cdots \exp(tV_0^2(t)) \exp(tV_0^1)x_0. \tag{6}$$

Indeed, $p_n(t) = \beta_n(t, x_0, \cdots x_n)$. We now begin our analysis of these curves. The following two results are essentially contained in Park, and Ravani [31].

Lemma 2.1. *The Lie algebra elements defined in* (4) *and* (5) *satisfy the following boundary conditions*

$$\begin{aligned} V_k^j(0) &= V_k^1, \\ V_k^j(1) &= V_{k+j-1}^1 \end{aligned} \qquad \forall k \in \mathbb{N}_0, \ \ \forall j \in \mathbb{N},$$

and the following symmetry

$$\exp(tV_k^n)\exp((t-1)V_k^{n-1}) = \exp((t-1)V_k^n)\exp(tV_{k+1}^{n-1}) \ \ \forall n, k.$$

Theorem 2.2. *The polynomial curve $t \to p_n(t)$ in G defined in* (6) *by*

$$p_n(t) = \exp(tV_0^n(t))\exp(tV_0^{n-1}(t))\cdots\exp(tV_0^2(t))\exp(tV_0^1)x_0$$

satisfies the following boundary conditions:

$$p_n(0) = x_0, \quad p_n(1) = x_n.$$

However, the following result is not quite so obvious and is essential in calculating the derivatives of the generalized polynomial curves at the endpoint $t = 1$.

Theorem 2.3. *For the polynomial curve $t \to p_n(t)$ in G defined in* (6) *we can write*

$$\begin{aligned} p_n(t) = \exp((t-1)V_0^n)\exp((t-1)V_1^{n-1})\ldots \\ \ldots\exp((t-1)V_{n-2}^2)\exp((t-1)V_{n-1}^1)x_n. \end{aligned}$$

Proof. We use induction on n. First of all, using (6) with $n = 1$, and also Eq. (4), we can write

$$p_1(t) = \exp(tV_0^1)x_0 = \exp((t-1)V_0^1)\exp(V_0^1)x_0 = \exp((t-1)V_0^1)x_1,$$

which shows that the result is true for $n = 1$. Now assume that it is true for $p_{n-1}(t)$. Thus, since $p_n(t) = \exp(tV_0^n)p_{n-1}(t)$, one obtains

$$\begin{aligned} p_n(t) = \exp(tV_0^n)\exp((t-1)V_0^{n-1})\exp((t-1)V_1^{n-2})\ldots \\ \ldots\exp((t-1)V_{n-2}^1)x_{n-1}. \end{aligned}$$

Applying repeatedly, the previous lemma we can write

$$\begin{aligned}
p_n(t) &= \exp((t-1)V_0^n)\exp(tV_1^{n-1})\exp((t-1)V_2^{n-2})\dots \\
&\quad \dots\exp((t-1)V_{n-2}^1)x_{n-1} = \\
&\quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
&= \exp((t-1)V_0^n)\exp((t-1)V_1^{n-1})\dots \\
&\quad \dots\exp(tV_{n-2}^2)\exp((t-1)V_{n-2}^1)x_{n-1} = \\
&= \exp((t-1)V_0^n)\exp((t-1)V_1^{n-1})\dots \\
&\quad \dots\exp((t-1)V_{n-2}^2)\exp(tV_{n-1}^1)x_{n-1} = \\
&= \exp((t-1)V_0^n)\exp((t-1)V_1^{n-1})\dots \\
&\quad \dots\exp((t-1)V_{n-1}^1)x_n,
\end{aligned}$$

which completes the proof of the theorem. □

Now, for each $t \in [0,1]$, we define another sequence of points $t \to q_k^n(t)$, $k = 0, \cdots, n$, in G recursively by

$$\begin{aligned}
q_0^n(t) &= x_n, \\
q_k^n(t) &= \exp((t-1)V_{n-k}^k(t))q_{k-1}^n(t), \quad k \geq 1.
\end{aligned}$$

In particular, for $k = n$, one obtains

$$\begin{aligned}
q_n^n(t) = \exp((t-1)V_0^n(t))\exp((t-1)V_1^{n-1}(t))\cdots \\
\dots\exp((t-1)V_{n-1}^1)x_n,
\end{aligned} \tag{7}$$

which, according to Theorem 2.3, is exactly equal to $p_n(t)$.

The following figure illustrates the situation for the 4^{th} order case. It is clear that the whole figure is symmetric with respect to t and $1-t$ in the following sense. The same point $p_n(t)$ can be reached in two different ways: either traveling forwards from x_0, along geodesic arcs passing through $p_1(t)$, $p_2(t), \cdots, p_n(t)$ at the instants of time $t, 2t, \cdots, nt$ respectively, or traveling backwards from x_n, along geodesic arcs passing through $q_1^n(t)$, $q_2^n(t), \cdots, q_n^n(t) = p_n(t)$ at the instants of time $(1-t)$, $2(1-t), \cdots, n(1-t)$ respectively.

The next task is to compute the derivatives of the polynomial curves at $t = 0$ and $t = 1$. For this one needs formulas for the derivatives with respect to t of $\exp(A(t))$, where $A(t) \in \mathcal{L}\ \forall t$, $\mathcal{L}$ is a matrix Lie algebra, which are presented in the next lemma. The proof follows Sattinger and Weaver [32], and indicates the complexity of the arguments and proofs.

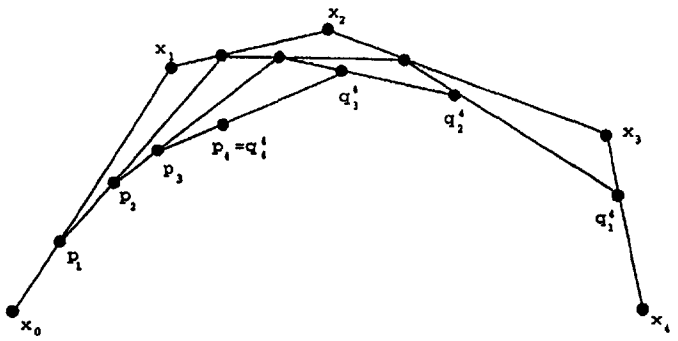


Fig. 1

Lemma 2.4. *If $A(t) \in \mathcal{L}$ is of class C^1, then the following identities hold*:

$$(1) \quad \frac{d}{dt}\exp(A(t)) = \Omega_A^L(t)\exp(A(t)), \tag{8}$$

$$\text{where} \quad \Omega_A^L(t) = \int_0^1 \exp(u\,\mathrm{ad}A(t))\dot{A}(t)\,du, \tag{9}$$

$$(2) \quad \frac{d}{dt}\exp(A(t)) = \exp(A(t))\Omega_A^R(t), \tag{10}$$

$$\text{where} \quad \Omega_A^R(t) = \int_0^1 \exp(-u\,\mathrm{ad}A(t))\dot{A}(t)du. \tag{11}$$

Theorem 2.5. *The derivatives of the polynomial curve $t \to p_n(t)$ in G defined in* (6) *satisfy the following boundary conditions*:

$$\frac{d}{dt}p_n(t)\Big|_{t=0} = nV_0^1 x_0, \qquad \frac{d}{dt}p_n(t)\Big|_{t=1} = nV_{n-1}^1 x_n.$$

Proof. According to (6), $p_n(t)$ can be written as

$$p_n(t) = \exp(tV_0^n(t))\exp(tV_0^{n-1}(t))\cdots\exp(tV_0^2(t))\exp(tV_0^1)x_0.$$

Thus, applying Lemma 2.4(1), it follows that $\frac{d}{dt}p_n(t) = \Omega(t)p_n(t)$, with

$$\begin{aligned}\Omega(t) = \Omega^L_{tV_0^n} &+ \exp(t\,\mathrm{ad}V_0^n)\Omega^L_{tV_0^{n-1}} + \\ &+ \exp(t\,\mathrm{ad}V_0^n)\exp(t\,\mathrm{ad}V_0^{n-1})\Omega^L_{tV_0^{n-2}} + \dots \\ &\dots + \exp(t\,\mathrm{ad}V_0^n)\exp(t\,\mathrm{ad}V_0^{n-1})\cdots\exp(t\,\mathrm{ad}V_0^2)\Omega^L_{tV_0^1}.\end{aligned}$$

But it also follows from (9) that

$$\begin{aligned}\Omega^L_{tV_0^k}(0) &= \int_0^1 \exp(u\,\mathrm{ad}(tV_0^k))(V_0^k(t) + t\dot{V}_0^k(t))du\Big|_{t=0} = \\ &= V_0^k(0) = V_0^1 \quad \forall k,\end{aligned}$$

and, consequently, $\Omega(0) = nV_0^1$ and $\frac{d}{dt}p_n(0) = \Omega(0)p_n(0) = nV_0^1x_0$.

To prove the second identity we use the other expression for $p_n(t)$ given by Theorem 2.3 which, after taking $s = t - 1$, becomes

$$p_n(s+1) = \exp(sV_0^n(s+1))\exp(sV_1^{n-1}(s+1))\cdots\exp(sV_{n-1}^1(s+1))x_n.$$

We then have $\frac{d}{dt}p_n(t)\Big|_{t=1} = \frac{d}{ds}p_n(s+1)\Big|_{s=0} = \Theta(s)\Big|_{s=0} p_n(1)$, where

$$\begin{aligned}\Theta(s) = \Omega^L_{sV_0^n} &+ \exp(s\,\mathrm{ad}V_0^n(s+1))\Omega^L_{sV_1^{n-1}} + \dots \\ &\dots + \exp(s\,\mathrm{ad}V_0^n(s+1))\dots\exp(s\,\mathrm{ad}V_{n-2}^2(s+1))\Omega^L_{sV_{n-1}^1}\end{aligned} \tag{12}$$

and $V_j^k = V_j^k(s+1)\ \forall j,k$. Now we take into consideration that, according to (9) and Lemma 2.1, $\Omega^L_{sV_j^k(s+1)}\Big|_{s=0} = V_j^k(1) = V^1_{k+j-1}$ and also note that, for all terms of the form $V_j^k(s+1)$ in the expression of $\Theta(s)$ above, we always have $j + k = n$. Therefore, $\Theta(0) = nV^1_{n-1}$ and, consequently, $\frac{d}{dt}p_n(t)\Big|_{t=1} = nV^1_{n-1}x_n$, which completes the proof. □

Lemma 2.6. *If $\Omega^L_{tV_0^k}$ is obtained from (9) replacing $A(t)$ by $tV_0^k(t)$, then*

$$\frac{d}{dt}\Omega^L_{tV_0^k}\Big|_{t=0} = 2\dot{V}_0^k(0). \tag{13}$$

Proof.

$$\frac{d}{dt}\Omega^L_{tV_0^k}\Big|_{t=0} = \frac{d}{dt}\left(\int_0^1 \exp(ut\,\mathrm{ad}V_0^k)(t\dot{V}_0^k + V_0^k)\,du\right)\Big|_{t=0} =$$

$$= \frac{d}{dt}\left(V_0^k + \int_0^1 \exp(ut\,\mathrm{ad}V_0^k)(t\dot{V}_0^k)\,du\right)\Big|_{t=0} =$$

$$= \dot{V}_0^k(0) + \int_0^1 \Big((u\,\mathrm{ad}V_0^k + ut\,\mathrm{ad}\dot{V}_0^k)\exp(ut\,\mathrm{ad}V_0^k)(t\dot{V}_0^k) +$$

$$+ \exp(ut\,\mathrm{ad}V_0^k)(\dot{V}_0^k + t\ddot{V}_0^k)\Big)\,du\Big|_{t=0} =$$

$$= \dot{V}_0^k(0) + \int_0^1 \dot{V}_0^k(0)\,du = 2\dot{V}_0^k(0). \quad \square$$

Lemma 2.7.

$$\frac{d}{dt}\exp(-t\,\mathrm{ad}V_k^j(t))\Big|_{t=0} = -\mathrm{ad}\,V_k^j(0) \quad \forall k, j. \tag{14}$$

Proof. It is an immediate consequence of Lemma 2.4 that

$$\frac{d}{dt}\exp(-t\,\mathrm{ad}V_k^j(t)) = \exp(-t\,\mathrm{ad}V_k^j(t))\Lambda(t),$$

with

$$\Lambda(t) = \int_0^1 \exp\Big(-u\,\mathrm{ad}_0(-t\,\mathrm{ad}V_k^j(t))\Big)\,(-\mathrm{ad}V_k^j(t) - t\,\mathrm{ad}\dot{V}_k^j(t))\,du.$$

thus, evaluating at $t = 0$, the result follows. $\square$

Lemma 2.8.

(1) $$\int_0^1 \exp(u\,\mathrm{ad}V_0^1)\dot{V}_0^k(0)\,du = (k-1)(V_1^1 - V_0^1) \quad \forall k \in \mathbb{N},$$

(2) $$\int_0^1 \exp(-u\,\mathrm{ad}V_{n-1}^1)\dot{V}_{n-k}^k(1)\,du = (k-1)(V_{n-1}^1 - V_{n-2}^1) \quad \forall k \in \mathbb{N}.$$

Proof (*By induction on k*). Identity (1) is clearly true for $k = 1$, since V_0^1 is constant. Now suppose that the identity holds for k. According to (5),

$$\exp(V_0^{k+1}) = \exp(tV_1^k)\exp((1-t)V_0^k).$$

Differentiating both sides of this equality, using Lemma 2.4(1), and then evaluating at $t = 0$ we obtain

$$\Omega^L_{V_0^{k+1}}(0)\exp(V_0^{k+1}(0)) = \left(\Omega^L_{tV_1^k}(0) + \Omega^L_{(1-t)V_0^k}(0)\right)\exp(V_0^k(0)).$$

But, according to Lemma 2.1, $V_0^{k+1}(0) = V_0^k(0) = V_0^1$ which implies

$$\Omega^L_{V_0^{k+1}}(0) = \Omega^L_{tV_1^k}(0) + \Omega^L_{(1-t)V_0^k}(0).$$

Now the result follows by using the definition of Ω^L and the induction assumption. The proof of the second identity can be done similarly, just using Lemma 2.4(2) instead of Lemma 2.4(1). □

Theorem 2.9. *If $t \to p_n(t)$ is the polynomial curve in G defined in* (6), *then:*

$$\left.\frac{D^2}{dt^2}p_n(t)\right|_{t=0} = \frac{n!}{(n-2)!}\Upsilon_0(V_1^1 - V_0^1)x_0,$$
$$\left.\frac{D^2}{dt^2}p_n(t)\right|_{t=1} = \frac{n!}{(n-2)!}\Upsilon_1(V_{n-1}^1 - V_{n-2}^1)x_n,$$

where Υ_0 and Υ_1 are respectively the inverses of the operators

$$\int_0^1 \exp(u\,\mathrm{ad}V_0^1)\,du \quad \textit{and} \quad \int_0^1 \exp(-u\,\mathrm{ad}V_{n-1}^1)\,du.$$

Before proving the theorem we show that the inverses of the operators just mentioned exist, at least when V_0^1 and V_{n-1}^1 are small. Indeed, if $W = \mathrm{ad}V_0^1$ and $\|\exp(W) - I\| < 1$ the power series $\sum_{m=0}^{+\infty} \frac{W^m}{(m+1)!}$ and $\sum_{m=0}^{+\infty}(-1)^m \frac{(\exp W - I)^m}{(m+1)}$ converge to $f(W) = \frac{\exp W - I}{W}$ and $g(\exp W) = \frac{\log(\exp W)}{\exp(W) - I}$ respectively. But $f(W)g(\exp W) = I$ and since $f(-W) = \int_0^1 \exp(-u\,\mathrm{ad}V_0^1)\,du$, it follows that the first operator above is invertible for V_0^1 small. If $\mathrm{ad}V_0^1$ is replaced by $\mathrm{ad}V_{n-1}^1$, the same conclusion can be reached for the second operator above.

Proof. Since $\frac{d}{dt}p_n(t) = \Omega(t)p_n(t)$ with

$$\begin{aligned}\Omega(t) = {} & \Omega^L_{tV_0^n} + \exp(t\,\mathrm{ad}V_0^n)\Omega^L_{tV_0^{n-1}} + \\ & + \exp(t\,\mathrm{ad}V_0^n)\exp(t\,\mathrm{ad}V_0^{n-1})\Omega^L_{tV_0^{n-2}} + \dots \\ & \dots + \exp(t\,\mathrm{ad}V_0^n)\exp(t\,\mathrm{ad}V_0^{n-1})\cdots\exp(t\,\mathrm{ad}V_0^2)\Omega^L_{tV_0^1},\end{aligned}$$

it follows easily from Lemmas 2.6 and 2.7 and also the identity $V_0^k(0) = V_0^1$ $\forall k$, that

$$\begin{aligned}\dot\Omega(0) = {} & 2\sum_{k=1}^{n}\dot V_0^k(0) - \mathrm{ad}V_0^n(0)V_0^{n-1}(0) + \\ & + \mathrm{ad}V_0^n(0)\mathrm{ad}V_0^{n-1}(0)V_0^{n-2}(0) + \dots \\ & \dots + (-1)^{n-1}\mathrm{ad}V_0^n(0)\cdots\mathrm{ad}V_0^2(0)V_0^1(0) = 2\sum_{k=1}^{n}\dot V_0^k(0).\end{aligned}$$

But, it follows from Lemma 2.8(1) that

$$\dot V_0^k(0) = (k-1)\left(\int_0^1 \exp(u\,\mathrm{ad}V_0^1)du\right)^{-1}(V_1^1 - V_0^1) = (k-1)\Upsilon_0(V_1^1 - V_0^1).$$

Therefore, replacing this in the previous expression, one obtains the first formula in the theorem.

The formula for the second covariant deivative at $t = 1$ can be proved similarly. Indeed, since $\frac{D^2}{dt^2}p_n(t)\Big|_{t=1} = \frac{D^2}{ds^2}p_n(s+1)\Big|_{s=0}$ and $\frac{d}{ds}p_n(s+1) = \Theta(s)p_n(s+1)$, with $\Theta(s)$ given by (12), it follows by applying Lemmas 2.6 and 2.7 that

$$\begin{aligned}\dot\Theta(s+1)\Big|_{s=0} = {} & 2\{\dot V_0^n(1) + \dot V_1^{n-1}(1) + \dots + \dot V_{n-1}^1(1)\} - \\ & - \mathrm{ad}V_0^n(1)\Omega^L_{sV_1^{n-1}}(1) + \mathrm{ad}V_0^n(1)\mathrm{ad}V_1^{n-1}(1)\Omega^L_{sV_2^{n-2}}(1) + \dots \\ & \dots + (-1)^{n-1}\mathrm{ad}V_0^n(1)\mathrm{ad}V_1^{n-1}(1)\cdots\mathrm{ad}V_{n-2}^2(1)\Omega^L_{sV_{n-1}^1}(1).\end{aligned}$$

But $\Omega^L_{sV_k^{n-k}}(1) = V_k^{n-k} = V_{n-1}^1$ $\forall k$, implies that all the terms in the last expression are zero, except the first one which is equal to $2\sum_{k=1}^{n}\dot V_{k-1}^{n-k+1}(1) = 2\sum_{k=2}^{n}\dot V_{n-k}^k(1)$, since V_{n-1}^1 is constant. Now from Lemma 2.8(2) we have

$\dot{V}^k_{n-k}(1) = (k-1)\Upsilon_1(V^1_{n-1} - V^1_{n-2})$ and, therefore,

$$\frac{D^2}{dt^2}p_n(t)\Big|_{t=1} = 2\sum_{k=2}^{n}(k-1)\Upsilon_1(V^1_{n-1} - V^1_{n-2})x_n =$$
$$= \frac{n!}{(n-2)!}\Upsilon_1(V^1_{n-1} - V^1_{n-2})x_n. \quad \square$$

Remark 2.10. Comparing these formulas for the second covariant derivatives with the classical formulas given in (2) for $r = 2$, the only difference is that the present case also involves the operators Υ_0 and Υ_1.

3. De Casteljau algorithm for $SO(m+1)$

In the last section we addressed the problem of finding a polynomial curve for Lie groups G, given a sequence of points in G. However, in most cases we are given some boundary data but not the points $x_0, x_1, \ldots x_n$. Depending on the given boundary data, there are two interesting methods to consider the problem.

Now we consider the implementation of the De Casteljau algorithm for the Lie groups $SO(m+1)$, $m \geq 2$. For the generalized cubic splines, we show how the control points can be obtained from the boundary data. Case 1 is conceptually the simplest, corresponding to Hermite boundary conditions. In Case 2 the data is not symmetrically specified. This case is particularly important in practical applications due to computational advantages over the first method whenever one is interested in piecing together several polynomial curves. More detail can be found in Crouch, Silva Leite, and Kun [13].

Case 1 ($SO(m+1)$). The boundary data are x_0, $x_3, \frac{dp_3}{dt}(0), \frac{dp_3}{dt}(1)$. Here x_0 and x_3 belong to $SO(m+1)$ and

$$\frac{dp_3}{dt}(0) = \Omega_0 x_0, \quad \frac{dp_3}{dt}(1) = \Omega_1 x_3,$$

where Ω_0 and Ω_1 belong to $so(m+1)$.

It follows from Theorem 2.5 that the two control points x_1 and x_2 are given by (4) as

$$x_1 = \exp(V^1_0)x_0, \quad x_2 = \exp(-V^1_2)x_3,$$

where $V^1_0 = 1/3\Omega_0$ and $V^1_2 = 1/3\Omega_1$.

Case 2 ($\mathrm{SO}(m+1)$). The boundary data are $x_0, x_3, \frac{dp_3}{dt}(0), \frac{D^2p_3}{dt^2}(0)$. In this case x_0 and x_3 also belong to $\mathrm{SO}(m+1)$ and

$$\frac{dp_3}{dt}(0) = \Omega_0 x_0, \quad \frac{D^2p_3}{dt^2}(0) = \Omega_2 x_0,$$

where Ω_0 and Ω_2 belong to $\mathrm{so}(m+1)$.

It follows from Theorems 2.5 and 2.9 that the two control points x_1 and x_2 are given by (4) as

$$x_1 = \exp(V_0^1)x_0, \quad x_2 = \exp(V_1^1)x_1,$$

where

$$V_0^1 = \frac{1}{3}\Omega_0, \quad V_1^1 = V_0^1 + \frac{1}{6}\Upsilon_0^{-1}(\Omega_2)$$

and $\Upsilon_0^{-1} = \int_0^1 \exp(u \operatorname{ad} V_0^1)\, du$ is a linear operator on the linear space $\mathrm{so}(m+1)$. Thus, apart from the evaluation of Υ_0^{-1}, we have reduced the evaluation of the generalized cubic polynomials to matrix algebra. Theoretically the implementation of the algorithm can proceed but, as already mentioned above it depends heavily on the ability to compute matrix exponentials and matrix logarithms.

For SO(3), the polynomial curves derived from the De Casteljau algorithm of degree n are given in (6). Thus, the computation of $p_n(t)$ given distinct points $x_0, \ldots x_n$ in SO(3) depends upon being able to calculate matrix exponentials in so(3) and logarithms in SO(3). In this case, the matrix exponentials and logarithms are given by simple expressions. If $v, w \in \mathbb{R}^3$ then the cross product in $\mathbb{R}^3$ can be expressed in the form $v \times w = S_v w$, where S_v is a 3×3 skew symmetric matrix.

The Lie algebra of SO(3) is so(3), the vector space of skew symmetric 3×3 matrices. For $S_v \in \mathrm{so}(3)$ and $R \in \mathrm{SO}(3)$ we have the following expressions:

$$e^{S_v} = I \cos\|v\| + \frac{\sin\|v\|}{\|v\|} S_v + \frac{I - \cos\|v\|}{\|v\|^2} vv^T, \tag{15}$$

$$\log R = \frac{\alpha}{2\sin\alpha}(R - R^T), \tag{16}$$

where $\cos\alpha = (\operatorname{trace}(R) - 1)/2$. (When $\operatorname{trace}(R) = -1$, log is not uniquely defined.)

We refer to Crouch, Kun, and Silva Leite [12] for detail concerning the implementation of the De Casteljau algorithm on SO(3) The following example illustrates the use of this algorithm in SO(3) with boundary data as in Case 2, but applied to a slightly more complicated problem of interpolating 3 points, s_0, s_1, and s_2 in SO(3) with the data as shown below. The

result is illustrated by Fig. 2. Relative size delineates distance from the viewer.

$$s_0 = p(0) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$s_1 = \begin{bmatrix} 0 & -1 & 0 \\ \frac{1}{2} & 0 & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & 0 & \frac{1}{2} \end{bmatrix},$$

$$s_2 = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & -1 \end{bmatrix},$$

$$V = \frac{d}{dt}p(t)\Big|_{t=0} = \begin{bmatrix} 0 & -\frac{\pi}{2} & 0 \\ \frac{\pi}{2} & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix},$$

$$W = \frac{D^2}{dt^2}p(t)\Big|_{t=0} = \begin{bmatrix} 0 & \frac{\pi}{4} & 0 \\ -\frac{\pi}{4} & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}.$$

Finally, for the general case $SO(m+1)$, $m \geq 3$, the polynomial curves $p_n(t)$ derived from the De Casteljau algorithm of degree n are again given in (6). However, for large m, there are no analytic formulas as in (15) and (16). Although theoretically one could treat the boundary value problems as we did for SO(3), the algorithm soon becomes computationally intensive. At this point we would like to mention recent developments of explicit formulas for the exponential, and of stable numerical methods for computing logarithms on matrix Lie groups, that can be used successfuly in the implementation of the De Casteljau algorithm on Lie groups. (See Crouch and Silva Leite [11] and Cardoso and Silva Leite [7]).

4. De Casteljau algorithm for S^m

In this section we consider the implementation of the De Casteljau algorithm for the spheres S^m, $m \geq 2$.

Fig. 2. The rotation along a cubic spline on SO(3)

In Crouch, Silva Leite, and Kun [13] we presented Case 1 and Case 2 for the 2-dimensional sphere, based on the observation that SO(3) acts transitively on S^2. Here we will show how to implement the generalized De Casteljau algorithm for spheres of any dimension and address the most interesting feature of dealing with boundary conditions.

There are two methods of considering. In the first we simply treat S^m as a Riemannian manifold equipped with the metric induced by the Euclidean metric in $\mathbb{R}^{m+1}$, compute the geodesics directly, and then employ the De Casteljau algorithm as given in Sec. 2. This is essentially a generalization of the approach taken by Nielson [27] and Chen [8] for the 2-dimensional case. This method serves to illustrate some interesting features of non-Euclidean splines, that will be discussed later, in Sec. 5. The main problem with this approach is in dealing with non-symmetrical boundary conditions, due to difficulties of computing the second derivatives of the formulas derived below. To overcome this problem one can use the fact that $\mathrm{SO}(m+1)$ acts transitively on the sphere S^m, so that a polynomial curve on the sphere is the projection of a particular polynomial curve on the rotation group. We will see how the boundary conditions on both the Lie group and the sphere are related so that we can use the somewhat simpler formulas for second derivatives on $\mathrm{SO}(m+1)$ to deal with the non-symmetrical boundary conditions on S^m.

The De Casteljau algorithm relies on the ability to compute geodesics on the sphere that join two points, say x_i (at $t = 0$) and x_{i+1} (at $t = 1$). As

for the 2-dimensional sphere, such curve is given by the following formula

$$\beta_1(t, x_i, x_{i+1}) = \frac{\sin((1-t)\theta_i^1)}{\sin\theta_i^1} x_i + \frac{\sin(t\theta_i^1)}{\sin\theta_i^1} x_{i+1}, \tag{17}$$

where $\theta_i^1 = \cos^{-1}(x_i^T x_{i+1})$ is the angle between the vectors x_i and x_{i+1}. This formula can be written in the following equivalent form

$$\beta_1(t, x_i, x_{i+1}) = \exp(t\theta_i^1 S_{x_i, x_{i+1}}) x_i, \tag{18}$$

where

$$S_{x_i, x_{i+1}} = -P_{x_i, x_{i+1}}^T A_{12} P_{x_i, x_{i+1}},$$

A_{12} is the elementary skew-symmetric matrix $E_{12} - E_{21}$ and

$$P_{x_i, x_{i+1}} = [\ \hat{x}_1 \quad \hat{x}_2 \quad \cdots \quad \hat{x}_{m-1}\ ]$$

denotes any orthogonal matrix satisfying $\hat{x}_1 = x_i$ and $\hat{x}_2 \in \operatorname{span}\{x_i, x_{i+1}\}$. Although the matrix $P_{x_i, x_{i+1}}$, which can be constructed applying the Gram–Schmidt algorithm, is not unique, the geodesic curve in (18) is unique, as long as x_i and x_{i+1} are not antipodal points.

Now it is clear that the following theorem holds.

Theorem 4.1. *Suppose we are given a set of points $\{x_0, x_1, \ldots, x_n\}$ in S^m, and the resulting nth order generalized polynomial $p_n(t) \in S^m$, obtained from the points $x_0, x_1, \ldots, x_n$ by the De Casteljau algorithm of Sec. 3. Then, there exists a set of points $\{g_1, \ldots, g_n\}$ in $\mathrm{SO}(m+1)$, such that the generalized polynomial $g_n(t) \in \mathrm{SO}(m+1)$ obtained by the De Casteljau algorithm from the set of points $\{g_0, g_1, \ldots, g_n\}$, where $g_0 = e$ = identity, satisfies*

$$ex_0 = x_0, \quad g_1 x_0 = x_1, \quad g_2 x_0 = x_2, \quad \ldots, \quad g_n x_0 = x_n$$

and

$$g_n(t) x_0 = p_n(t), \quad t \in [0, 1].$$

Now we can proceed with the De Casteljau algorithm on S^m. Using (17) or (18) for the geodesic arc joining two points, the generalized cubic polynomials on the sphere can be defined as follows. Set

$$\begin{aligned} &\theta_i^1 := \cos^{-1}\langle x_i, x_{i+1}\rangle, \quad i = 0, 1, 2, \\ &\theta_i^2(t) := \cos^{-1}\langle \beta_1(t, x_i, x_{i+1}), \beta_1(t, x_{i+1}, x_{i+2})\rangle, \quad i = 0, 1, \\ &\theta_0^3(t) := \cos^{-1}\langle \beta_R 2(t, x_0, x_1, x_2), \beta_2(t, x_1, x_2, x_3)\rangle. \end{aligned} \tag{19}$$

$$f(t, \theta(t)) := \frac{\sin(t\theta(t))}{\sin\theta(t)}, \qquad g(t, \theta(t)) := \frac{\sin(1-t)\theta(t))}{\sin\theta(t)}. \tag{20}$$

Then,

$$\begin{aligned} p(t) = {} & g(t,\theta_0^3(t))g(t,\theta_0^2(t))g(t,\theta_0^1)x_0 + \\ & + \{g(t,\theta_0^3(t))g(t,\theta_0^2(t))f(t,\theta_0^1) + \\ & + g(t,\theta_0^3(t))f(t,\theta_0^2(t))g(t,\theta_1^1) + \\ & + f(t,\theta_0^3(t))g(t,\theta_1^2(t))g(t,\theta_1^1)\}\, x_1 + \\ & + \{g(t,\theta_0^3(t))f(t,\theta_0^2(t))f(t,\theta_0^1) + \\ & + f(t,\theta_0^3(t))g(t,\theta_1^2(t))f(t,\theta_1^1) + \\ & + f(t,\theta_0^3(t))f(t,\theta_1^2(t))g(t,\theta_2^1)\}x_2 + \\ & + f(t,\theta_0^3(t))f(t,\theta_1^2(t))f(t,\theta_2^1)x_3. \end{aligned} \tag{21}$$

The functions f and g defined in (20), together with their derivatives, satisfy:

$$\begin{aligned} & f(0) = 0, \quad g(0) = 1, \quad f(1) = 1, \quad g(1) = 0, \\ & \frac{df}{dt}(0) = \frac{\theta(0)}{\sin\theta(0)}, \qquad \frac{df}{dt}(1) = \frac{\theta(1)\cos\theta(1)}{\sin\theta(1)}, \\ & \frac{dg}{dt}(0) = \frac{-\theta(0)\cos\theta(0)}{\sin\theta(0)}, \quad \frac{dg}{dt}(1) = \frac{-\theta(1)}{\sin\theta(1)}. \end{aligned} \tag{22}$$

We can directly compute the following result.

Theorem 4.2. *The cubic polynomial, $t \to p(t)$, on S^m defined by* (21) *satisfies the following boundary value conditions.*

$$\begin{aligned} & p(0) = x_0, \quad \frac{dp}{dt}(0) = 3\frac{d}{dt}\beta_1(t,x_0,x_1)\Big|_{t=0}, \\ & p(1) = x_3, \quad \frac{dp}{dt}(1) = 3\frac{d}{dt}\beta_1(t,x_2,x_3)\Big|_{t=1}. \end{aligned} \tag{23}$$

Using this result, we can treat the Hermite boundary data (Case 1) by directly computing the generalized cubic polynomials that satisfy the boundary conditions

$$\begin{aligned} & p(0) = x_0, \qquad p(1) = x_3, \\ & \frac{dp}{dt}(0) = \hat{V}_0, \quad \frac{dp}{dt}(1) = \hat{V}_1 \end{aligned}$$

where $\hat{V}_0$ is tangent to S^m at x_0, so that $x_0^T\hat{V}_0 = 0$, and $\hat{V}_1$ is tangent to S^m at x_3, so that $x_3^T\hat{V}_1 = 0$.

Deriving the analogue of Case 2 is more complicated, since we need to generalize Theorem 4.2 to second derivatives. An alternative way is to

combine the results in Theorems 4.1 and 4.2 for producing cubic polynomials on the sphere S^m, as projections of cubic polynomials on $\mathrm{SO}(m+1)$. The problem encountered with this method now involves the situation where, as usual, we are not given the points in S^m, but only initial and final points, together with derivatives. As in $\mathrm{SO}(m+1)$ we consider two cases, and only cubic polynomials, $n = 3$.

Case 1 (S^m). The boundary data are x_0, x_3, $\frac{dp_3}{dt}(0)$, $\frac{dp_3}{dt}(1)$. Here x_0 and x_3 are unit vectors in $\mathbb{R}^{m+1}$ and

$$\frac{dp_3}{dt}(0) = \hat{V}_0, \quad \frac{dp_3}{dt}(1) = \hat{V}_1$$

are vectors in $\mathbb{R}^{m+1}$, with $\hat{V}_0$ tangent to S^m at x_0 and $\hat{V}_1$ tangent to S^m at x_3, so that $\hat{V}_0^T x_0 = 0$ and $\hat{V}_1^T x_3 = 0$.

To make use of Theorem 4.1, we must identify the points g_1, g_2, and g_3 in $\mathrm{SO}(m+1)$ and points x_1 and x_2 in $S^m \subset \mathbb{R}^{m+1}$ such that

$$g_1 x_0 = x_1, \quad g_2 x_0 = x_2, \quad g_3 x_0 = x_3,$$
$$\frac{dg_3}{dt}(0)x_0 = \frac{dp_3}{dt}(0) = \hat{V}_0,$$
$$\frac{dg_3}{dt}(1)x_0 = \frac{dp_3}{dt}(1) = \hat{V}_1.$$

From Theorem 2.5, applied to our generalized polynomial $g_3(t)$ in $\mathrm{SO}(m+1)$, we have

$$\frac{dg_3}{dt}(0) = 3V_0^1, \quad \frac{dg_3}{dt}(1) = 3V_2^1 g_3.$$

Thus,

$$\frac{dg_3}{dt}(0)x_0 = 3V_0^1 x_0 = \hat{V}_0,$$

$$\frac{dg_3}{dt}(1)x_0 = 3V_2^1 g_3 x_0 = 3V_2^1 x_3 = \hat{V}_1.$$

Hence, we need to determine solutions V_0^1, V_2^1 of the system of equations

$$V_0^1 x_0 = 1/3\hat{V}_0, \quad V_2^1 x_3 = 1/3\hat{V}_1. \tag{24}$$

We note here that V_0^1 and V_2^1 are elements in the Lie algebra of $\mathrm{SO}(m+1)$, and are therefore skew-symmetric matrices of order $(m+1)$. Since $\hat{V}_0^T x_0 = \hat{V}_1^T x_3 = 0$, we note also that Eqs. (24) are consistent, i.e.,

$$x_0^T V_0^1 x_0 = 1/3 x_0^T \hat{V}_0 = 0,$$
$$x_3^T V_2^1 x_3 = 1/3 x_3^T \hat{V}_1 = 0.$$

However, Eqs. (24) do not determine V_0^1 and V_2^1 uniquely. But, by Theorem 4.2, x_0 is joined to x_1 by a geodesic in S^m, with tangent vector $1/3\hat{V}_0$ (at x_0). Thus we know that V_0^1 must be an infinitesimal rotation acting on the plane $\Pi_1 = \text{span}\{x_0, \hat{V}_0\}$ and keeping invariant the hyperplane $\Pi_1^\perp$. Similarly, V_2^1 must be an infinitesimal rotation acting on the plane $\Pi_2 = \text{span}\{x_3, \hat{V}_1\}$ and keeping invariant the hyperplane $\Pi_2^\perp$. Thus,

$$V_0^1 = \theta_0 S_{x_0,\hat{V}_0}, \quad V_2^1 = \theta_1 S_{x_3,\hat{V}_1}, \tag{25}$$

where $S_{x_0,\hat{V}_0} = -P^T_{x_0,\hat{V}_0} A_{12} P_{x_0,\hat{V}_0}$ and $S_{x_3,\hat{V}_1} = -P^T_{x_3,\hat{V}_1} A_{12} P_{x_3,\hat{V}_1}$. (Note that when $m = 2$, $S_{a,b} = S_{a\times b}$.)

Finally, Eqs. (24) become

$$\theta_0 S_{x_0,\hat{V}_0} x_0 = 1/3\hat{V}_0, \quad \theta_1 S_{x_3,\hat{V}_1} x_3 = 1/3\hat{V}_1. \tag{26}$$

These equations define θ_0, θ_1, $0 \le \theta_0$, $\theta_1 \le \pi$, uniquely and hence V_0^1 and V_2^1.

Having obtained V_0^1 and V_2^1, we can now compute x_1 and x_2 from

$$x_1 = \exp(V_1^0)x_0 \quad \text{and} \quad x_2 = \exp(-V_2^1)x_3.$$

Note that, according to (25), the computation of $\exp(V_0^1)$ and $\exp(-V_2^1)$ only requires the trivial computation of $\exp(A_{12})$ and matrix multiplications. Thus, the control points g_1, g_2, and g_3 can be obtained, as before, by $g_i g_{i-1}^{-1} = \exp(\theta_i^1 S_{x_{i-1},x_i})$, $\theta_i^1 = \cos^{-1}(x_{i-1}^T x_i)$, $1 \le i \le 3$. This completes the problem of generating generalized cubics in S^m, with the boundary data of Case 1 (S^m). We now introduce the boundary data of Case 2.

Case 2 (S^m). The boundary data are x_0, x_3, $\dfrac{dp_3}{dt}(0)$, $\dfrac{D^2p_3}{dt}(0)$. Here x_0 and x_3 are unit vectors in $\mathbb{R}^{m+1}$, $\dfrac{dp_3}{dt}(0) = \hat{V}_0$ and $\dfrac{D^2p_3}{dt^2}(0) = \hat{W}_0$ are vectors in $\mathbb{R}^{m+1}$ tangent to S^m at x_0, that is, $\hat{V}_0^T x_0 = \hat{W}_0^T x_0 = 0$.

To make use of Theorem 4.1, we must once again identify the points g_1, g_2, g_3 in $\text{SO}(m+1)$, and the points x_1 and x_2 in S^m, such that

$$g_1 x_0 = x_1, \quad g_2 x_0 = x_2, \quad g_3 x_0 = x_3,$$
$$\frac{dg_3}{dt}(0)x_0 = \frac{dp_3}{dt}(0) = \hat{V}_0,$$
$$\frac{D^2_{S^m}}{dt^2}(g_3(0)x_0) = \frac{D^2_{S^m}p_3}{dt^2}(0) = \hat{W}_0.$$

(Note that here we use the covariant derivative on S^m.)

From Theorems 2.5 and 2.9 applied to our generalized polynomial in $SO(m+1)$, $g_3(t)$, we have

$$\frac{dg_3}{dt}(0) = 3V_0^1, \quad \frac{D^2_{SO(3)}g_3}{dt^2}(0) = 6\gamma_0(V_1^1 - V_0^1),$$

where $\gamma_0^{-1} = \int_0^1 \exp(u \operatorname{ad} V_0^1)\, du$. (Note that now we use the covariant derivative on $SO(m+1)$.)

We proceed to calculate V_0^1 as before from (24) and (25), namely from the equations

$$V_0^1 x_0 = 1/3 \hat{V}_0, \quad V_0^1 = \theta_0 S_{x_0, \hat{V}_0}.$$

Now, having V_0^1, we compute $x_1 = \exp(V_0^1)x_0$ and $g_1 = \exp(\theta_1^1 S_{x_0,x_1})$, where $\theta_1^1 = \cos^{-1}(x_0^T x_1)$.

Now we need to calculate V_1^1. We know that $\frac{d}{dt}g_3(t) = \Omega(t)g_3(t)$ where $\Omega(t) \in so(m+1)$ and that $\frac{D^2_{SO(m+1)}g_3}{dt^2}(t) = \dot{\Omega}(t)g_3(t)$, where $\dot{\Omega}(t) \in so(m+1)$. Thus,

$$\frac{D^2_{SO(m+1)}g_3}{dt^2}(0) = \dot{\Omega}(0) = 6\Upsilon_0(V_1^1 - V_0^1),$$

or equivalently

$$V_1^1 = V_0^1 + \frac{1}{6}\Upsilon_0^{-1}\dot{\Omega}(0). \tag{27}$$

However

$$\frac{d}{dt}p_3(t) = \frac{d}{dt}g_3(t)x_0 = \Omega(t)g_3(t)x_0$$

and, therefore,

$$\begin{aligned}\frac{D^2_{S^m}p_3}{dt^2}(t) &= \ddot{p}_3(t) - p_3^T(t)\ddot{p}_3(t)p_3(t) = \dot{\Omega}(t)g_3(t)x_0 + \\ &+ \Omega^2(t)g_3(t)x_0 - p_3^T(t)(\dot{\Omega}(t)g_3(t)x_0 + \Omega^2(t)g_3(t)x_0)p_3(t) = \\ &= \dot{\Omega}(t)p_3(t) + \Omega^2(t)p_3(t) - p_3^T(t)\Omega^2(t)p_3(t)p_3(t).\end{aligned}$$

Hence,

$$\frac{D^2_{S^m}p_3}{dt^2}(0) = \dot{\Omega}(0)x_0 + \Omega^2(0)x_0 - x_0^T\Omega^2(0)x_0x_0 = \hat{W}_0. \tag{28}$$

Now we show that $\Omega^2(0)x_0 - x_0^T\Omega^2(0)x_0x_0 = 0$.

First notice that $p(t) = e^{tV_0^1}x_0$ is a geodesic arc on S^m, joining x_0 (at $t=0$) and x_1 (at $t=1$). Consequently,

$$\frac{D^2_{S^m}p}{dt^2}(t) = \ddot{p}(t) - p^T(t)\ddot{p}(t)p(t) = 0 \ \ \forall t.$$

In particular, $\frac{D^2_{S^m} p}{dt^2}(0) = \ddot{p}(0) - p^T(0)\ddot{p}(0)p(0) = 0$ and the result above follows by taking into consideration that $V_0^1 = 1/3\Omega(0)$. Thus, we obtain from (28) that

$$\dot{\Omega}(0)x_0 = \hat{W}_0. \tag{29}$$

Putting together Eqs. (27) and (29), it follows that

$$V_1^1 x_0 = V_0^1 x_0 + \frac{1}{6}\Upsilon_0^{-1}\hat{W}_0. \tag{30}$$

Although the right-hand side of this equation is already known, V_1^1 cannot be determined uniquely from this. However, since $\exp(V_1^1)g_1 = g_2$ and $g_i x_0 = x_i$, $i = 0, \cdots, 3$, one has $\exp(V_1^1)x_1 = g_2 g_1^{-1} x_1 = g_2 x_1 = x_2$. Thus, $\exp(V_1^1)$ is a rotation on the plane spanned by x_1 and x_2, say,

$$V_1^1 = \alpha S_{x_1, x_2} \quad \text{for some} \quad \alpha \in \mathbb{R}. \tag{31}$$

Now, as before, we can solve uniquely for V_1^1, from (30) and (31) as long as x_0 and x_1 are in general position on the sphere. In such cases we can solve for g_2 and x_2 as above. Finally g_3 is calculated as before $g_3 g_2^{-1} = \exp(\theta_3^1 S_{x_2, x_3})$, $\theta_3^1 = \cos^{-1}(x_2^T x_3)$.

This completes the problem of generating generalized cubics in S^{m+1}, with the boundary data of Case 2.

4.1. Example. We have used the formula (18) recursively to implement the De Casteljau algorithm for a cubic polynomial on the sphere S^3. In order to visualize the end effect, we use the well known fact that each unit quaternion q induces a rotation matrix R_q in $\mathbb{R}^3$ in the following way. If $q = \cos\alpha + \hat{e}\sin\alpha$, $\hat{e} \in \mathbb{R}^3$, $\|\hat{e}\| = 1$, is the trigonometric representation of the quaternion q, then $R_q = \exp(2\alpha S_{\hat{e}})$, where $S_{\hat{e}}$ is the skew-symmetric matrix defined by $S_{\hat{e}} y = \hat{e} \times y \ \forall y \in \mathbb{R}^3$. The two figures below show the position of a rigid body, where position is defined by cubic polynomials, one on SO(3) and the other on S^3. The similarity indicates that there are no significant differences in the final effect, although one of the algorithms (on S^3) is substantially faster, due to the fact that it avoids computing logarithms and exponentials of 3×3 matrices. Figure (a) was obtained using the algorithm for SO(3) described in detail in our paper [12], with

initial data

$$g_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$g_1 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$g_2 = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & -1 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \end{bmatrix},$$

$$g_3 = \begin{bmatrix} 0 & -1 & 0 \\ \frac{1}{2} & 0 & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & 0 & \frac{1}{2} \end{bmatrix}.$$

Figure (b) was obtained using the algorithm for S^3 described above, having as initial data the unit quaternions q_0, q_1, q_2, and q_3, associated with g_0, g_1, g_2 and g_3 respectively.

5. Comparison with the variational approach

Interpolating curves satisfying arbitrary boundary conditions on a Riemannian manifold can also be obtained using a variational approach. While in the Euclidean case both methods produce exactly the same curves, for general Riemannian manifolds this situation is highly unlikely. The variational approach on a Riemannian manifold was first introduced in the work of Noakes et. al [29] and more recently has received considerable attention by Camarinha, Crouch, and Silva Leite in a series of papers ([5], [6], [10], and [9]). In this context, cubic polynomials, in particular, are the solutions of the Euler–Lagrange equations for the following functional:

$$\int_0^1 \left\langle \frac{D^2x(t)}{dt^2}, \frac{D^2x(t)}{dt^2} \right\rangle dt.$$

We refer to the value of this integral for a particular curve as the average acceleration. At each point on the sphere S^m, $\frac{D^2x(t)}{dt^2}$ is simply

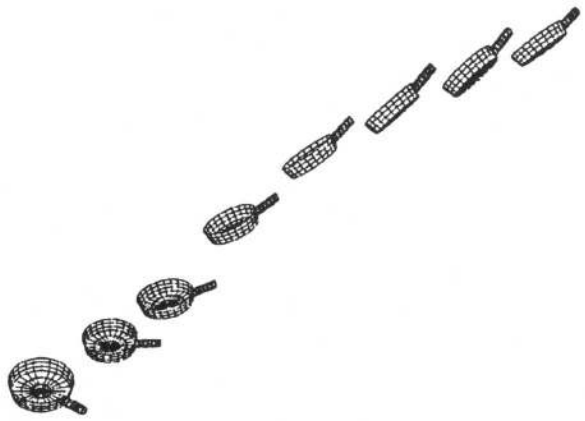


Fig. 3 (a). Using the algorithm for SO(3)

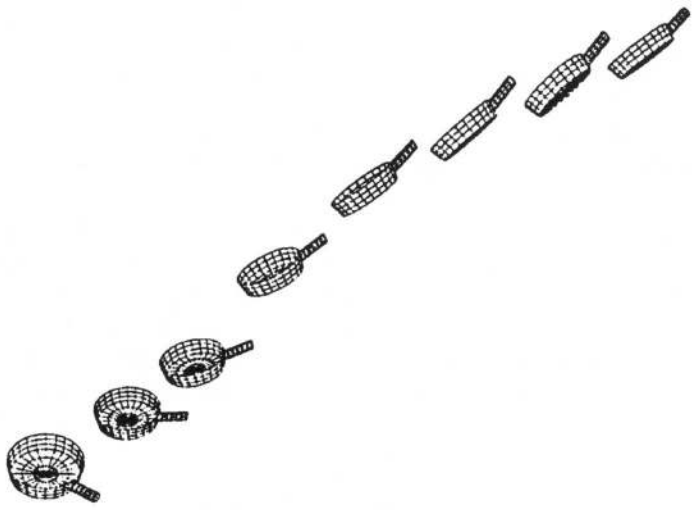


Fig. 3 (b). Using the described algorithm on S^3

the projection of $\ddot{x}(t)$ onto the tangent space to S^m at that point. Thus, $\frac{D^2x}{dt^2} = \ddot{x} - \langle x, \ddot{x}\rangle x$ on S^m, and the average acceleration reduces to

$$\int_0^1 \left(\langle \ddot{x}, \ddot{x}\rangle - \langle x, \ddot{x}\rangle^2\right) dt. \tag{32}$$

With this simplified formula one can easily compute the energy along any cubic polynomial produced by the De Casteljau algorithm on the spheres. This will be used in the examples below for the 2-dimensional sphere, as we further attempt to understand generalized polynomial curves.

In Fig. 4.(a) and (b), we plot two cubic polynomials on S^2, both satisfying the same boundary conditions

$$\begin{aligned} p(0) &= [-0.8660, 0.5, 0], & p(1) &= [-1, 0, 0], \\ \dot{p}(0) &= [1.897, 3.286, 4.382], & \dot{p}(1) &= [0, -1.814, -2.565]. \end{aligned}$$

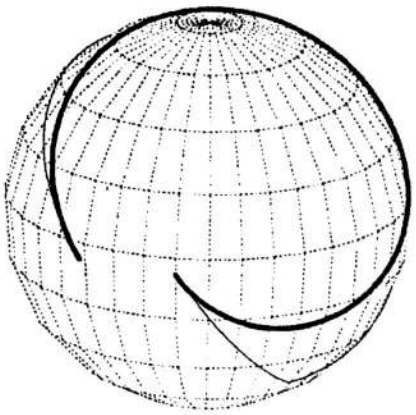
None of the four points employed in the implementation of the De Casteljau algorithm are antipodal. The two plots correspond to two different choices of the geodesic arc joining the intermediate points x_1 and x_2. In Plot (b) the geodesic is length minimizing while in Plot (a) is not. The average acceleration of the curves (a) and (b), calculated using the formula (32), is (≈ 250) and (≈ 1150) respectively. In Plot (a) the length of the curve is substantially larger than the length of the curve in Plot (b). The actual length of the curves is 5.2 in Plot (a) and 2.2 in Plot (b).

This example demonstrates that choosing what might seem to be the natural choice of the length minimizing geodesic joining the control points in the De Casteljau algorithm (Plot (b)) results in a curve which is less aesthetic (as measured by the much greater average acceleration), while at the same time yields an interpolating curve of smaller length.

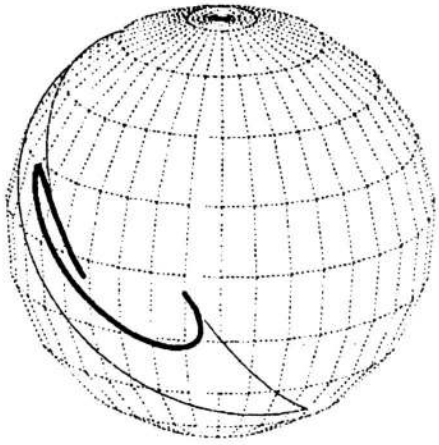
Figure 5 illustrates the case when the prescribed data give rise to control points in the De Casteljau algorithm that are antipodal. As a consequence, the De Casteljau algorithm produces infinitely many cubic polynomials satisfying the same boundary conditions. Here we compare the energy along several cubic polynomials and the numerical calculation shows that, unlike the length of the curves, the average acceleration is more or less equal (≈ 340).

These examples demonstrate that the issues of constructing aesthetic interpolating curves, described in detail by Farin [19] for the Euclidean case, can become even more complex in the non-Euclidean case.

For general manifolds with Riemannian metric $\langle \cdot, \cdot \rangle$, one way of defining polynomial curves of degree $n = 2k - 1$ is through the Euler–Lagrange



(a) length ≈ 5.2
average acceleration ≈ 250



(b) length ≈ 2.2
average acceleration ≈ 1150

Fig. 4. Two curves for the same boundary value problem

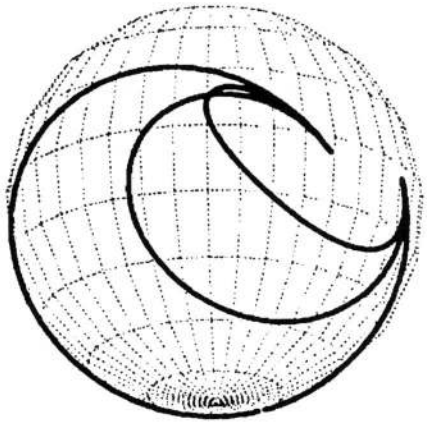


Fig. 5. The antipodal case

equation associated with the functional

$$\int_0^1 \left\langle \frac{D^k x}{dt^k}, \frac{D^k x}{dt^k} \right\rangle dt$$

(see Camarinha, Silva Leite, and Crouch [4]). To compare our results here on the generalized De Casteljau algorithm with the variational approach, one needs to compute higher order derivatives. For compact Lie groups, in Crouch and Silva Leite [9], we have derived all the ingredients to compute higher order covariant derivatives along polynomial curves. However, the process of obtaining closed forms for these derivatives soon becomes extremely hard and involves many tedious calculations. In the case where G is an abelian Lie group, the results of Sec. 3 simplify substantially and it is possible to show that the polynomial curves obtained via the De Casteljau algorithm are exactly those produced via the variational approach.

Theorem 5.1. *If G is a compact, connected, and abelian Lie group, the polynomial curves of degree $n = 2k - 1$ generated by the De Casteljau algorithm are also solutions of the Euler–Lagrange equation associated with the functional*

$$\mathcal{J}(x) = \int_0^1 \left\langle \frac{D^k x}{dt^k}, \frac{D^k x}{dt^k} \right\rangle dt.$$

Proof. First of all we note that, according to Camarinha, Silva Leite, and Crouch [4], the Euler–Lagrange equation for this variational problem on a

connected, compact, and abelian Lie group reduces to $\frac{d^n V_t}{dt^n} = 0$, where $V_t = \dot{x}(t)x^{-1}(t)$.

On the other hand, for the abelian case it follows from Lemma 2.4 that if $A(t)$ is a curve of class $\mathcal{C}^1$ in $\mathcal{L}$, then $\frac{d}{dt}\exp(tA(t)) = (A(t)+t\dot{A}(t))\exp A(t)$. Using this after taking derivatives on both sides of the expression (6), for the polynomial curve of degree n obtained via the De Casteljau algorithm, we easily obtain

$$\begin{aligned}\dot{p}_n(t) = \{&V_0^n(t) + V_0^{n-1}(t) + \cdots + V_0^2(t) + V_0^1 + \\ &+ t(\dot{V}_0^n(t) + \dot{V}_0^{n-1}(t) + \cdots + \dot{V}_0^2(t))\}p_n(t). \end{aligned} \tag{33}$$

Therefore, setting $V(t) = V_0^n(t) + V_0^{n-1}(t) + \cdots + V_0^2(t) + V_0^1$, to complete the proof it is enough to show that

$$\frac{d^n}{dt^n}\left\{V(t) + t(\dot{V}(t)\right\} = (n+1)\frac{d^n}{dt^n}V(t) + t\frac{d^{n+1}}{dt^{n+1}}V(t) = 0.$$

To prove this, first note that, in the present situation, formula (5) reduces to

$$V_k^j(t) = tV_{k+1}^{j-1}(t) + (1-t)V_k^{j-1}(t)$$

and using this new formula it is easy to prove by induction that

$$V_k^j(t) = \sum_{i=0}^{j-1}\binom{j-1}{i}t^i(1-t)^{j-1-i}V_{i+k}^1 \quad \forall j,k. \tag{34}$$

This can be written in terms of the Bernstein polynomials defined by $B_i^j(t) = \binom{cj}{i}t^i(1-t)^{j-i}$. (See Farin [19] for detail). These polynomials have degree j and $\frac{d^{j+1}}{dt^{j+1}}B_i^j(t) = 0\ \forall i = 0,\cdots,j$. Therefore, according to (34), we can write

$$V_0^j(t) = \sum_{i=0}^{j-1}B_i^{j-1}(t)V_i^1$$

and, consequently, $\frac{d^n}{dt^n}V_0^j(t) = \sum_{i=0}^{j-1}\frac{d^n}{dt^n}(B_i^{j-1}(t))V_i^1 = 0\ \forall j = 0,\cdots,n,$ which, in turn, implies that $\frac{d^n}{dt^n}V(t) = 0$. This completes the proof of the theorem. □

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) **26** (1992), No. 2, 313–320.
2. P. Bézier, The mathematical basis of the UNISURF CAD System. *Butterworths, London*, 1986.
3. W. M. Boothby, An introduction to differential manifolds and Riemannian geometry. *Academic Press, Pure and Appl. Math. Series of Monographs and Textbooks*, Vol. 63, *New York*, 1975.
4. M. Camarinha, F. Silva Leite, and P. Crouch, Splines of class C^k on non-Euclidean spaces. *J. Math. Control and Inform.* **12** (1995), 399–410.
5. ———, Second order optimality conditions for a higher order variational problem on a Riemannian manifold. In: *Proc. 35th IEEE CDC, Kobe-Japan*, 11–13 December (1996), Vol. II, 1636–1641.
6. ———, Sufficient conditions for an optimization problem on a Riemannian manifold. In: *Proc. CONTROLO-96, Porto-Portugal*, 11–13 September (1996), Vol. I, 127–131.
7. J. Cardoso and F. Silva Leite, Logarithms and exponentials for the Lie group of P-orthogonal matrices. *Submitted in* 1998.
8. Chao-Chi Chen, Interpolation of orientation matrices using sphere splines in computer animation. *Master of Science Thesis*, Arizona State University (1990).
9. P. Crouch and F. Silva Leite, The dynamic interpolation problem on riemannian manifolds, Lie groups and symmetric spaces, *J. Dynam. Control Syst.* **1** (1995), No. 2, 177–202.
10. ———, Geometry and the dynamic interpolation problem, In: *Proc. Am. Control Confer. Boston*, 26–29 July (1991).
11. ———, Closed forms for the exponential mapping on matrix Lie groups based on Putzer's method. To appear in: *J. Math. Physics.*
12. P. Crouch, G. Kun, and F. Silva Leite, De Casteljau algorithm for cubic polynomials on the rotation group. In: *Proc. CONTROLO-96*, 11-13 September (1996), Vol. II, 547–552.
13. ———, Geometric splines. To appear in: *Proc. 14th IFAC World Congress*, 5–9 July (1999), Beijing, P. R. China.
14. P. Crouch and J. Jackson, A non-holonomic dynamic interpolation problem. In: *Proc. Confer. Anal. Control Dynam. Syst. Lion-France*, 1990, *Birkhäuser, series Progress in Systems and Control*, 1991, 156–166.
15. ———, Dynamic interpolation and application to flight control. To appear in: *Proc. IEEE Decision and Control Confer.* Honolulu, Hawaii (1990).

16. ———, Dynamic interpolation and application to flight control To appear in: *J. Guidance, Control and Dynam.*
17. J. Jackson, Dynamic interpolation and application to flight control. *Ph D thesis, Arizona State University*, 1990.
18. P. De Casteljau, Outillages Méthodes Calcul. *Technical Report, A. Citroen, Paris*, 1959.
19. G. Farin, Curves and surfaces for CAGD. *Academic Press*, Third Edition, 1993.
20. Q. J. Ge and B. Ravani, Computer aided geometric design of motion interpolants. In: *Proc. ASME Design Automation Conf. Miami, Fl, September*, 1991, 33–41.
21. ———, Computational geometry and mechanical design synthesis, In: *Proc. 13th IMACS World Congress on Computation and Applied Mathematics, Dublin, Ireland*, 1991, 1013–1015.
22. ———, Geometric construction of Bezier motions. *ASME J. Mechan. Design* **116** (1994), 749–755.
23. R. Hirschorn, Curves in homogeneous spaces. *Can. J. Math.* **29** (1977), No. 1, 77–83.
24. B. Jutler, Visualization of moving objects using dual quaternion curves. *Computers and Graphics* **18** (1994), No. 3, 315–326.
25. M. J. Kim, M. S. Kim, and S. Y. Shin, A general construction scheme for unit quaternion curves with simple high order derivatives. In: *Computer Graphics Proc., Annual Conf. Series, (SIGRAPH* 95), *Los Angeles, CA.*, 1995, 369–376.
26. J. Milnor, Morse theory. *Ann. Math. Stud.* **51** (1969), *Princeton University Press.*
27. G. Nielson and R. Heiland, Animated rotations using quaternions and splines on a 4D sphere. In: *Programmirovanie* (Russia) *Springer Verlag*, English edition. *Programming and Computer Software, Plenum Pub.* NY 17–27 (1992).
28. G. Nielson, Smooth interpolation of orientations. *Models and Techniques in Computer Animation* (N. Magnenat Thalmann and D. Thalmann Eds.), *Springer Verlag, Tokyo*, 1993, 75–93.
29. L. Noakes, G. Heinzinger and B. Paden, Cubic splines on curved spaces. *IMA J. Math. Control Inform.* **6** (1989), 465–473.
30. K. Nomizu, Foundations of differential geometry. In: *Interscience Tracts in Pure and Applied Mathematics, John Willey and Sons*, Vol. I, II (1969).
31. F. Park and B. Ravani, Bézier curves on Riemannian manifolds and Lie groups with kinematic applications. *ASME J. Mechan. Design* **117** (1995), 36–40.

32. D. Sattinger and O. Weaver, Lie groups and algebras with applications to physics, geometry and mechanics. *Appl. Math. Sci.* **61**, Springer Verlag, 1986.
33. K. Shoemake, Animating rotations with quaternion curves. *ACM SIGRAPH* 85 **19** 245–254 (1985).

(Received 06.04.1999)

Authors' addresses:
P. Crouch
Center for Systems Science and Engineering,
Arizona State University,
Tempe, AZ 85287–USA
E-mail: peter.crouch@asu.edu

G. Kun
Lehrstuhl C für Mathematik,
RWTH Aachen,
D–52056 Aachen, Germany
E-mail: kun@math2.rwth-aachen.de

F. Silva Leite
Departamento de Matemática,
Universidade de Coimbra,
3000 Coimbra–Portugal
E-mail: fleite@mat.uc.pt