Cubic Splines in the Grassmann Manifold

Fátima Pina[1,2] and Fátima Silva Leite[1,2(✉)]

[1] Department of Mathematics, University of Coimbra, 3001-501 Coimbra, Portugal
fpina@mat.uc.pt
[2] Institute of Systems and Robotics, DEEC-UC, 3030-290 Coimbra, Portugal
fleite@mat.uc.pt

Abstract. We present a detailed implementation of the De Casteljau algorithm to generate cubic splines that solve certain interpolation problems in the Grassmann manifold.

Keywords: Cubic splines · De Casteljau algorithm · Geodesics · Grassmann manifolds · Interpolating curves

1 Introduction

Interpolating nonlinear data arises in many different areas, ranging from robotics and computer vision to industrial and medical applications (see, e.g., [2]). A particular illustrative example is the classical problem of reconstructing a scene from several snapshots taken at different time instants. This can be formulated as an interpolation problem in the Grassmann manifold. This manifold, consisting of all k-dimensional subspaces in $\mathbb{R}^n$, plays an important role in many computer vision applications where subspace methods are used to represent an image set by a linear subspace that is spanned by all the images in the set.

In Euclidean spaces, cubic splines, which are $\mathcal{C}^2$-smooth curves obtained by piecing together cubic polynomials, are particularly important since they minimise the average acceleration. A well-known recursive procedure to generate interpolating polynomial curves in Euclidean spaces is the classical De Casteljau algorithm [4]. Generalizations of such curves to non-Euclidean manifolds is particularly useful in many engineering applications and were first introduced in [8]. In the present paper we implement the generalised De Casteljau algorithm for the Grassmann manifold, following closely the work in [3] concerning the reinterpretation of the De Casteljau algorithm for connected and compact Lie groups.

The main feature of the algorithm is based on recursive geodesic interpolation in order to find a polynomial curve that solves a 2-boundary value problem. These boundary conditions might be of Hermite type, i.e., consisting of initial and final points together with initial and final velocities, or consists of initial and final points, initial velocity and initial intrinsic acceleration. While the first conditions, together with other interpolation requirements are more natural in

J. A. Gonçalves et al. (Eds.): CONTROLO 2020, LNEE 695, pp. 243–252, 2021.
https://doi.org/10.1007/978-3-030-58653-9_23

applications, they pose difficulties that do not arise if the second type of boundary conditions is prescribed. After presenting some preliminary results in Sect. 2 and the geometry of the Grassmann manifold in Sect. 3, we dedicate the main section to the implementation of the De Casteljau algorithm for that important manifold. At the end of this section we derive relations between the two types of boundary conditions and explain how to generate a geometric cubic spline from cubic polynomials obtained by using the proposed algorithm.

2 Notations and Auxiliary Results

In the sequel, $\mathfrak{gl}(n)$ denotes the Lie algebra of $n \times n$ real matrices, equipped with the Lie bracket defined by the commutator, i.e., $[A, B] := AB - BA$. The adjoint operator in $\mathfrak{gl}(n)$ is defined by $\mathrm{ad}_A B := [A, B]$. The vector space of $n \times n$ symmetric matrices is denoted by $\mathfrak{s}(n)$ and the Lie subalgebra of $\mathfrak{gl}(n)$, consisting of the skewsymmetric $n \times n$ matrices is denoted by $\mathfrak{so}(n)$. The rotation group $SO(n)$, having $\mathfrak{so}(n)$ as its Lie algebra, will also play an important role here.

Given $A \in \mathfrak{gl}(n)$, the *matrix exponential of* A is defined by $\mathrm{e}^A = \sum_{k=0}^{+\infty} \frac{A^k}{k!}$. For $X \in \mathfrak{gl}(n)$ not having eigenvalues in the closed negative real line, i.e., $\sigma(X) \cap \mathbb{R}_0^- = \varnothing$, there exists a unique real logarithm of X whose spectrum lies in the infinite horizontal strip $\{z \in \mathbb{C} : -\pi < Im(z) < \pi\}$. This logarithm is usually called *principal logarithm* of X and is the only logarithm that we consider here. When $X \in SO(n)$, $\log X \in \mathfrak{so}(n)$. When $\|X - I\| < 1$, $\log X$ is uniquely defined by the convergent power series: $\log X = \sum_{k=1}^{+\infty} (-1)^{k+1} \frac{(X-I)^k}{k}$. Properties of these matrix functions can be found in [6] and [5], but we emphasise the formula $\mathrm{e}^A B \, \mathrm{e}^{-A} = \mathrm{e}^{\mathrm{ad}_A}(B) = B + [A, B] + \frac{1}{2!}[A, [A, B]] + \cdots$, which plays an important role here. In the sequel, we also assume the following notations:

$$f(z) = \tfrac{\mathrm{e}^z - 1}{z} \text{ denotes the sum of the series } \textstyle\sum_{k=0}^{+\infty} \frac{z^k}{(k+1)!}; \tag{1}$$

$$g(z) = \tfrac{\log z}{z-1} \text{ denotes the sum of the series } \textstyle\sum_{k=0}^{+\infty} (-1)^k \frac{(z-1)^k}{k+1}, \text{ if } |z-1| < 1. \tag{2}$$

Note that $f(z)g(e^z) = 1$. The next result can be found in [10].

Lemma 1. *Let $t \longmapsto A(t)$ be a differentiable matrix valued function. Then,*

$$\frac{\mathrm{d}}{\mathrm{d}\, t} \mathrm{e}^{A(t)} = \Delta^L_{A(t)}(t)\, \mathrm{e}^{A(t)}, \qquad \textit{where} \tag{3}$$

$$\Delta^L_{A(t)}(t) = \int_0^1 \mathrm{e}^{u \, \mathrm{ad}_{A(t)}}(\dot{A}(t)) du = f\left(\mathrm{ad}_{A(t)}\right)\left(\dot{A}(t)\right). \tag{4}$$

The following lemma contains important properties of the operator defined in (4). Proof details can be found in Sect. 2 of [9].

Lemma 2. *Let $t \longmapsto A(t)$ be a differentiable matrix valued function. Then*

$$\begin{gathered} \mathrm{e}^{A(t)}\, \Delta^L_{-A(t)}(t)\, \mathrm{e}^{-A(t)} = -\Delta^L_{A(t)}(t); \\ \Delta^L_{tA(t)}(t)\Big|_{t=0} = A(0) \quad \textit{and} \quad \Delta^L_{(t-1)A(t)}(t)\Big|_{t=1} = A(1); \\ \frac{\mathrm{d}}{\mathrm{d}\,t}\Big|_{t=0} \Delta^L_{tA(t)}(t) = 2\dot{A}(0) \quad \textit{and} \quad \frac{\mathrm{d}}{\mathrm{d}\,t}\Big|_{t=1} \Delta^L_{(t-1)A(t)}(t) = 2\dot{A}(1). \end{gathered} \tag{5}$$

3 The Geometry of the Grassmann Manifold

The Grassmann manifold is the set of all k-dimensional subspaces in $\mathbb{R}^n$. Here, similarly to [7] and [1], we use the following matrix representation of the Grassmann manifold

$$G_{k,n} := \big\{P \in \mathfrak{s}(n) \,:\, P^2 = P \text{ and } \operatorname{rank}(P) = k\big\}, \tag{6}$$

which is compact and connected, of real dimension $k(n-k)$, $0 < k < n$.

For an arbitrary point $P \in G_{k,n}$, define the following sets of matrices

$$\mathfrak{gl}_P(n) := \{A \in \mathfrak{gl}(n) \,:\, A = PA + AP\}; \quad \mathfrak{so}_P(n) := \mathfrak{so}(n) \cap \mathfrak{gl}_P(n). \tag{7}$$

The tangent space at a point $P \in G_{k,n}$ can be defined by

$$T_P G_{k,n} = \big\{[P, \Omega] \,:\, \Omega \in \mathfrak{so}_P(n)\big\} = \big\{\mathrm{ad}_P^2(S) \,:\, S \in \mathfrak{s}(n)\big\}. \tag{8}$$

We consider the Grassmann manifold equipped with the Riemannian metric, induced by the Euclidean inner product on each tangent space, given by

$$\langle [P, \Omega_1], [P, \Omega_2] \rangle = -\mathrm{tr}\big(\Omega_1 \Omega_2\big). \tag{9}$$

Using the last description of the tangent space at P, the normal space at P, with respect to the Riemannian metric (9), can be defined by

$$(T_P G_{k,n})^{\perp} = \big\{S - \mathrm{ad}_P^2(S) \,:\, S \in \mathfrak{s}(n)\big\}. \tag{10}$$

The proof of the next two lemmas can be found in [9].

Lemma 3. *Let $P \in G_{k,n}$ and $\Omega \in \mathfrak{so}_P(n)$. Then, $[\Omega, [\Omega, P]] \in (T_P G_{k,n})^{\perp}$.*

Lemma 4. *Let $P \in G_{k,n}$, $A, B \in \mathfrak{gl}_P(n)$, and $t \in \mathbb{R}$. Then,*

$$[A, P] = [B, P] \Longleftrightarrow A = B; \tag{11}$$

$$\mathrm{e}^{2tA}(I - 2P) = \mathrm{e}^{\mathrm{ad}_{tA}}(I - 2P). \tag{12}$$

Now, we present some results about geodesics in the Grassmann manifold with respect to the Riemannian metric in (9).

Lemma 5 ([1]). *The unique geodesic $t \mapsto \gamma(t)$ in $G_{k,n}$, satisfying the initial conditions $\gamma(0) = P$ and $\dot{\gamma}(0) = [\Omega, P]$, where $\Omega \in \mathfrak{so}_P(n)$, is given by*

$$\gamma(t) = \mathrm{e}^{t\Omega}\, P\, \mathrm{e}^{-t\Omega}\,. \tag{13}$$

The next result gives an explicit formula for the minimising geodesic arc connecting two points in $G_{k,n}$.

Proposition 1 ([1]). *Let $P, Q \in G_{k,n}$ be such that the matrix $(I-2Q)(I-2P)$ has no negative real eigenvalues. Then, the minimising geodesic arc in $G_{k,n}$ that joins P (at $t=0$) to Q (at $t=1$), is parameterised explicitly by (13), with $\Omega = \frac{1}{2}\log((I-2Q)(I-2P))$.*

4 De Casteljau Algorithm in the Grassmann Manifold

4.1 An Interpolation Problem in $G_{k,n}$

Problem 1. Given $\ell+1$ distinct points $p_i \in G_{k,n}$, with $i = 0, 1, \ldots, \ell$, a discrete sequence of $\ell+1$ fixed times, $t_0 < t_1 < \cdots < t_{\ell-1} < t_\ell$, and vectors ξ_0, η_0 tangent to $G_{k,n}$ at p_0, and ξ_ℓ tangent to $G_{k,n}$ at p_ℓ, solve the following problem:

Find a $\mathcal{C}^2$-smooth curve $\gamma : [t_0, t_\ell] \to G_{k,n}$, satisfying interpolation conditions

$$\gamma(t_i) = p_i, \qquad 1 \leq i \leq \ell - 1, \tag{14}$$

and boundary conditions (of Hermite type):

$$\begin{array}{ll} \gamma(t_0) = p_0, & \gamma(t_\ell) = p_\ell, \\ \dot{\gamma}(t_0) = \xi_0 \in T_{p_0}G_{k,n}, & \dot{\gamma}(t_\ell) = \xi_\ell \in T_{p_\ell}G_{k,n}, \end{array} \tag{15}$$

or, alternatively, boundary conditions:

$$\begin{array}{rl} \gamma(t_0) = p_0, & \gamma(t_\ell) = p_\ell, \\ \dot{\gamma}(t_0) = \xi_0 \in T_{p_0}G_{k,n}, & \dfrac{D\dot{\gamma}}{dt}(t_0) = \eta_0 \in T_{p_0}G_{k,n}, \end{array} \tag{16}$$

where $\dfrac{D\dot{\gamma}}{dt}$ stands for the covariant derivative of the velocity vector field $\dot{\gamma}$.

Without loss of generality, in the sequel we consider $t_0 = 0$ and $t_\ell = 1$, since the reparametrisation $(t \to s)$ defined by $s = t(t_\ell - t_0) + t_0$ maps $[0,1]$ to $[t_0, t_\ell]$.

The solution of this problem is a *geometric cubic spline*, obtained by piecing together geometric cubic polynomials generated by the generalised De Casteljau algorithm. For geodesically complete manifolds, this algorithm consists of three successive geodesic interpolation steps that first appeared in [8].

4.2 Implementation of the De Casteljau Algorithm in $G_{k,n}$

Although the Grassmann manifold is geodesically complete, we have seen that an explicit formula for the geodesic that joins two points may be unknown in some particular situations. So, in this case, the implementation of the algorithm is restricted to a convex open subset of the manifold where the expression to compute geodesic arcs is known. The generation of a geometric cubic polynomial

in $G_{k,n}$, starts from four given points x_0, x_1, x_2, x_3 in $G_{k,n}$. The superscripts in the operators Ω_i^j below are chosen according to the corresponding step.

Algorithm

Step 1 - Construct three geodesic arcs

$$\beta_1(t, x_i, x_{i+1}) = \mathrm{e}^{t\Omega_i^1}\, x_i\, \mathrm{e}^{-t\Omega_i^1} = \mathrm{e}^{t\ \mathrm{ad}_{\Omega_i^1}}\, x_i, \quad i = 0, 1, 2, \tag{17}$$

$$\Omega_i^1 := \frac{1}{2}\log((I - 2x_{i+1})(I - 2x_i)) \in \mathfrak{so}_{x_i}(n). \tag{18}$$

Step 2 - Construct two families of geodesic arcs

$$\beta_2(t, x_0, x_1, x_2) = \mathrm{e}^{t\Omega_0^2(t)}\, \beta_1(t, x_0, x_1)\, \mathrm{e}^{-t\Omega_0^2(t)} = \mathrm{e}^{t\ \mathrm{ad}_{\Omega_0^2(t)}}\, \beta_1(t, x_0, x_1), \tag{19}$$

$$\beta_2(t, x_1, x_2, x_3) = \mathrm{e}^{t\Omega_1^2(t)}\, \beta_1(t, x_1, x_2)\, \mathrm{e}^{-t\Omega_1^2(t)} = \mathrm{e}^{t\ \mathrm{ad}_{\Omega_1^2(t)}}\, \beta_1(t, x_1, x_2), \tag{20}$$

$$\Omega_0^2(t) := \frac{1}{2}\log((I - 2\beta_1(t, x_1, x_2))(I - 2\beta_1(t, x_0, x_1))) \in \mathfrak{so}_{\beta_1(t,x_0,x_1)}(n), \tag{21}$$

$$\Omega_1^2(t) := \frac{1}{2}\log((I - 2\beta_1(t, x_2, x_3))(I - 2\beta_1(t, x_1, x_2))) \in \mathfrak{so}_{\beta_1(t,x_1,x_2)}(n). \tag{22}$$

Step 3 - Construct a family of geodesic arcs

$$\beta_3(t, x_0, x_1, x_2, x_3) = \mathrm{e}^{t\Omega_0^3(t)}\, \beta_2(t, x_0, x_1, x_2)\, \mathrm{e}^{-t\Omega_0^3(t)} = \mathrm{e}^{t\ \mathrm{ad}_{\Omega_0^3(t)}}\, \beta_2(t, x_0, x_1, x_2), \tag{23}$$

$$\Omega_0^3(t) := \frac{1}{2}\log((I - 2\beta_2(t, x_1, x_2, x_3))(I - 2\beta_2(t, x_0, x_1, x_2))) \in \mathfrak{so}_{\beta_2(t,x_0,x_1,x_2)}(n). \tag{24}$$

As a result, and taking into consideration (17), (19), (20) and (23), we obtain the geometric cubic polynomial in the Grassmann manifold.

Definition 1. *The curve* $t \in [0, 1] \mapsto \beta_3(t) := \beta_3(t, x_0, x_1, x_2, x_3)$*, defined by*

$$\beta_3(t) = \mathrm{e}^{t\ \mathrm{ad}_{\Omega_0^3(t)}}\, \mathrm{e}^{t\ \mathrm{ad}_{\Omega_0^2(t)}}\, \mathrm{e}^{t\ \mathrm{ad}_{\Omega_0^1}}\, x_0, \tag{25}$$

with Ω_0^1*,* Ω_0^2 *and* Ω_0^3 *given by* (18)*,* (21)*, and* (24)*, is called a geometric cubic polynomial in the Grassmann manifold, associated to the points* x_i*,* $i = 0, 1, 2, 3$*.*

Remark 1. Notice that $\Omega_0^2(0) = \Omega_0^3(0) = \Omega_0^1$, $\Omega_1^2(0) = \Omega_1^1$, $\Omega_1^2(1) = \Omega_0^3(1) = \Omega_2^1$, $\Omega_0^2(1) = \Omega_1^1$, and that the curve β_3, just defined, joins the point x_0 (at $t = 0$) to x_3 (at $t = 1$). However, it doesn't pass through x_1 and x_2, which are called *control points* since they are responsible for the shape of the curve.

Lemma 6 ([9]). *Let* Ω_i^j *be defined by* (18)*,* (21)*,* (22) *and* (24)*. Then,*

$$\begin{aligned} \mathrm{e}^{2\Omega_0^2(t)} &= \mathrm{e}^{2t\Omega_1^1}\, \mathrm{e}^{2(1-t)\Omega_0^1}, \\ \mathrm{e}^{2\Omega_1^2(t)} &= \mathrm{e}^{2t\Omega_2^1}\, \mathrm{e}^{2(1-t)\Omega_1^1}, \\ \mathrm{e}^{2\Omega_0^3(t)} &= \mathrm{e}^{2t\Omega_1^2(t)}\, \mathrm{e}^{2(1-t)\Omega_0^2(t)}; \end{aligned} \tag{26}$$

$$\begin{aligned} \mathrm{e}^{(t-1)\,\mathrm{ad}_{\Omega_0^2(t)}}\, \mathrm{e}^{t\,\mathrm{ad}_{\Omega_1^1}} &= \mathrm{e}^{t\,\mathrm{ad}_{\Omega_0^2(t)}}\, \mathrm{e}^{(t-1)\,\mathrm{ad}_{\Omega_0^1}}, \\ \mathrm{e}^{(t-1)\,\mathrm{ad}_{\Omega_1^2(t)}}\, \mathrm{e}^{t\,\mathrm{ad}_{\Omega_2^1}} &= \mathrm{e}^{t\,\mathrm{ad}_{\Omega_1^2(t)}}\, \mathrm{e}^{(t-1)\,\mathrm{ad}_{\Omega_1^1}}, \\ \mathrm{e}^{(t-1)\,\mathrm{ad}_{\Omega_0^3(t)}}\, \mathrm{e}^{t\,\mathrm{ad}_{\Omega_1^2(t)}} &= \mathrm{e}^{t\,\mathrm{ad}_{\Omega_0^3(t)}}\, \mathrm{e}^{(t-1)\,\mathrm{ad}_{\Omega_0^2(t)}}. \end{aligned} \tag{27}$$

We are now in conditions to state the following result which contains an alternative way of defining the geometric cubic polynomial β_3 in $G_{k,n}$. This will be particularly useful in the computation of the derivatives of the cubic polynomial at the endpoint ($t = 1$).

Theorem 1. *Let $t \in [0,1] \mapsto \beta_3(t)$ be the geometric cubic polynomial in $G_{k,n}$ defined in Definition 1. Define another curve $t \in [0,1] \mapsto \gamma(t)$ in $G_{k,n}$, by*

$$\gamma(t) = \mathrm{e}^{(t-1)\ \mathrm{ad}_{\Omega_0^3(t)}}\, \mathrm{e}^{(t-1)\ \mathrm{ad}_{\Omega_1^2(t)}}\, \mathrm{e}^{(t-1)\ \mathrm{ad}_{\Omega_2^1}}\, x_3, \tag{28}$$

where Ω_0^3, Ω_1^2 and Ω_2^1 are as defined at the beginning of Subsect. 4.2. Then,

$$\beta_3(t) = \gamma(t), \quad t \in [0,1].$$

Proof. The proof results from simple computations, attending to Remark 1 and applying formulas (27) in Lemma 6. QED

Lemma 7 ([9])**.** *For $j = 2, 3$, let $i = 3 - j$. Then, the following holds.*

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\left(\mathrm{e}^{2\Omega_0^j(t)}\right)\Big|_{t=0} = 2\,\chi_0\left(\dot{\Omega}_0^j(0)\right)\mathrm{e}^{2\Omega_0^1}, \quad \textit{where} \quad \chi_0 := f(\mathrm{ad}_{2\Omega_0^1}); \tag{29}$$

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\left(\mathrm{e}^{2\Omega_i^j(t)}\right)\Big|_{t=1} = 2\,\chi_1\left(\dot{\Omega}_i^j(1)\right)\mathrm{e}^{2\Omega_2^1}, \quad \textit{where} \quad \chi_1 := f(\mathrm{ad}_{2\Omega_2^1}); \tag{30}$$

$$\begin{aligned} \Delta^L_{2(1-t)\Omega_0^j(t)}(t)\Big|_{t=0} &= -2\,\Omega_0^1 + 2\,\chi_0\left(\dot{\Omega}_0^j(0)\right); \\ \Delta^L_{2t\Omega_i^j(t)}(t)\Big|_{t=1} &= 2\,\Omega_2^1 + 2\,\chi_1\left(\dot{\Omega}_i^j(1)\right). \end{aligned} \tag{31}$$

Remark 2. In what follows, we must guarantee that the operators χ_0 and χ_1 have inverse. From the definition of f and g in (1) and (2) respectively, we know that $f(A)g(\mathrm{e}^A) = I$, for $\|e^A - I\| < 1$. So, if this restriction holds for $A = \mathrm{ad}_{2\Omega_0^1}$ and for $A = \mathrm{ad}_{2\Omega_2^1}$, taking into account the definitions of χ_0 and χ_1 above, we immediately obtain

$$\chi_0^{-1} := g(\mathrm{e}^{\mathrm{ad}_{2\Omega_0^1}}) \quad \text{and} \quad \chi_1^{-1} := g(\mathrm{e}^{\mathrm{ad}_{2\Omega_2^1}}). \tag{32}$$

Lemma 8 ([9])**.** *For $j = 2, 3$, let $i = 3 - j$. Then*

$$\dot{\Omega}_0^j(0) = (j-1)\,\chi_0^{-1}\left(\Omega_1^1 - \Omega_0^1\right), \tag{33}$$

$$\dot{\Omega}_i^j(1) = (j-1)\,\chi_1^{-1}\left(\Omega_2^1 - \mathrm{e}^{2\Omega_2^1}\,\Omega_1^1\,\mathrm{e}^{-2\Omega_2^1}\right). \tag{34}$$

Theorem 2. *The curve $t \in [0,1] \mapsto \beta_3(t)$ in $G_{k,n}$ defined in (25) satisfies the following boundary conditions:*

$$\beta_3(0) = x_0, \quad \beta_3(1) = x_3; \tag{35}$$

$$\dot{\beta}_3(0) = [3\Omega_0^1, x_0], \quad \frac{D\dot{\beta}_3}{dt}(0) = 6\left[\chi_0^{-1}\left(\Omega_1^1 - \Omega_0^1\right), x_0\right]; \tag{36}$$

$$\dot{\beta}_3(1) = [3\Omega_2^1, x_3], \quad \frac{D\dot{\beta}_3}{dt}(1) = 6\left[\chi_1^{-1}\left(\Omega_2^1 - \mathrm{e}^{2\Omega_2^1}\,\Omega_1^1\,\mathrm{e}^{-2\Omega_2^1}\right), x_3\right]. \tag{37}$$

Proof. We have already pointed out in Remark 1 that β_3 satisfies the first set of boundary conditions.

To prove the first condition in (36), we differentiate both sides of (25), use the identity $\mathrm{e}^{t\ \mathrm{ad}_{\Omega_0^1}}\,\Omega_0^1 = \Omega_0^1$, and Lemma 2, with $A(t)$ replaced by $t\Omega_0^j(t)$, for $j = 2, 3$, as follows.

$$\begin{aligned}\dot\beta_3(t) &= \Delta^L_{t\Omega_0^3(t)}(t)\beta_3(t) + \mathrm{e}^{t\Omega_0^3(t)}\,\Delta^L_{t\Omega_0^2(t)}(t)\,\mathrm{e}^{-t\Omega_0^3(t)}\,\beta_3(t)\\ &\quad + \left[\mathrm{e}^{t\Omega_0^3(t)}\,\mathrm{e}^{t\Omega_0^2(t)}\,\Omega_0^1\,\mathrm{e}^{-t\Omega_0^2(t)}\,\mathrm{e}^{-t\Omega_0^3(t)}, \beta_3(t)\right]\\ &\quad + \beta_3(t)\,\mathrm{e}^{t\Omega_0^3(t)}\,\mathrm{e}^{t\Omega_0^2(t)}\,\Delta^L_{-t\Omega_0^2(t)}(t)\,\mathrm{e}^{-t\Omega_0^2(t)}\,\mathrm{e}^{-t\Omega_0^3(t)}\\ &\quad + \beta_3(t)\,\mathrm{e}^{t\Omega_0^3(t)}\,\Delta^L_{-t\Omega_0^3(t)}(t)\,\mathrm{e}^{-t\Omega_0^3(t)}\\ &= [\Omega(t), \beta_3(t)]\,,\end{aligned} \tag{38}$$

where

$$\Omega(t) := \Delta^L_{t\Omega_0^3(t)}(t) + \mathrm{e}^{t\,\mathrm{ad}_{\Omega_0^3(t)}}\,\Delta^L_{t\Omega_0^2(t)}(t) + \mathrm{e}^{t\,\mathrm{ad}_{\Omega_0^3(t)}}\,\mathrm{e}^{t\,\mathrm{ad}_{\Omega_0^2(t)}}\,\Omega_0^1 \in \mathfrak{so}_{\beta_3(t)}(n).$$

Evaluating at $t = 0$, and taking into consideration Lemma 2 and $\Omega_0^2(0) = \Omega_0^3(0) = \Omega_0^1$, we obtain $\Omega(0) = 3\Omega_0^1$, which implies $\dot\beta_3(0) = [3\Omega_0^1, x_0]$.

To prove the second condition in (36), differentiate $\dot\beta_3(t)$ to get

$$\ddot\beta_3(t) = [\dot\Omega(t), \beta_3(t)] + [\Omega(t), [\Omega(t), \beta_3(t)]].$$

The first term above belongs to $T_{\beta_3(t)}G_{k,n}$ and, according to Lemma 3, the second term belongs to $(T_{\beta_3(t)}G_{k,n})^\perp$. So, since the covariant derivative of $\dot\beta_3$ is the tangential projection of $\ddot\beta_3$, it follows that

$$\frac{D\dot\beta_3}{dt}(t) = \left[\dot\Omega(t), \beta_3(t)\right].$$

Now, it is enough to evaluate at $t = 0$. This involves using the results in Lemmas 2 and 8, and several computations that are omitted due to lack of space, but can be found in [9]. Finally, the proof of identities (37) is similar to that of identities (36), but using instead of (25) the alternative expression for β_3 in Theorem 1, and evaluating at $t = 1$. QED

Obtaining the Control Points from the Boundary Conditions. In this subsection we will show how to get the control points from the boundary conditions, in order to implement the De Casteljau algorithm.

- **Case 1 - The Boundary Conditions are of Type (15)**

The boundary conditions (15) in Problem 1 can be written as:

$$\beta_3(0) = x_0, \quad \beta_3(1) = x_3, \quad \dot\beta_3(0) = [V_0, x_0], \quad \dot\beta_3(1) = [V_3, x_3], \tag{39}$$

where $x_0, x_3 \in G_{k,n}$, $V_0 \in \mathfrak{so}_{x_0}(n)$, and $V_3 \in \mathfrak{so}_{x_3}(n)$. According to the implementation of the algorithm, in order to generate the cubic polynomial that satisfies (39), we must be able to choose the control points x_1 and x_2 from those boundary conditions. The following theorem answers this question.

Theorem 3. *The control points x_1 and x_2, used in the De Casteljau algorithm to generate the geometric cubic polynomial that satisfies the boundary conditions (39), are given by:*

$$x_1 = \frac{1}{2}\left(I - \mathrm{e}^{\frac{2}{3}V_0}(I - 2x_0)\right), \quad x_2 = \frac{1}{2}\left(I - (I - 2x_3)\,\mathrm{e}^{\frac{2}{3}V_3}\right). \tag{40}$$

Proof. We know that $\mathrm{e}^{2\Omega_0^1} = (I - 2x_1)(I - 2x_0)$, or, equivalently, $I - 2x_1 = \mathrm{e}^{2\Omega_0^1}(I - 2x_0)$. Therefore, solving the last equation for x_1, and using $\Omega_0^1 = V_0/3$, which follows immediately by comparing the value of $\dot{\beta}_3(0)$ in Theorem 2 with the condition $\dot{\beta}_3(0) = [V_0, x_0]$ above, we obtain $x_1 = \frac{1}{2}\left(I - \mathrm{e}^{\frac{2}{3}V_0}(I - 2x_0)\right)$. The second identity is obtained with similar arguments, using the definition of Ω_2^1 and the obvious relation $\Omega_2^1 = V_3/3$. QED

- **Case 2 - The Boundary Conditions are of Type (16)**

The boundary conditions (16) in Problem 1 can be written as:

$$\beta_3(0) = x_0, \quad \beta_3(1) = x_3, \quad \dot{\beta}_3(0) = [V_0, x_0], \quad \frac{D\dot{\beta}_3}{dt}(0) = [W_0, x_0], \tag{41}$$

where $x_0, x_3 \in G_{k,n}$, and V_0, $W_0 \in \mathfrak{so}_{x_0}(n)$.

Theorem 4. *The control points x_1 and x_2, used in the De Casteljau algorithm to generate the geometric cubic polynomial that satisfies the boundary conditions (41), are given by:*

$$x_1 = \frac{1}{2}\left(I - \mathrm{e}^{\frac{2}{3}V_0}(I - 2x_0)\right), \; x_2 = \frac{1}{2}\left(I - \mathrm{e}^{\frac{1}{3}\chi_0(W_0) + \frac{2}{3}V_0}\,\mathrm{e}^{\frac{2}{3}V_0}(I - 2x_0)\right). \tag{42}$$

Proof. It is enough to obtain the control point x_2. Taking into account (41) and Theorem 2, we must have $6\chi_0^{-1}\left(\Omega_1^1 - \Omega_0^1\right) = W_0$, i.e., $2\Omega_1^1 = \frac{1}{3}\chi_0(W_0) + \frac{2}{3}V_0$. On the other hand, from the definition of Ω_1^1, $\mathrm{e}^{2\Omega_1^1} = (I - 2x_2)(I - 2x_1)$. So, solving for x_2, one gets $x_2 = \frac{1}{2}\left(I - \mathrm{e}^{2\Omega_1^1}(I - 2x_1)\right)$. Now, using the expression of Ω_1^1 and of x_1 in terms of x_0 and V_0, we obtain $x_2 = (I - \mathrm{e}^{\frac{1}{3}\chi_0(W_0) + \frac{2}{3}V_0}\,\mathrm{e}^{\frac{2}{3}V_0}(I - 2x_0))/2$, as required. QED

The Case 1, corresponding to the Hermite boundary conditions, can be considered simpler than the Case 2, since it doesn't involves the computation of covariant derivatives. However, the Case 2, where the data is not symmetrically specified, has computational advantages over the Case 1, namely whenever the goal is to generated cubic splines, i.e., piecing together several geometric cubic polynomials so that the overall curve is $\mathcal{C}^2$-smooth.

Corollary 1. *The relationship between the boundary conditions of types (15) and (16) is the following:*

$$W_0 = 3\chi_0^{-1}\left(\log\left((I-2x_3)\,\mathrm{e}^{\frac{2}{3}V_3}\;\mathrm{e}^{\frac{2}{3}V_0}(I-2x_0)\right) - \frac{2}{3}V_0\right),$$
$$V_3 = \frac{3}{2}\log\left((I-2x_3)\,\mathrm{e}^{\frac{1}{3}\chi_0(W_0)+\frac{2}{3}V_0}\,(I-2x_0)\,\mathrm{e}^{-\frac{2}{3}V_0}\right).$$

5 Generating Cubic Splines in $G_{k,n}$

We now explain how to solve the interpolation Problem 1 for the boundary conditions of type (16). The objective is to generate a geometric cubic spline, i.e., a $\mathcal{C}^2$-smooth curve that satisfies the interpolation and the boundary conditions and such that when restricted to each subinterval is a geometric cubic polynomial. The crucial procedure is the generation of the first cubic polynomial, denoted by γ_1, joining p_0 to p_1 and having prescribed initial velocity $[V_0, p_0]$ and initial covariant acceleration $[W_0, p_0]$. Although this has already been described in the previous section, we summarise the results here for the sake of completeness. We also adapt the notations so that the curve is given in terms of the data. The interpolation curve γ of Problem 1 may be generated by piecing together cubic polynomials defined on each subinterval $[t_i, t_{i+1}]$, $i = 0, 1, \ldots, \ell - 1$. Without loss of generality, we assume that all spline segments are parameterised in the $[0, 1]$ time interval.

5.1 Generating the First Spline Segment

Apply the De Casteljau algorithm to obtain the first spline segment

$$\gamma_1(t) = \mathrm{e}^{t\,\mathrm{ad}_{\Omega_0^3(t)}}\,\mathrm{e}^{t\,\mathrm{ad}_{\Omega_0^2(t)}}\,\mathrm{e}^{t\,\mathrm{ad}_{\Omega_0^1}}\,p_0, \tag{43}$$

where

$$\begin{aligned}
\Omega_0^1 &= \frac{1}{2}\log((I-2x_1)(I-2p_0));\\
\Omega_1^1 &= \frac{1}{2}\log((I-2x_2)(I-2x_1));\\
\Omega_2^1 &= \frac{1}{2}\log((I-2p_1)(I-2x_2));\\
\Omega_0^2(t) &= \frac{1}{2}\log((I-2\,\mathrm{e}^{t\,\mathrm{ad}_{\Omega_1^1}}\,x_1)(I-2\,\mathrm{e}^{t\,\mathrm{ad}_{\Omega_0^1}}\,p_0));\\
\Omega_1^2(t) &= \frac{1}{2}\log((I-2\,\mathrm{e}^{t\,\mathrm{ad}_{\Omega_2^1}}\,x_2)(I-2\,\mathrm{e}^{t\,\mathrm{ad}_{\Omega_1^1}}\,x_1));\\
\Omega_0^3(t) &= \frac{1}{2}\log((I-2\,\mathrm{e}^{t\,\mathrm{ad}_{\Omega_1^2(t)}}\,\mathrm{e}^{t\,\mathrm{ad}_{\Omega_1^1}}\,x_1)(I-2\,\mathrm{e}^{t\,\mathrm{ad}_{\Omega_0^2(t)}}\,\mathrm{e}^{t\,\mathrm{ad}_{\Omega_0^1}}\,p_0)),
\end{aligned}$$

and the control points are given by

$$x_1 = \frac{1}{2}(I - \mathrm{e}^{\frac{2}{3}V_0}(I-2p_0)); \quad x_2 = \frac{1}{2}(I - \mathrm{e}^{\frac{1}{3}\chi_0(W_0)+\frac{2}{3}V_0}\;\mathrm{e}^{\frac{2}{3}V_0}(I-2p_0)).$$

5.2 Generating Consecutive Spline Segments

After having generated the first spline segment, one continues in a similar way for the second spline segment. Since the cubic spline is required to be $\mathcal{C}^2$-smooth, the initial velocity and initial covariant acceleration for this second spline segment must equal the end velocity and the end covariant acceleration of the previous spline segment, which are given by the formulas in Theorem 2. The other $\ell - 2$ consecutive segments are generated similarly. The solution of Problem 1 is the cubic spline curve resulting from the concatenation of the ℓ consecutive segments.

6 Conclusion

We have presented all the details to implement the De Casteljau algorithm in the Grassmann manifold. For practical applications, one still has to rely on computing stable matrix exponentials and logarithms of structured matrices. This is out of the scope of our work, but efficient numerical methods to compute matrix functions have been developed along the years (see, for instance, [5]), and are expanding at a fast rate.

Acknowledgements. The authors acknowledge Fundação para a Ciência e a Tecnologia (FCT) and COMPETE 2020 program for financial support to project UIDB/00048/2020.

References

1. Batzies, E., Hüper, K., Machado, L., Silva Leite, F.: Geometric mean and geodesic regression on grassmannians. Linear Algebra Appl. **466**, 83–101 (2015)
2. Bressan, B.: From Physics to Daily Life: Applications in Informatics, Energy, and Environment. Wiley-Blackwell, Germany (2014)
3. Crouch, P., Kun, G., Silva Leite, F.: The de Casteljau algorithm on Lie groups and spheres. J. Dynamical Control Syst. **5**(3), 397–429 (1999)
4. de Casteljau, P.: Outillages Méthodes Calcul. Technical Report, André Citroën Automobiles SA (1959)
5. Higham, N.: Functions of Matrices: Theory and Computation. SIAM (2008)
6. Horn, R.A., Johnson, C.R.: Topics in Matrix Analysis. Cambridge University Press, New York (1991)
7. Hüper, K., Silva Leite, F.: On the geometry of rolling and interpolation curves on S^n, $\mathrm{SO_n}$, and Grassmann manifolds. J. Dynamical Control Syst. **13**(4), 467–502 (2007)
8. Park, F., Ravani, B.: Bézier curves on Riemannian manifolds and Lie groups with kinematics applications. ASME J. Mech. Des. **117**, 36–40 (1995)
9. Pina, F., Silva Leite, F.: Cubic splines in the Grassmann manifold generated by the De Casteljau algorithm. In: Pré-Publicações do Departamento de Matemática - Universidade de Coimbra, Portugal, number 20-07 (2020)
10. Sattinger, D.H., Weaver, O.L.: Lie Groups and Algebras with Applications to Physics, Geometry, and Mechanics. Applied Mathematical Sciences, vol. 61. Springer, New York (1986)