Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 33(10): 6493–6513.
DOI: 10.3934/era.2025286
Received: 28 August 2025
Revised: 09 October 2025
Accepted: 11 October 2025
Published: 29 October 2025

Research article

Determining Riemannian cubics and cubic splines with given boundary conditions

Erchuan Zhang[1] **and Simone Fiori**[2,*]

[1] School of Science, Sun Yat-sen University, Shenzhen 518107, China

[2] Faculty of Engineering, Marches Polytechnic University, Ancona 60131, Italy

* **Correspondence:** Email: s.fiori@staff.univpm.it.

Abstract: Riemannian cubics are critical points of the total squared norm of the accelerations of the curves on Riemannian manifolds with given end-points and end-velocities. Riemannian cubic splines are C^2 track sums of Riemannian cubics. Determining Riemannian cubics with given boundary data is equivalent to solving fourth-order ordinary differential equations with boundary conditions, while finding Riemannian cubic splines is tantamount to solving boundary and interior value problems, which are both hard in practice. The present paper proposes Riemannian gradient-based methods to tackle the former problem by taking advantage of the variational principle and the shooting method. The core idea is to add a number of junctions and associated velocities between given boundary points and to adjust such junctions and associated velocities according to a variational principle. Based on the obtained Riemannian cubics, Riemannian cubic splines are determined by constructing piecewise Riemannian cubics of class C^2 at interior points. The effectiveness of the proposed method is assessed by numerical experiments on a unit sphere.

Keywords: Riemannian cubic; Riemannian cubic spline; Riemannian gradient based method; Jacobi equation; Euler-Lagrange equation

1. Introduction

Riemannian cubics can be viewed as generalizations of cubic polynomials in Euclidean spaces to Riemannian manifolds, which were originally contributed by two independent works of Gabriel and Kajiya [1] and Noakes et al. [2]. In some emerging areas, such as quantum computing and computer vision, there is a growing need for interpolation on curved non-Euclidean spaces. For instances, quantum control of qubits gives rise to an interpolation problem in the complex projective space $\mathbb{CP}^n$ [3]. In computer science, intrinsic polynomial regression is adopted in the context of shape analysis in Kendall space and diffeomorphic landmark space [4, 5].

In Computer Vision, we shall reference to the work of Fletcher [6] that discusses geodesic regression as an intrinsic generalization of linear regression to Riemannian manifolds, providing a least-squares framework for modeling trends along geodesics, with theoretical guarantees for symmetric spaces and applications to rotation data and brain shape analysis. We shall also mention the work of Aubray and Nicol [7] on polynomial regression in Control Engineering, which presents a method for estimating the trajectory of a mobile system, such as a drone, from noisy position measurements by fitting geodesics in the Lie group SE(3) using a Riemannian least-squares approach, with results demonstrated on simulated data.

The generalization of cubic polynomials to Riemannian manifolds is not straightforward and may be achieved by several distinct methods. One approach is to minimize the acceleration from the perspective of geometric properties, which leads to solving a boundary value problem, i.e., a fourth-order ODE with boundary conditions. Another approach is to approximate Riemannian cubics by generalizing the classical de Casteljau algorithm to the Riemannian setting, where line segments are replaced by geodesic arcs [8]. The approximation error was analyzed in [9]. The present paper aligns with the effort to develop approximate methods for constructing Riemannian cubics and Riemannian cubic splines with given boundary data.

1.1. Riemannian cubics and cubic splines

Let M be a finite-dimensional path-connected Riemannian manifold endowed with a Riemannian metric $\langle\cdot,\cdot\rangle$, Levi-Civita connection ∇, and a Riemannian curvature tensor R. Given two points $x_0,\ x_T \in M$ and two tangent vectors $v_0 \in T_{x_0}M$ and $v_T \in T_{x_T}M$, we shall denote by C the space of all smooth curves $x : [0,T] \to M$ satisfying $x(0) = x_0$, $\dot{x}(0) = v_0$, $x(T) = x_T$, $\dot{x}(T) = v_T$, where $\dot{x}$ denotes the derivative of x with respect to the affine parameter t. *Riemannian cubics* are defined as the critical points of the following functional*

$$\mathcal{F}(x) := \frac{1}{2}\int_0^T \langle \nabla_t \dot{x}(t), \nabla_t \dot{x}(t)\rangle_{x(t)}\,\mathrm{d}t \tag{1.1}$$

over the set C. By the variational principle, it can be shown that Riemannian cubics satisfy the following Euler-Lagrange equation [2]

$$\nabla_t^3 \dot{x}(t) + R(\nabla_t \dot{x}(t), \dot{x}(t))\dot{x}(t) = \mathbf{0}. \tag{1.2}$$

When M is a Euclidean space, (1.2) can be reduced to $\frac{\mathrm{d}^4 x(t)}{\mathrm{d}t^4} = 0$, which means that x is a cubic polynomial in the variable t. We mention that the notion of polynomial curve on a Riemannian manifold, that cubic curve is a special case of, is not uniquely accepted, as there exist definitions that are not tied to a variational principle (see, for instance, the paper [10] on polynomial interpolation). The definition adopted depends essentially on the applications the theory is oriented to.

Defining cubic curves on a manifold via an action functional offers a variational framework that naturally incorporates boundary conditions and yields curves as minimizers of an energy-like quantity, providing geometric and physical interpretability, whereas purely setting covariant derivatives to zero prescribes local differential constraints without guaranteeing global optimality or boundary control. A

*The symbol ∇_t is used as a shortening for the symbol $\nabla_{d/dt}$, which denotes the covariant derivative of a vector field along the curve $x(t)$ with respect to time t.

cubic curve based on the criterion (1.1) by definition enjoys a well-understood smoothness property, namely, minimal covariant acceleration along the cubic. By the same token, a quintic will be endowed with a minimal-jerk character with clear advantages [11, 12]. We shall also mention that a cubic of the form (1.2) is a special trajectory of a fourth-order dynamical system defined on the basis of a more general functional than (1.1) studied in [13]. The existence and uniqueness of Riemannian cubics under some conditions have been studied in the literature, for example, [14–16].

In very few cases, closed forms of the solution of (1.2) can be given. In practice, solving the fourth-order ODE (1.2) with given boundary data is not easy. One commonly used way to circumvent this problem is the shooting method [17], where an unknown initial quantity is guessed and improved iteratively using the terminal error of a solution to an initial value problem. Note that the performance of the shooting method is highly dependent on the quality of the initial guess.

Riemannian cubic splines, on the other hand, are C^2 track sum of Riemannian cubics. That is, for a given data list (t_0, x_0), (t_1, x_1), ..., (t_l, x_l), where $0 = t_0 < t_1 < \cdots < t_l = T$, $x_j \in M$, $0 \le j \le l$, $v_0 \in T_{x_0}M$, $v_l \in T_{x_l}M$, a curve $x : [0, T] \to M$ is called a *Riemannian cubic spline* if and only if it satisfies

$$\begin{cases} \nabla_t^3 \dot{x}(t) + R(\nabla_t \dot{x}(t), \dot{x}(t))\dot{x}(t) = \mathbf{0}, \ \ 0 \le t \le T, \ \ t \ne t_j, \ \ j = 1, 2, \ldots, l-1, \\ x(t_j) = x_j, \ \ j = 0, 1, \ldots, l, \ \ \dot{x}(0) = v_0, \ \ \dot{x}(T) = v_l, \\ \dot{x}(t_j^+) = \dot{x}(t_j^-), \ \ \nabla_t \dot{x}(t_j^+) = \nabla_t \dot{x}(t_j^-), \ \ j = 1, 2, \ldots, l-1, \end{cases} \tag{1.3}$$

where t_j^+ (respectively t_j^-) denotes the limitation approaches to t_j from the right-hand (respectively left-hand) side. Specifically, a *natural* Riemannian cubic spline is a C^2 track-sum of Riemannian cubics with vanishing covariant accelerations at $t = 0$ and $t = T$, i.e., one obtained by replacing the conditions $\dot{x}(0) = v_0$, $\dot{x}(T) = v_l$ with $\nabla_t \dot{x}(0) = \mathbf{0}$, $\nabla_t \dot{x}(T) = \mathbf{0}$ in (1.3).

In comparison to Riemannian cubics, it is much more difficult to calculate Riemannian cubic splines as solutions of boundary and interior point problems. To the authors' best knowledge, intrinsic and effective methods are still needed to construct Riemannian cubic splines. In [18], Noakes proposed a method for finding approximations to natural Riemannian cubic splines in symmetric spaces based on bi-Jacobi fields along geodesics. Indeed, the proposed method is easy to implement, but it is restricted to near-geodesic natural Riemannian cubic splines in symmetric spaces. In [4], Kim et al. developed a smoothing spline fitting technique to Riemannian manifold data by generalizing the method of unrolling, unwrapping, and wrapping proposed by Jupp and Kent. However, their definition of smoothing spline is slightly different in the sense that the smoothing spline on Riemannian manifold is the wrapping of the unrolling cubic smoothing spline fitted to the unwrapped data on the tangent space. More references on theoretical analyses and computations of splines on Riemannian manifolds can be found in [19–22].

1.2. Motivation of the present endeavor

Consider first the problem of determining a Riemannian cubic fitting given boundary conditions. If the boundaries are too far apart, the problem might lead to inconsistent solutions. As a workaround, let us introduce a number n of segments delimited by junctions $z_0 := x_0$, z_1, ..., z_{n-1}, $z_n := x_T$ at time $0 =: t_0 < t_1 < \cdots < t_{n-1} < t_n := T$, where the number of junctions and the values of the parameters t_j are fixed, while the locations of the junctions in M are to be determined. Throughout this paper,

we assume there is a unique Riemannian cubic joining given boundary conditions. The core idea is to *determine such locations in a way that makes the obtained curve coincide exactly with the sought cubic*. To do so, we shall proceed by expressing the functional $\mathcal{F}$ in terms of pieces and by claiming that the set of junctions that makes such functional stationary builds a cubic spline as a whole.

The first variation of a curve x in the functional (1.1) is calculated by introducing a family of curves $x_\epsilon(t) \in C$, where $\epsilon \in (-a, a)$, with $a > 0$, denotes a continuous index such that $\epsilon = 0$ corresponds to the actual path. Then, a variation δ is interpreted to act as $\nabla_\epsilon|_{\epsilon=0}$ on vector fields and as $\partial_\epsilon|_{\epsilon=0}$ on scalars and on the path. The variation of the functional $\mathcal{F}$ is hence given by

$$\begin{aligned}
\delta\mathcal{F} &= \frac{1}{2}\sum_{i=0}^{n-1}\int_{t_i}^{t_{i+1}} \delta\langle\nabla_t\dot{x}(t), \nabla_t\dot{x}(t)\rangle_{x(t)}\,\mathrm{d}t \\
&= \sum_{i=0}^{n-1}\int_{t_i}^{t_{i+1}} \langle\nabla_t\dot{x}(t), \nabla_t^2(\delta x(t)) + R(\delta x(t), \dot{x}(t))\dot{x}(t)\rangle_{x(t)}\,\mathrm{d}t \\
&= \sum_{i=0}^{n-1}\Big[\langle\nabla_t\dot{x}(t), \nabla_t(\delta x(t))\rangle_{x(t)}\big|_{t_i}^{t_{i+1}} - \langle\nabla_t^2\dot{x}(t), \delta x(t)\rangle_{x(t)}\big|_{t_i}^{t_{i+1}} \\
&\quad + \int_{t_i}^{t_{i+1}} \langle\nabla_t^3\dot{x}(t) + R(\nabla_t\dot{x}(t), \dot{x}(t))\dot{x}(t), \delta x(t)\rangle_{x(t)}\,\mathrm{d}t\Big]
\end{aligned} \tag{1.4}$$

where x is a piecewise Riemannian cubic on $\{[t_i, t_{i+1}]\}_{i=0}^{n-1}$ with given boundary data, hence it satisfies (1.2). It is the case to recall that, at each t_i, for $i = 1, 2, \ldots, n-1$, two cubics (one from the 'left-hand side' and one from the 'right-hand side') that meet, share velocity $\dot{x}$, but not acceleration $\nabla_t\dot{x}$ nor jerk $\nabla_t^2\dot{x}$. In addition, both δx and $\nabla_t\delta x$ are arbitrary and can be chosen in such a way that they join smoothly at each point t_i, namely $\delta x(t_i^-) \equiv \delta x(t_i^+)$ and $\nabla_t\delta x(t_i^-) \equiv \nabla_t\delta x(t_i^+)$, where t_i^+ (respectively t_i^-) denotes the limitation approaches to t_i from the right-hand (respectively left-hand) side. On the basis of the above observation, the variation of the functional $\mathcal{F}$ may be recast as

$$\delta\mathcal{F} = \sum_{i=1}^{n-1}\Big[\langle\nabla_t\dot{x}(t_i^+) - \nabla_t\dot{x}(t_i^-), \nabla_t\delta x(t_i)\rangle_{x(t_i)} - \langle\nabla_t^2\dot{x}(t_i^+) - \nabla_t^2\dot{x}(t_i^-), \delta x(t_i)\rangle_{x(t_i)}\Big]. \tag{1.5}$$

Then, the variation (1.5) vanishes provided $\nabla_t\dot{x}(t)$ and $\nabla_t^2\dot{x}(t)$ are continuous across all junctions. This motivates the core idea of constructing Riemannian cubics in this paper by minimizing the following function:

$$f(\{(z_i, v_i)\}_{i=1}^{n-1}) := \frac{1}{2}\sum_{i=1}^{n-1}\Big(\|\nabla_t\dot{x}(t_i^+) - \nabla_t\dot{x}(t_i^-)\|^2 + \|\nabla_t^2\dot{x}(t_i^+) - \nabla_t^2\dot{x}(t_i^-)\|^2\Big) \tag{1.6}$$

over the $(n-1)$-fold tangent bundle $(TM)^{n-1}$, where $z_i = x(t_i^+) = x(t_i^-)$, $v_i = \dot{x}(t_i^+) = \dot{x}(t_i^-)$, $1 \le i \le n-1$.

Successive boundary conditions are reasonably close so that Riemannian cubics on each time interval can be easily found using the shooting method. This is because when the boundary conditions are close enough, the Riemannian cubic curve is not too far from Euclidean cubic polynomial, and then the projections of the initial values of the cubic polynomial can be used as initial guesses for the shooting method.

For solving the Riemannian cubic spline problem (1.3), let x be a piecewise Riemannian cubic in $\{[t_j, t_{j+1}]\}_{j=0}^{l-1}$ satisfying $x(t_j) = x_j$, $\dot{x}(t_j) = v_j$, $j = 0, 1, \ldots, l$, which is found by minimizing (1.6) or by using the shooting method. To make sure x is C^2 at x_j, $j = 1, 2, \ldots, l-1$, we intend to minimizing the following function,

$$g(\{v_j\}_{j=1}^{l-1}) := \frac{1}{2}\sum_{j=1}^{l-1} \|\nabla_t \dot{x}(t_j^+) - \nabla_t \dot{x}(t_j^-)\|^2, \tag{1.7}$$

where $v_j = \dot{x}(t_j^+) = \dot{x}(t_j^-)$, $j = 1, 2, \ldots, l-1$.

This paper is organized as follows. In Section 2, we present some key blocks to build our algorithms, including how to find local Riemannian cubics that join consecutive boundary conditions and how to calculate the Riemannian gradient of covariant acceleration and second-order covariant acceleration with respect to interior points and velocities. By incorporating the building blocks developed in Section 2, the general framework for determining Riemannian cubics and cubic splines is summarized in Section 3. Then, in the following section, we demonstrate the feasibility and efficiency of the proposed methods by implementing numerical experiments about the unit sphere $\mathbb{S}^2$, where the well-known manifold optimization package Manopt is used as the backbone of our Riemannian gradient-based methods [23]. Section 5 concludes this paper with comments on the proposed method and outlines possible future endeavors.

2. Building blocks for determining Riemannian cubics and cubic splines

In order to get the minimizer of the function (1.6) and that of (1.7) using Riemannian-gradient-based methods, such as Riemannian gradient descent [24,25], Riemannian conjugate gradient descent [26,27] and Riemannian limited-memory BFGS [28,29], we have two key blocks to build: (1) finding the local Riemannian cubics joining the given boundary conditions (z_i, v_i) and (z_{i+1}, v_{i+1}) (which is supposed to work only if they are not too far away), and (2) evaluating the Riemannian gradients of $\nabla_t \dot{x}(t_i^+)$, $\nabla_t^2 \dot{x}(t_i^+)$, $\nabla_t \dot{x}(t_{i+1}^-)$, $\nabla_t^2 \dot{x}(t_{i+1}^-)$ with respect to z_i, v_i, z_{i+1}, v_{i+1}, where $i = 0, 1, \ldots, n-1$. The first block can be built by using the single shooting method to solve the Riemannian cubic equation (1.2), provided good initial guesses are available, which are achieved by the projection of the Euclidean cubic polynomials. The second block can be built by solving the Jacobi equation along the Riemannian cubics.

2.1. Initial guesses for solving Riemannian cubic equation

It is possible that we may find multiple Riemannian cubics to join the boundary conditions (z_0, v_0) and (z_n, v_n), which may also depend on the initial choice for $(z_1, v_1), \ldots, (z_{n-1}, v_{n-1}) \in TM$. From now on, we shall assume that consecutive conditions are reasonably close in the sense that (z_i, v_i) and (z_{i+1}, v_{i+1}) can be joined by a *unique* Riemannian cubic curve with fairly easy initial guesses for the single shooting method.

By the embedding theorem, any Riemannian manifold can be embedded in a higher-dimensional Euclidean space. Then, the cubic polynomial $y^{(i)}$ in Euclidean space joining the boundary conditions

$(z_i, v_i) \in TM$ (at $t = t_i$) and $(z_{i+1}, v_{i+1}) \in TM$ (at $t = t_{i+1}$) can be written as

$$\begin{aligned} y^{(i)}(t) = z_i + v_i(t - t_i) + \left(\frac{3(z_{i+1} - z_i)}{(t_{i+1} - t_i)^2} - \frac{v_{i+1} + 2v_i}{t_{i+1} - t_i}\right)(t - t_i)^2 \\ + \left(\frac{v_{i+1} + v_i}{(t_{i+1} - t_i)^2} - \frac{2(z_{i+1} - z_i)}{(t_{i+1} - t_i)^3}\right)(t - t_i)^3. \end{aligned} \tag{2.1}$$

Let $\mathcal{P}$ denote the projection from Euclidean space to the tangent space of the Riemannian manifold M at z_i. Then, the tangent vectors

$$\mathcal{P}(\ddot{y}^{(i)}(t_i)) = \mathcal{P}\left(\frac{6(z_{i+1} - z_i)}{(t_{i+1} - t_i)^2} - \frac{2(v_{i+1} + 2v_i)}{t_{i+1} - t_i}\right) \tag{2.2}$$

and

$$\mathcal{P}(\dddot{y}^{(i)}(t_i)) = \mathcal{P}\left(\frac{6(v_{i+1} + v_i)}{(t_{i+1} - t_i)^2} - \frac{12(z_{i+1} - z_i)}{(t_{i+1} - t_i)^3}\right) \tag{2.3}$$

can serve as the initial guesses of high quality for solving the Riemannian cubic equation (1.2) by the single shooting method.

Example 1. Suppose M is a m-dimensional Riemannian manifold embedded in a $(m + 1)$-dimensional Euclidean space $\mathbb{E}^{m+1}$ isometrically, then $M = l^{-1}(c)$ for some smooth function $l : M \subset \mathbb{E}^{m+1} \to \mathbb{R}$ and some constant $c \in \mathbb{R}$. Any vector v in $\mathbb{E}^{m+1}$ at $x \in M$ can be projected onto the tangent space T_xM of M at x as follows,

$$\mathcal{P}(v) = v - \langle v, \mathbf{n}_\mathbf{x}\rangle \mathbf{n}_\mathbf{x}, \tag{2.4}$$

where $\mathbf{n}_\mathbf{x} = \frac{\text{grad } l(x)}{\|\text{grad } l(x)\|}$ denotes the unit normal to the manifold M, grad l is the Euclidean gradient of l, and $\|\cdot\|$ is the norm induced by the standard Euclidean metric.

In particular, if M is the m-dimensional unit sphere $\mathbb{S}^m$, then $l(x) = x^\top x = 1$ for any $x \in \mathbb{S}^m$, grad $l(x) = 2x$, and $\mathbf{n}_\mathbf{x} = x$, which implies the projection

$$\mathcal{P}(v) = v - \langle v, x\rangle x \tag{2.5}$$

for any $v \in \mathbb{E}^{m+1}$.

2.2. Evaluate Riemannian gradients

For $0 \le i \le n - 1$, let $x^{(i)}$ represent the Riemannian cubic joining the boundary conditions (z_i, v_i) (at $t = t_i$) and (z_{i+1}, v_{i+1}) (at $t = t_{i+1}$) determined by the aforementioned method. The aim of this subsection is to evaluate the Riemannian gradients of $\nabla_t \dot{x}^{(i)}(t_i^+)$, $\nabla_t^2 \dot{x}^{(i)}(t_i^+)$, $\nabla_t \dot{x}^{(i)}(t_{i+1}^-)$, and $\nabla_t^2 \dot{x}^{(i)}(t_{i+1}^-)$ with respect to the boundary conditions $z_i, v_i, z_{i+1}, v_{i+1}$. Such gradients will serve the purpose of iteratively updating the initial guess of the junctions z_i and their associated velocity vectors until a reasonable approximation of the cubic sought is attained.

To derive the *Jacobi equation for Riemannian cubics*, we consider a variation $x_\epsilon^{(i)} : [t_i, t_{i+1}] \times (-a, a) \to M$ of $x^{(i)}$ for some small $a > 0$. In short notations, we write x for $x^{(i)}(t)$, and x_ϵ for $x_\epsilon^{(i)}(t)$. Let $J(t) := (\partial_\epsilon x_\epsilon(t))|_{\epsilon=0}$, and let us recall the properties (see [30])

$$\nabla_J \dot{x} = \nabla_t J \quad \text{and} \quad \nabla_J \nabla_t v = \nabla_t \nabla_J v + R(J, \dot{x})v, \tag{2.6}$$

for every smooth vector field $v \in \Gamma(TM)$. The Jacobi equation for Riemannian cubics is an analogous of the Jacobi equation for geodesics and may be derived from the Riemannian cubic equation (1.2) by applying the covariant derivative operator ∇_J to both sides, which readily gives

$$\nabla_J \nabla_t^3 \dot{x} + \nabla_J R(\nabla_t \dot{x}, \dot{x})\dot{x} = \mathbf{0}. \tag{2.7}$$

The two addenda on the left-hand side are calculated as follows:

$$\begin{aligned}
\nabla_J \nabla_t^3 \dot{x} &= \nabla_t \nabla_J \nabla_t^2 \dot{x} + R(J, \dot{x})\nabla_t^2 \dot{x} \\
&= \nabla_t(\nabla_t \nabla_J \nabla_t \dot{x} + R(J, \dot{x})\nabla_t \dot{x}) + R(J, \dot{x})\nabla_t^2 \dot{x} \\
&= \nabla_t(\nabla_t(\nabla_t \nabla_t J + R(J, \dot{x})\dot{x}) + R(J, \dot{x})\nabla_t \dot{x}) + R(J, \dot{x})\nabla_t^2 \dot{x} \\
&= \nabla_t(\nabla_t^3 J + (\nabla_t R)(J, \dot{x})\dot{x} + R(\nabla_t J, \dot{x})\dot{x} + R(J, \nabla_t \dot{x})\dot{x} + R(J, \dot{x})\nabla_t \dot{x}) \\
&\quad + (\nabla_t R)(J, \dot{x})\nabla_t \dot{x} + R(\nabla_t J, \dot{x})\nabla_t \dot{x} + R(J, \nabla_t \dot{x})\nabla_t \dot{x} + 2R(J, \dot{x})\nabla_t^2 \dot{x} \\
&= \nabla_t^4 J + (\nabla_t^2 R)(J, \dot{x})\dot{x} + (\nabla_t R)(\nabla_t J, \dot{x})\dot{x} + (\nabla_t R)(J, \nabla_t \dot{x})\dot{x} + (\nabla_t R)(J, \dot{x})\nabla_t \dot{x} \\
&\quad + (\nabla_t R)(\nabla_t J, \dot{x})\dot{x} + R(\nabla_t^2 J, \dot{x})\dot{x} + R(\nabla_t J, \nabla_t \dot{x})\dot{x} + R(\nabla_t J, \dot{x})\nabla_t \dot{x} \\
&\quad + (\nabla_t R)(J, \nabla_t \dot{x})\dot{x} + R(\nabla_t J, \nabla_t \dot{x})\dot{x} + R(J, \nabla_t^2 \dot{x})\dot{x} + R(J, \nabla_t \dot{x})\nabla_t \dot{x} \\
&\quad + (\nabla_t R)(J, \dot{x})\nabla_t \dot{x} + R(\nabla_t J, \dot{x})\nabla_t \dot{x} + R(J, \nabla_t \dot{x})\nabla_t \dot{x} + R(J, \dot{x})\nabla_t^2 \dot{x} \\
&\quad + (\nabla_t R)(J, \dot{x})\nabla_t \dot{x} + R(\nabla_t J, \dot{x})\nabla_t \dot{x} + R(J, \nabla_t \dot{x})\nabla_t \dot{x} + 2R(J, \dot{x})\nabla_t^2 \dot{x} \\
&= \nabla_t^4 J + (\nabla_t^2 R)(J, \dot{x})\dot{x} + 2(\nabla_t R)(\nabla_t J, \dot{x})\dot{x} + 2(\nabla_t R)(J, \nabla_t \dot{x})\dot{x} + 3(\nabla_t R)(J, \dot{x})\nabla_t \dot{x} \\
&\quad + R(\nabla_t^2 J, \dot{x})\dot{x} + 2R(\nabla_t J, \nabla_t \dot{x})\dot{x} + 3R(\nabla_t J, \dot{x})\nabla_t \dot{x} + R(J, \nabla_t^2 \dot{x})\dot{x} \\
&\quad + 3R(J, \nabla_t \dot{x})\nabla_t \dot{x} + 3R(J, \dot{x})\nabla_t^2 \dot{x}
\end{aligned}$$

and

$$\begin{aligned}
\nabla_J R(\nabla_t \dot{x}, \dot{x})\dot{x} &= (\nabla_J R)(\nabla_t \dot{x}, \dot{x})\dot{x} + R(\nabla_J \nabla_t \dot{x}, \dot{x})\dot{x} + R(\nabla_t \dot{x}, \nabla_J \dot{x})\dot{x} + R(\nabla_t \dot{x}, \dot{x})\nabla_J \dot{x} \\
&= (\nabla_J R)(\nabla_t \dot{x}, \dot{x})\dot{x} + R(\nabla_t \nabla_t J + R(J, \dot{x})\dot{x}, \dot{x})\dot{x} + R(\nabla_t \dot{x}, \nabla_t J)\dot{x} + R(\nabla_t \dot{x}, \dot{x})\nabla_t J \\
&= (\nabla_J R)(\nabla_t \dot{x}, \dot{x})\dot{x} + R(\nabla_t^2 J, \dot{x})\dot{x} + R(\nabla_t \dot{x}, \nabla_t J)\dot{x} + R(\nabla_t \dot{x}, \dot{x})\nabla_t J + R(R(J, \dot{x})\dot{x}, \dot{x})\dot{x}.
\end{aligned}$$

Therefore, we get the Jacobi equation of Riemannian cubics as follows:

$$\begin{aligned}
&\nabla_t^4 J + (\nabla_t^2 R)(J, \dot{x})\dot{x} + (\nabla_J R)(\nabla_t \dot{x}, \dot{x})\dot{x} + R(R(J, \dot{x})\dot{x}, \dot{x})\dot{x} \\
&+ R(J, \nabla_t^2 \dot{x})\dot{x} + 2\left((\nabla_t R)(\nabla_t J, \dot{x})\dot{x} + (\nabla_t R)(J, \nabla_t \dot{x})\dot{x} + R(\nabla_t^2 J, \dot{x})\dot{x}\right) \\
&+ 3\left((\nabla_t R)(J, \dot{x})\nabla_t \dot{x} + R(J, \dot{x})\nabla_t^2 \dot{x} + R(J, \nabla_t \dot{x})\nabla_t \dot{x}\right) + 4R(\nabla_t J, \dot{x})\nabla_t \dot{x} = \mathbf{0},
\end{aligned} \tag{2.8}$$

where we have used the algebraic Bianchi identity together with the skew-symmetry property of the curvature tensor to get

$$R(\nabla_t J, \nabla_t \dot{x})\dot{x} + R(\nabla_t \dot{x}, \dot{x})\nabla_t J = -R(\dot{x}, \nabla_t J)\nabla_t \dot{x} = R(\nabla_t J, \dot{x})\nabla_t \dot{x}.$$

The Jacobi equation (2.8) appears as a fourth-order differential equation in the Jacobi field J. Notice that a similar equation can be found in the work of Crouch and Leite [31] (see relation (14) therein) but with a different notation. Solving the differential equation (2.8) on M is hard in practice, however, upon resorting to an embedding into a larger Euclidean space, it can be tackled by one of the tools available such as the ODE45 method in MATLAB®.

Next, we investigate the Riemannian gradients of $\nabla_t\dot{x}^{(i)}(t_i^+)$, $\nabla_t^2\dot{x}^{(i)}(t_i^+)$, $\nabla_t\dot{x}^{(i)}(t_{i+1}^-)$, and $\nabla_t^2\dot{x}^{(i)}(t_{i+1}^-)$ with respect to $z_i, v_i, z_{i+1}, v_{i+1}$. In order to perform the following calculations, we found it convenient to resort again to an appropriate embedding of the base manifold M. Let Γ denote the Christoffel symbol associated with the Levi-Civita connection ∇. Then[†]

$$\begin{cases}\nabla_t\dot{x} = \ddot{x} + \Gamma_x(\dot{x},\dot{x}),\\ \nabla_t^2\dot{x} = \dddot{x} + 3\Gamma_x(\dot{x},\ddot{x}) + d\Gamma_x(\dot{x})(\dot{x},\dot{x}) + \Gamma_x(\dot{x},\Gamma_x(\dot{x},\dot{x}))\end{cases}$$

(for reference, one might consult, e.g., the tutorial papers [32, 33]). Thus, we first consider the (Euclidean) derivatives of $\ddot{x}^{(i)}(t_i^+)$, $\dddot{x}^{(i)}(t_i^+)$, $\ddot{x}^{(i)}(t_{i+1}^-)$, and $\dddot{x}^{(i)}(t_{i+1}^-)$ with respect to z_i, v_i, z_{i+1}, and v_{i+1}. In short, we write u_i, w_i, u_{i+1}, and w_{i+1} for $\ddot{x}^{(i)}(t_i^+)$, $\dddot{x}^{(i)}(t_i^+)$, $\ddot{x}^{(i)}(t_{i+1}^-)$, and $\dddot{x}^{(i)}(t_{i+1}^-)$, respectively.

Suppose $E : [t_i,\, t_{i+1}] \to M$ is the Riemannian cubic joining the boundary points of data (z_i, v_i) and (z_{i+1}, v_{i+1}), denoted by[‡]

$$E(t) := E\left(z_i, \frac{t-t_i}{t_{i+1}-t_i}v_i, \frac{t-t_i}{t_{i+1}-t_i}u_i, \frac{t-t_i}{t_{i+1}-t_i}w_i\right),$$

where $E(t_k) = z_k$, $\dot{E}(t_k) = v_k$, $\ddot{E}(t_k) = u_k$, and $\dddot{E}(t_k) = w_k$, if $k = i,\, i+1$. Differentiating both sides of the relation[§] $E(z_i, v_i, u_i, w_i) = z_{i+1}$ with respect to z_{i+1} and v_{i+1}, respectively, we get

$$\partial_{u_i}E\frac{\partial u_i}{\partial z_{i+1}} + \partial_{w_i}E\frac{\partial w_i}{\partial z_{i+1}} = I,\;\; \partial_{u_i}E\frac{\partial u_i}{\partial v_{i+1}} + \partial_{w_i}E\frac{\partial w_i}{\partial v_{i+1}} = 0. \tag{2.9}$$

Differentiating[¶] $\dot{E}(z_i, v_i, u_i, w_i) = v_{i+1}$ with respect to z_{i+1} and v_{i+1}, respectively, we get

$$\partial_{u_i}\dot{E}\frac{\partial u_i}{\partial z_{i+1}} + \partial_{w_i}\dot{E}\frac{\partial w_i}{\partial z_{i+1}} = 0,\;\; \partial_{u_i}\dot{E}\frac{\partial u_i}{\partial v_{i+1}} + \partial_{w_i}\dot{E}\frac{\partial w_i}{\partial v_{i+1}} = I. \tag{2.10}$$

Combining (2.9) and (2.10), we find

$$\begin{bmatrix}\partial_{u_i}E & \partial_{w_i}E\\ \partial_{u_i}\dot{E} & \partial_{w_i}\dot{E}\end{bmatrix}\cdot\begin{bmatrix}\frac{\partial u_i}{\partial z_{i+1}} & \frac{\partial u_i}{\partial v_{i+1}}\\ \frac{\partial w_i}{\partial z_{i+1}} & \frac{\partial w_i}{\partial v_{i+1}}\end{bmatrix} = I \implies \begin{bmatrix}\frac{\partial u_i}{\partial z_{i+1}} & \frac{\partial u_i}{\partial v_{i+1}}\\ \frac{\partial w_i}{\partial z_{i+1}} & \frac{\partial w_i}{\partial v_{i+1}}\end{bmatrix} = \begin{bmatrix}\partial_{u_i}E & \partial_{w_i}E\\ \partial_{u_i}\dot{E} & \partial_{w_i}\dot{E}\end{bmatrix}^{-1}. \tag{2.11}$$

Similarly, straightforward calculations yield

$$\begin{aligned}&\partial_{z_i}E + \partial_{u_i}E\frac{\partial u_i}{\partial z_i} + \partial_{w_i}E\frac{\partial w_i}{\partial z_i} = 0,\;\; \partial_{v_i}E + \partial_{u_i}E\frac{\partial u_i}{\partial v_i} + \partial_{w_i}E\frac{\partial w_i}{\partial v_i} = 0,\\ &\partial_{z_i}\dot{E} + \partial_{u_i}\dot{E}\frac{\partial u_i}{\partial z_i} + \partial_{w_i}\dot{E}\frac{\partial w_i}{\partial z_i} = 0,\;\; \partial_{v_i}\dot{E} + \partial_{u_i}\dot{E}\frac{\partial u_i}{\partial v_i} + \partial_{w_i}\dot{E}\frac{\partial w_i}{\partial v_i} = 0,\\ &\implies \begin{bmatrix}\frac{\partial u_i}{\partial z_i} & \frac{\partial u_i}{\partial v_i}\\ \frac{\partial w_i}{\partial z_i} & \frac{\partial w_i}{\partial v_i}\end{bmatrix} = -\begin{bmatrix}\partial_{u_i}E & \partial_{w_i}E\\ \partial_{u_i}\dot{E} & \partial_{w_i}\dot{E}\end{bmatrix}^{-1}\cdot\begin{bmatrix}\partial_{z_i}E & \partial_{v_i}E\\ \partial_{z_i}\dot{E} & \partial_{v_i}\dot{E}\end{bmatrix}.\end{aligned} \tag{2.12}$$

[†]The second identity holds because $\nabla_t^2\dot{x} = \frac{d}{dt}(\nabla_t\dot{x}) + \Gamma_x(\dot{x},\nabla_t\dot{x}) = \frac{d}{dt}(\ddot{x}+\Gamma_x(\dot{x},\dot{x})) + \Gamma_x(\dot{x},\ddot{x}+\Gamma_x(\dot{x},\dot{x})) = \dddot{x} + 3\Gamma_x(\dot{x},\ddot{x}) + d\Gamma_x(\dot{x})(\dot{x},\dot{x}) + \Gamma_x(\dot{x},\Gamma_x(\dot{x},\dot{x}))$.

[‡]We use this notation simply because E is a solution of the fourth-order ODE (1.2), which can be determined by four initial conditions. Besides, the notation mimics the standard exponential map $\exp_{z_i}\left(\frac{t-t_i}{t_{i+1}-t_i}v_i\right)$, which is a geodesic starting from z_i and along the direction v_i (determined by two initial conditions).

[§]Note that $E(t_{i+1}) = E(z_i, v_i, u_i, w_i) = z_{i+1}$ by the definition of E.

[¶]Note that $\dot{E}(t_{i+1}) = \dot{E}(z_i, v_i, u_i, w_i) = v_{i+1}$ by the definition of E, which is the velocity of the Riemannian cubic curve E at $t = t_{i+1}$.

$$\begin{aligned}
&\partial_{z_i}\ddot{E} + \partial_{u_i}\ddot{E}\frac{\partial u_i}{\partial z_i} + \partial_{w_i}\ddot{E}\frac{\partial w_i}{\partial z_i} = \frac{\partial u_{i+1}}{\partial z_i},\quad \partial_{v_i}\ddot{E} + \partial_{u_i}\ddot{E}\frac{\partial u_i}{\partial v_i} + \partial_{w_i}\ddot{E}\frac{\partial w_i}{\partial v_i} = \frac{\partial u_{i+1}}{\partial v_i},\\
&\partial_{z_i}\dddot{E} + \partial_{u_i}\dddot{E}\frac{\partial u_i}{\partial z_i} + \partial_{w_i}\dddot{E}\frac{\partial w_i}{\partial z_i} = \frac{\partial w_{i+1}}{\partial z_i},\quad \partial_{v_i}\dddot{E} + \partial_{u_i}\dddot{E}\frac{\partial u_i}{\partial v_i} + \partial_{w_i}\dddot{E}\frac{\partial w_i}{\partial v_i} = \frac{\partial w_{i+1}}{\partial v_i},\\
&\Longrightarrow \begin{bmatrix} \frac{\partial u_{i+1}}{\partial z_i} & \frac{\partial u_{i+1}}{\partial v_i} \\ \frac{\partial w_{i+1}}{\partial z_i} & \frac{\partial w_{i+1}}{\partial v_i} \end{bmatrix} = \begin{bmatrix} \partial_{z_i}\ddot{E} & \partial_{v_i}\ddot{E} \\ \partial_{z_i}\dddot{E} & \partial_{v_i}\dddot{E} \end{bmatrix} + \begin{bmatrix} \partial_{u_i}\ddot{E} & \partial_{w_i}\ddot{E} \\ \partial_{u_i}\dddot{E} & \partial_{w_i}\dddot{E} \end{bmatrix}\cdot\begin{bmatrix} \frac{\partial u_i}{\partial z_i} & \frac{\partial u_i}{\partial v_i} \\ \frac{\partial w_i}{\partial z_i} & \frac{\partial w_i}{\partial v_i} \end{bmatrix}.
\end{aligned} \tag{2.13}$$

$$\begin{aligned}
&\partial_{u_i}\ddot{E}\frac{\partial u_i}{\partial z_{i+1}} + \partial_{w_i}\ddot{E}\frac{\partial w_i}{\partial z_{i+1}} = \frac{\partial u_{i+1}}{\partial z_{i+1}},\quad \partial_{u_i}\ddot{E}\frac{\partial u_i}{\partial v_{i+1}} + \partial_{w_i}\ddot{E}\frac{\partial w_i}{\partial v_{i+1}} = \frac{\partial u_{i+1}}{\partial v_{i+1}},\\
&\partial_{u_i}\dddot{E}\frac{\partial u_i}{\partial z_{i+1}} + \partial_{w_i}\dddot{E}\frac{\partial w_i}{\partial z_{i+1}} = \frac{\partial w_{i+1}}{\partial z_{i+1}},\quad \partial_{u_i}\dddot{E}\frac{\partial u_i}{\partial v_{i+1}} + \partial_{w_i}\dddot{E}\frac{\partial w_i}{\partial v_{i+1}} = \frac{\partial w_{i+1}}{\partial v_{i+1}},\\
&\Longrightarrow \begin{bmatrix} \frac{\partial u_{i+1}}{\partial z_{i+1}} & \frac{\partial u_{i+1}}{\partial v_{i+1}} \\ \frac{\partial w_{i+1}}{\partial z_{i+1}} & \frac{\partial w_{i+1}}{\partial v_{i+1}} \end{bmatrix} = \begin{bmatrix} \partial_{u_i}\ddot{E} & \partial_{w_i}\ddot{E} \\ \partial_{u_i}\dddot{E} & \partial_{w_i}\dddot{E} \end{bmatrix}\cdot\begin{bmatrix} \frac{\partial u_i}{\partial z_{i+1}} & \frac{\partial u_i}{\partial v_{i+1}} \\ \frac{\partial w_i}{\partial z_{i+1}} & \frac{\partial w_i}{\partial v_{i+1}} \end{bmatrix}.
\end{aligned} \tag{2.14}$$

Now, let E_ϵ represent a variation of the curve E defined by

$$E_\epsilon(t) = E\left(\exp(z_i, \epsilon\alpha), \frac{t-t_i}{t_{i+1}-t_i}v_i(\epsilon), \frac{t-t_i}{t_{i+1}-t_i}u_i(\epsilon), \frac{t-t_i}{t_{i+1}-t_i}w_i(\epsilon)\right),$$

where $\alpha \in T_{z_i}M$ defines a variation of the point z_i along the geodesic $\exp(z_i, \epsilon\alpha)$. Vectors v_i, u_i, and w_i have been extended as vector fields $v_i(\epsilon)$, $u_i(\epsilon)$, and $w_i(\epsilon)$ along $\exp(z_i, \epsilon\alpha)$ via parallel translations. This variation determines a Jacobi field via the relation (2.7), which is a solution J of the fourth-order linear dynamical system (2.8). Then,[‖]

$$\partial_{z_i}E = J(t_{i+1}),\quad \partial_{z_i}\dot{E} = \dot{J}(t_{i+1}),\quad \partial_{z_i}\ddot{E} = \ddot{J}(t_{i+1}),\quad \partial_{z_i}\dddot{E} = \dddot{J}(t_{i+1}), \tag{2.15}$$

where the Jacobi field J is obtained by numerically solving (2.8) with the initial conditions $J(t_i) = \alpha$, $\dot{J}(t_i) = \ddot{J}(t_i) = \dddot{J}(t_i) = 0$.

Let e denote the basis of the tangent space $T_{z_i}M$. In abstract notation, if we iteratively choose $(J(t_i), \dot{J}(t_i), \ddot{J}(t_i), \dddot{J}(t_i))^\top$ as $(e,0,0,0)^\top$, $(0,e,0,0)^\top$, $(0,0,e,0)^\top$, and $(0,0,0,e)^\top$, then we get the following derivatives,

$$\begin{bmatrix} J(t_i) \\ \dot{J}(t_i) \\ \ddot{J}(t_i) \\ \dddot{J}(t_i) \end{bmatrix} = \begin{bmatrix} e & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & e & 0 \\ 0 & 0 & 0 & e \end{bmatrix} \xrightarrow{\text{Solve (2.8)}} \begin{bmatrix} J(t_{i+1}) \\ \dot{J}(t_{i+1}) \\ \ddot{J}(t_{i+1}) \\ \dddot{J}(t_{i+1}) \end{bmatrix} = \begin{bmatrix} \partial_{z_i}E & \partial_{v_i}E & \partial_{u_i}E & \partial_{w_i}E \\ \partial_{z_i}\dot{E} & \partial_{v_i}\dot{E} & \partial_{u_i}\dot{E} & \partial_{w_i}\dot{E} \\ \partial_{z_i}\ddot{E} & \partial_{v_i}\ddot{E} & \partial_{u_i}\ddot{E} & \partial_{w_i}\ddot{E} \\ \partial_{z_i}\dddot{E} & \partial_{v_i}\dddot{E} & \partial_{u_i}\dddot{E} & \partial_{w_i}\dddot{E} \end{bmatrix}.$$

Let us define, for notational convenience, the following matrices:

$$\begin{aligned}
A &:= \begin{bmatrix} \partial_{z_i}E & \partial_{v_i}E \\ \partial_{z_i}\dot{E} & \partial_{v_i}\dot{E} \end{bmatrix},\quad B := \begin{bmatrix} \partial_{u_i}E & \partial_{w_i}E \\ \partial_{u_i}\dot{E} & \partial_{w_i}\dot{E} \end{bmatrix},\\
C &:= \begin{bmatrix} \partial_{z_i}\ddot{E} & \partial_{v_i}\ddot{E} \\ \partial_{z_i}\dddot{E} & \partial_{v_i}\dddot{E} \end{bmatrix},\quad D := \begin{bmatrix} \partial_{u_i}\ddot{E} & \partial_{w_i}\ddot{E} \\ \partial_{u_i}\dddot{E} & \partial_{w_i}\dddot{E} \end{bmatrix}.
\end{aligned}$$

[‖]Note that $\partial_\epsilon E(t_{i+1})|_{\epsilon=0} = \partial_{z_i}E(t_{i+1})$.

Then, combining the above with (2.11)–(2.14), we readily obtain

$$\begin{aligned}&\begin{bmatrix}\frac{\partial u_i}{\partial z_i} & \frac{\partial u_i}{\partial v_i}\\ \frac{\partial w_i}{\partial z_i} & \frac{\partial w_i}{\partial v_i}\end{bmatrix} = -B^{-1}A, \quad \begin{bmatrix}\frac{\partial u_i}{\partial z_{i+1}} & \frac{\partial u_i}{\partial v_{i+1}}\\ \frac{\partial w_i}{\partial z_{i+1}} & \frac{\partial w_i}{\partial v_{i+1}}\end{bmatrix} = B^{-1},\\ &\begin{bmatrix}\frac{\partial u_{i+1}}{\partial z_i} & \frac{\partial u_{i+1}}{\partial v_i}\\ \frac{\partial w_{i+1}}{\partial z_i} & \frac{\partial w_{i+1}}{\partial v_i}\end{bmatrix} = C - DB^{-1}A, \quad \begin{bmatrix}\frac{\partial u_{i+1}}{\partial z_{i+1}} & \frac{\partial u_{i+1}}{\partial v_{i+1}}\\ \frac{\partial w_{i+1}}{\partial z_{i+1}} & \frac{\partial w_{i+1}}{\partial v_{i+1}}\end{bmatrix} = DB^{-1}\end{aligned} \tag{2.16}$$

To convert the derivatives above as Riemannian gradients of $\nabla_t \dot{x}$ and $\nabla_t^2 \dot{x}$, we can use the Christoffel symbol and the projection of vectors to the tangent space of the manifold. For example, the Riemannian gradient of $\nabla_t \dot{x}(t_i^+)$ with respect to z_i is given by

$$\mathcal{P}\left(\frac{\partial}{\partial z_i}\nabla_t \dot{x}(t_i^+)\right) = \mathcal{P}\left(\frac{\partial}{\partial z_i}\left(u_i + \Gamma_{z_i}(v_i, v_i)\right)\right) = \mathcal{P}\left(\frac{\partial u_i}{\partial z_i} + d\Gamma_{z_i}(e)(v_i, v_i)\right).$$

The Riemannian gradient of $\nabla_t^2 \dot{x}(t_i^+)$ with respect to z_i is given by

$$\begin{aligned}\mathcal{P}\left(\frac{\partial}{\partial z_i}\nabla_t^2 \dot{x}(t_i^+)\right) =& \mathcal{P}\left(\frac{\partial}{\partial z_i}\left(w_i + 3\Gamma_{z_i}(v_i, u_i) + d\Gamma_{z_i}(v_i)(v_i, v_i) + \Gamma_{z_i}(v_i, \Gamma_{z_i}(v_i, v_i))\right)\right)\\ =& \mathcal{P}\left(\frac{\partial w_i}{\partial z_i} + 3d\Gamma_{z_i}(e)(v_i, \ddot{x}_i) + 3\Gamma_{z_i}\left(v_i, \frac{\partial u_i}{\partial z_i}\right) + d^2\Gamma_{z_i}(v_i, e)(v_i, v_i)\right.\\ &\left. + d\Gamma_{z_i}(e)(v_i, \Gamma_{z_i}(v_i, v_i)) + \Gamma_{z_i}(v_i, d\Gamma_{z_i}(e)(v_i, v_i))\right),\end{aligned}$$

where $\mathcal{P}$ is the projection of vectors to the tangent space of the manifold. See Appendix 5 for a full list of the Riemannian gradients of $\nabla_t \dot{x}^{(i)}(t_i^+)$, $\nabla_t^2 \dot{x}^{(i)}(t_i^+)$, $\nabla_t \dot{x}^{(i)}(t_{i+1}^-)$, and $\nabla_t^2 \dot{x}^{(i)}(t_{i+1}^-)$ with respect to z_i, v_i, z_{i+1}, and v_{i+1}.

This section has built the two key blocks for the proposed method, i.e., finding the good initial guesses for solving the Riemannian cubic equation (1.2) by the single shooting method, and deriving the derivatives of covariant accelerations and jerks. In what follows, we will put all "ingredients" together to show the general framework for finding Riemannian cubics and cubic splines.

3. General framework for determining Riemannian cubics and cubic splines

In this section, we present the general framework for determining Riemannian cubics and cubic splines using Riemannian-gradient-based methods, such as Riemannian gradient descent, Riemannian conjugate gradient descent [26, 27], and Riemannian limited-memory BFGS [28, 29].

Given boundary conditions (x_0, v_0) (at $t = 0$) and (x_T, v_T) (at $t = T$), let $(z_i, v_i) \in TM$ be the junction and its associated velocity at time t_i, $i = 1, 2, \ldots, n-1$. We set $z_0 := x_0$, $z_n := x_T$, $v_n := v_T$, $\mathbf{z} := (z_1, z_2, \ldots, z_{n-1})$, $\mathbf{v} := (v_1, v_2, \ldots, v_{n-1})$, and $\nabla_t^j \dot{x}(t_0^+) = \nabla_t^j \dot{x}(t_n^-) = \mathbf{0}$, $j = 1, 2$. For simplicity, we may divide the total time interval $[0, T]$ equally, that is, $t_i = \frac{i}{n}T$, $i = 0, 1, \ldots, n$. Then, the Euclidean gradients of the function (1.6) with respect to $\mathbf{z}$ and $\mathbf{v}$ are given as follows:

$$\partial_{\mathbf{z}} f = (\partial_{z_1} f, \partial_{z_2} f, \ldots, \partial_{z_{n-1}} f), \quad \partial_{\mathbf{v}} f = (\partial_{v_1} f, \partial_{v_2} f, \ldots, \partial_{v_{n-1}} f), \tag{3.1}$$

where

$$\begin{aligned}\partial_{z_i} f =& \left(\frac{\partial \alpha_i}{\partial z_i}\right)^{\dagger}(\alpha_i) + \left(\frac{\partial}{\partial z_i}(\nabla_t \dot{x}(t_{i-1}^{+}))\right)^{\dagger}(\alpha_{i-1}) - \left(\frac{\partial}{\partial z_i}(\nabla_t \dot{x}(t_{i+1}^{-}))\right)^{\dagger}(\alpha_{i+1}) \\ &+ \left(\frac{\partial \beta_i}{\partial z_i}\right)^{\dagger}(\beta_i) + \left(\frac{\partial}{\partial z_i}(\nabla_t^2 \dot{x}(t_{i-1}^{+}))\right)^{\dagger}(\beta_{i-1}) - \left(\frac{\partial}{\partial z_i}(\nabla_t^2 \dot{x}(t_{i+1}^{-}))\right)^{\dagger}(\beta_{i+1}),\end{aligned} \tag{3.2}$$

and

$$\begin{aligned}\partial_{v_i} f =& \left(\frac{\partial \alpha_i}{\partial v_i}\right)^{\dagger}(\alpha_i) + \left(\frac{\partial}{\partial v_i}(\nabla_t \dot{x}(t_{i-1}^{+}))\right)^{\dagger}(\alpha_{i-1}) - \left(\frac{\partial}{\partial v_i}(\nabla_t \dot{x}(t_{i+1}^{-}))\right)^{\dagger}(\alpha_{i+1}) \\ &+ \left(\frac{\partial \beta_i}{\partial v_i}\right)^{\dagger}(\beta_i) + \left(\frac{\partial}{\partial v_i}(\nabla_t^2 \dot{x}(t_{i-1}^{+}))\right)^{\dagger}(\beta_{i-1}) - \left(\frac{\partial}{\partial v_i}(\nabla_t^2 \dot{x}(t_{i+1}^{-}))\right)^{\dagger}(\beta_{i+1}),\end{aligned} \tag{3.3}$$

with $\alpha_i := \nabla_t \dot{x}(t_i^{+}) - \nabla_t \dot{x}(t_i^{-})$, $\beta_i := \nabla_t^2 \dot{x}(t_i^{+}) - \nabla_t^2 \dot{x}(t_i^{-})$, and the symbol † denotes the adjoint operator defined by

$$\left\langle \mu, \frac{\partial \omega}{\partial \nu}(\delta) \right\rangle = \left\langle \left(\frac{\partial \omega}{\partial \nu}\right)^{\dagger}(\mu), \delta \right\rangle.$$

Since the change of z_i affects the values of $\nabla_t^k \dot{x}(t_i^{+})$, $\nabla_t^k \dot{x}(t_i^{-})$, $\nabla_t^k \dot{x}(t_{i-1}^{+})$, and $\nabla_t^k \dot{x}(t_{i+1}^{-})$, $k = 1, 2$, (3.2) follows from the following fact:

$$\begin{aligned}\langle \partial_{z_i} f, w \rangle =& \left\langle \alpha_i, \frac{\partial \alpha_i}{\partial z_i}(w) \right\rangle + \left\langle \alpha_{i-1}, \frac{\partial \alpha_{i-1}}{\partial z_i}(w) \right\rangle + \left\langle \alpha_{i+1}, \frac{\partial \alpha_{i+1}}{\partial z_i}(w) \right\rangle + \\ & \left\langle \beta_i, \frac{\partial \beta_i}{\partial z_i}(w) \right\rangle + \left\langle \beta_{i-1}, \frac{\partial \beta_{i-1}}{\partial z_i}(w) \right\rangle + \left\langle \beta_{i+1}, \frac{\partial \beta_{i+1}}{\partial z_i}(w) \right\rangle \\ =& \left\langle \left(\frac{\partial \alpha_i}{\partial z_i}\right)^{\dagger}(\alpha_i) + \left(\frac{\partial}{\partial z_i}(\nabla_t \dot{x}(t_{i-1}^{+}))\right)^{\dagger}(\alpha_{i-1}) - \left(\frac{\partial}{\partial z_i}(\nabla_t \dot{x}(t_{i+1}^{-}))\right)^{\dagger}(\alpha_{i+1}) + \right. \\ & \left. \left(\frac{\partial \beta_i}{\partial z_i}\right)^{\dagger}(\beta_i) + \left(\frac{\partial}{\partial z_i}(\nabla_t^2 \dot{x}(t_{i-1}^{+}))\right)^{\dagger}(\beta_{i-1}) - \left(\frac{\partial}{\partial z_i}(\nabla_t^2 \dot{x}(t_{i+1}^{-}))\right)^{\dagger}(\beta_{i+1}), w \right\rangle.\end{aligned}$$

Likewise, $\partial_{v_i} f$ is similar to calculate. Then, we convert the Euclidean gradients $\partial_{\mathbf{z}} f$ and $\partial_{\mathbf{v}} f$ to the Riemannian gradients $\partial_{\mathbf{z}}^R f$ and $\partial_{\mathbf{v}}^R f$ by the projection $\mathcal{P}$ to the tangent bundle of M, namely,

$$\partial_{\mathbf{z}}^R f := (\mathcal{P}(\partial_{z_1} f), \mathcal{P}(\partial_{z_2} f), \ldots, \mathcal{P}(\partial_{z_{n-1}} f)), \quad \partial_{\mathbf{v}}^R f := (\mathcal{P}(\partial_{v_1} f), \mathcal{P}(\partial_{v_2} f), \ldots, \mathcal{P}(\partial_{v_{n-1}} f)). \tag{3.4}$$

Now we have all the "ingredients" needed to perform optimization with the Riemannian gradient-based algorithms; our proposed method for approximating a Riemannian cubic joining (x_0, v_0) (at $t = 0$) and (x_T, v_T) (at $t = T$) is summarized as Algorithm 1.

Algorithm 1: Riemannian-gradient-based method for determining Riemannian cubics

Input: Boundary conditions $(x_0, v_0), (x_T, v_T) \in TM$, number of sub-intervals n, maximum number of iterations N, stop error $\epsilon > 0$.

Output: Riemannian cubic $\gamma : [0, T] \to M$ with $\gamma(0) = x_0$, $\dot{\gamma}(0) = v_0$, $\gamma(T) = x_T$ and $\dot{\gamma}(T) = v_T$.

Choose an arbitrary curve $\tilde{\gamma} : [0, T] \to M$ joining (x_0, v_0) and (x_T, v_T), and $z_i := \tilde{\gamma}(iT/n)$, $v_i := \tilde{\gamma}(iT/n)$, $i = 1, 2, \cdots, n-1$.

Replace $\tilde{\gamma}$ by a piecewise Riemannian cubic γ joining all (z_i, v_i)'s via the single shooting method.

for $k \leftarrow 1$ **to** N **do**

  for $i \leftarrow 1$ **to** $n-1$ **do**

    Calculate the Euclidean gradient of (1.6) via (3.1).

    Calculate the Riemannian gradient of (1.6) via (3.4).

    Update z_i, v_i via Riemannian gradient based method with the cost function (1.6) and Riemannian gradients (3.4). // Manifold optimization package Manopt can be used as the backbone.

  end for

  if $\|\partial^R_{\mathbf{z}} f\| \leqslant \epsilon$ **and** $\|\partial^R_{\mathbf{v}} f\| \leqslant \epsilon$ **or** $f \leqslant \epsilon$ **then**

    Update the piecewise Riemannian cubic γ based on new (z_i, v_i)'s.

    Stop.

  end if

end for

return γ

Now we turn to the general framework for finding Riemannian cubic splines. The input data are boundary data (x_0, v_0) (at $t = 0$) and (x_T, v_T) (at $t = T$), and interior data $x_1, x_2, \ldots, x_{l-1}$ (at $t = t_1, t_2, \ldots, t_{l-1}$, respectively). Let $v_i \in T_{x_i}M$ denote the tangent vector at x_i, $i = 1, 2, \ldots, l-1$. We set $t_0 := 0$, $t_l := T$, $x_l := x_T$, $v_l := v_T$, $\mathbf{v} := (v_1, v_2, \ldots, v_{l-1})$, $\nabla_t \dot{x}(t_0^+) := \mathbf{0}$, and $\nabla_t \dot{x}(t_l^-) = \mathbf{0}$.

For $0 \leq i \leq l-1$, suppose the Riemannian cubic joining (x_i, v_i) and (x_{i+1}, v_{i+1}) has been approximated by Algorithm 1 or other methods. Similarly as before, the Riemannian gradient of the cost function (1.7) with respect to $\mathbf{v}$ is given as follows:

$$\partial^R_{\mathbf{v}} g = (\mathcal{P}(\partial_{v_1} g), \mathcal{P}(\partial_{v_2} g), \ldots, \mathcal{P}(\partial_{v_{l-1}} g)), \tag{3.5}$$

where

$$\partial_{v_i} g = \left(\frac{\partial \alpha_i}{\partial v_i}\right)^{\dagger} (\alpha_i) + \left(\frac{\partial}{\partial v_i}(\nabla_t \dot{x}(t_{i-1}^+))\right)^{\dagger} (\alpha_{i-1}) - \left(\frac{\partial}{\partial v_i}(\nabla_t \dot{x}(t_{i+1}^-))\right)^{\dagger} (\alpha_{i+1}), \tag{3.6}$$

with $\alpha_i := \nabla_t \dot{x}(t_i^+) - \nabla_t \dot{x}(t_i^-)$.

The Riemannian-gradient-based method for approximating a Riemannian cubic spline joining (x_0, v_0) (at $t = 0$), $x_1, x_2, \ldots, x_{k-1}$ (at $t = t_1, t_2, \ldots, t_{k-1}$), and (x_T, v_T) (at $t = T$) is summarized in Algorithm 2.

Algorithm 2: Riemannian-gradient-based method for determining Riemannian cubic splines

Input: Boundary conditions $(x_0, v_0), (x_T, v_T) \in TM$, interior conditions $(t_1, x_1), (t_2, x_2), \ldots, (t_{l-1}, x_{l-1})$, maximum number of iterations N, stop error $\epsilon > 0$.

Output: Riemannian cubic spline $\gamma : [0, T] \to M$ with $\gamma(0) = x_0$, $\dot{\gamma}(0) = v_0$, $\gamma(T) = x_T$, $\dot{\gamma}(T) = v_T$, $\gamma(t_i) = x_i$, $i = 1, 2, \ldots, l-1$.

```
Choose initial velocities v_i ∈ T_{x_i}M and find the Riemannian cubic curve joining (x_i, v_i) and (x_{i+1}, v_{i+1}).
for k ← 1 to N do
    for i ← 1 to n − 1 do
        Calculate the Riemannian gradient of (1.7) via (3.5).
        Update v_i via Riemannian gradient based method with the cost function (1.7) and Riemannian gradients (3.5). // Manifold optimization package Manopt can be used as the backbone.
    end for
    if ||∂^R_v g|| ⩽ ϵ or g ⩽ ϵ then
        Calculate the piecewise Riemannian cubic spline γ based on new (x_i, v_i)'s.
        Stop.
    end if
end for
return γ
```

4. Numerical experiments on the unit sphere

To test the feasibility and efficiency of our proposed methods, this section focuses on determining Riemannian cubics and cubic splines on the unit sphere. We implemented all experiments in MATLAB® 2023a on a computer with an AMD 3600 6-core processor, 8GB RAM, and Windows 10 Enterprise. We adopt Riemannian gradient descent (RGD), Riemannian conjugate gradient descent (RGD), and Riemannian limited-memory BFGS (RLBFGS) in the well known manifold optimization package Manopt as the backbone of our methods [23]. MATLAB® codes are available on request for reasonable purposes.

Let $\mathbb{S}^m \subset \mathbb{E}^{m+1}$ denote the unit sphere endowed with the induced standard Euclidean metric. For $p \in \mathbb{S}^m$, $u_1, u_2, u_3 \in T_p\mathbb{S}^m$, the Levi-Civita connection is given by

$$\nabla_{u_1} u_2 = \partial_{u_1} u_2 + \langle u_1, u_2 \rangle p,$$

and the Riemannian curvature is written as

$$R(u_1, u_2)u_3 = \langle u_2, u_3 \rangle u_1 - \langle u_1, u_3 \rangle u_2.$$

Then, the Riemannian cubic equation (1.2) on the sphere can be reduced to (see also Eq (51) in [31])

$$\begin{aligned} x^{(4)}(t) + 4\langle \dot{x}(t), \dddot{x}(t) \rangle x(t) + 3\langle \ddot{x}(t), \ddot{x}(t) \rangle x(t) + 4\langle \dot{x}(t), \ddot{x}(t) \rangle \dot{x}(t) \\ + 2\langle \dot{x}(t), \dot{x}(t) \rangle \ddot{x}(t) + 2\langle \dot{x}(t), \dot{x}(t) \rangle^2 x(t) = 0. \end{aligned} \tag{4.1}$$

The Jacobi equation (2.8) on the sphere takes the form

$$\begin{aligned} J^{(4)}(t) &+ 4\langle \dddot{J}(t), \dot{x}(t)\rangle x(t) + 6\langle \ddot{J}(t), \ddot{x}(t)\rangle x(t) + 4\langle \ddot{J}(t), \dot{x}(t)\rangle \dot{x}(t) \\ &+ 2\langle \dot{x}(t), \dot{x}(t)\rangle \ddot{J}(t) + 4\langle \dot{J}(t), \dddot{x}(t)\rangle x(t) + 4\langle \dot{J}(t), \ddot{x}(t)\rangle \dot{x}(t) \\ &+ 4\langle \ddot{x}(t), \dot{x}(t)\rangle \dot{J}(t) + 4\langle \dot{J}(t), \dot{x}(t)\rangle \ddot{x}(t) + 8\langle \dot{J}(t), \dot{x}(t)\rangle \langle \dot{x}(t), \dot{x}(t)\rangle x(t) \\ &+ 4\langle \dddot{x}(t), \dot{x}(t)\rangle J(t) + 3\langle \ddot{x}(t), \ddot{x}(t)\rangle J(t) + 2\langle \dot{x}(t), \dot{x}(t)\rangle^2 J(t) = 0. \end{aligned} \tag{4.2}$$

Example 2. To generate a reference Riemannian cubic on the sphere $\mathbb{S}^2$, we choose initial conditions $x(0) = (1, 0, 0)^\top$, $\dot{x}(0) = (0, 0.2, -0.2)^\top$, $\ddot{x}(0) = (-0.0800, 0.1180, 0.0429)^\top$, $\dddot{x}(0) = (-0.0451, -0.0572, 0.0300)^\top$, time $T = 7$, and solve the equation (4.1) by the MATLAB® method ODE45. Then, we read the endpoint $x(T) = (0.0023, 1.0000, -9.2229 \times 10^{-4})^\top$ and end-velocity $\dot{x}(T) = (0.1675, -3.9341 \times 10^{-4}, -3.1805 \times 10^{-4})^\top$ from the numerically solved Riemannian cubic.

Now we turn to approximating the Riemannian cubic joining the boundary conditions $(x(0), \dot{x}(0))$ and $(x(T), \dot{x}(T))$ on $[0, T]$ by the proposed methods. To attain such an approximation, we set the number of segments $n = 3$. We first join $(x(0), \dot{x}(0))$ and $(x(T), \dot{x}(T))$ by the Euclidean cubic polynomial $\tilde{\gamma}$, then choose (z_i, v_i) as the initial interior conditions, where

$$z_i = \frac{\tilde{\gamma}(iT/n)}{\|\tilde{\gamma}(iT/n)\|}, \quad v_i = \dot{\tilde{\gamma}}(iT/n) - \langle \dot{\tilde{\gamma}}(iT/n), z_i\rangle z_i, \quad i = 1, 2. \tag{4.3}$$

Instead of minimizing the cost function (1.6) on the 2-fold tangent bundle**, we set out to alternatively minimize (1.6) over the 2-fold manifold $\mathbb{S}^2 \times \mathbb{S}^2$ and 2-fold tangent space $T_{v_1}\mathbb{S}^2 \times T_{v_2}\mathbb{S}^2$, where the manifold factories in the MANOPT package are directly applicable [23]. Figure 1 shows some iteration statistics including the cost function value, the norm of the gradient of the cost function, and the running time for RGD, RCG, and RLBFGS.

We observe that RCG stops after 40 iterations while RGD and RLBFGS stop after 156 iterations though the stopping tolerances on the cost function value (10^{-8}) and on the gradient norm (10^{-8}), and the minimal stepsize (10^{-10}) are set to be the same for all of them. Roughly speaking, RLBFGS generates the Riemannian cubic with the smallest error but RCG runs faster. Besides, the process of minimizing over $T_{v_1}\mathbb{S}^2 \times T_{v_2}\mathbb{S}^2$ is generally faster than that of minimizing over $\mathbb{S}^2 \times \mathbb{S}^2$ as can be readily inferred from the bottom panel†† in Figure 1.

**The cost function (1.6) is defined on the $(n-1)$-fold tangent bundle TM. Since all the manifold factories in the MANOPT package are implemented on the manifolds, not on the tangent bundles, we minimize the cost function on the $(n-1)$-fold manifolds and the $(n-1)$-fold tangent spaces separately and alternatively. Mathematically, the optimization problem $\min_{\{(z_i,v_i)\}_{i=1}^{n-1} \in (TM)^{n-1}} f(\{(z_i, v_i)\}_{i=1}^{n-1})$ is reformulated as $\min_{\{(z_i)\}_{i=1}^{n-1} \in M^{n-1}} f(\{(z_i, v_i)\}_{i=1}^{n-1})$ with v_i fixed and $\min_{\{(v_i)\}_{i=1}^{n-1} \in (T_{z_i}M)^{n-1}} f(\{(z_i, v_i)\}_{i=1}^{n-1})$ with z_i fixed.

††Note the tangent space $T_{v_1}\mathbb{S}^2 \times T_{v_2}\mathbb{S}^2$ is flat, which might make numerical optimization easier.

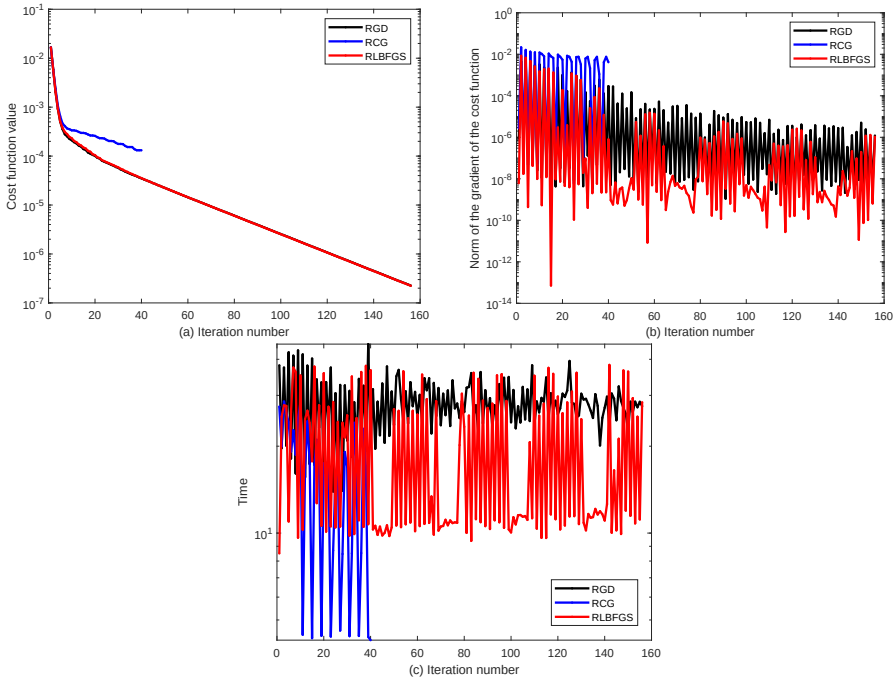

Figure 1. An example of determining Riemannian cubics on the sphere $\mathbb{S}^2$ using Algorithm 1 with RGD, RCG, and RLBFGS as backbones: (a) Convergence behavior of the cost function value; (b) Convergence behavior of the norm of the gradient of the cost function; (c) Running time in each iteration.

The qualitative performance of RLBFGS for finding Riemannian cubics is shown in Figure 2, where the blue, red, and green curves represent the projection of Euclidean cubic polynomial, the piecewise Riemannian cubic generated by Algorithm 1 with RLBFGS, and the reference Riemannian cubic, respectively. The red curve almost coincides with the green curve, which demonstrates the performance of our proposed methods.

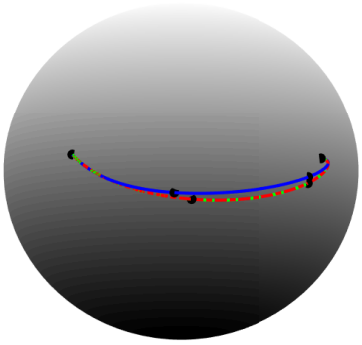

Figure 2. Qualitative result of our methods for determining Riemannian cubics, blue, red, and green curves represent the projection of Euclidean cubic polynomial, the piecewise Riemannian cubic generated by Algorithm 1 with RLBFGS, and the reference Riemannian cubic, respectively.

Example 3. In this example, we consider determining a Riemannian cubic spline joining

$$(x_0, v_0) = ((1,0,0)^\top, (0, 0.2, -0.2)^\top) \ \ (\text{at } t = 0)$$

and

$$(x_T, v_T) = ((0,1,0)^\top, (0.1675, 0, 0)^\top) \ \ (\text{at } t = 7),$$

and passing through

$$x_1 = (0.2598, 0.9496, -0.1753)^\top \ \text{ at } \ t = 2.3333,$$
$$x_2 = (0.7081, 0.6588, -0.2541)^\top \ \text{ at } \ t = 4.6667.$$

The computer code is initialized by selecting the tentative initial velocities

$$v_1 = (-0.0584, -0.0137, -0.1605)^\top \in T_{x_1}\mathbb{S}^2,$$
$$v_2 = (0.0835, -0.0781, 0.0301)^\top \in T_{x_2}\mathbb{S}^2.$$

Then, the code proceeds to minimize the cost function (1.7) using Algorithm 2, whose results are shown in Figure 3. Under the same stopping tolerances on the cost value (10^{-8}) and on the gradient norm (10^{-8}), and minimal stepsize (10^{-10}), the RGD takes 13 iterations while the RLBFGS only needs 6 iterations. The running time of RGD, RCG, and RLBFGS are 23.0278 seconds, 13.2719 seconds, 6.9692 seconds, respectively.

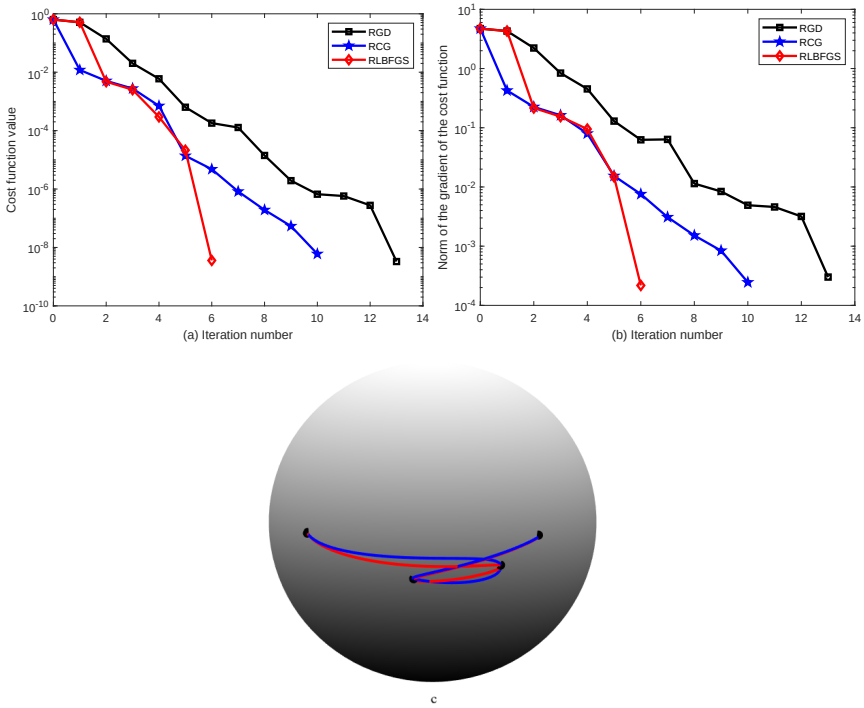

Figure 3. An example of determining a Riemannian cubic spline on the unit sphere $\mathbb{S}^2$ using Algorithm 2 with RGD, RCG and RLBFGS as backbones: (a) Convergence behavior of the cost function value; (b) Convergence behavior of the norm of the gradient of the cost function; (c) Blue and red curves represent initial piecewise Riemannian cubic spline, and final piecewise Riemannian cubic spline generated by Algorithm 2 with RLBFGS as the backbone.

5. Conclusions

As generalizations of cubic polynomials in Euclidean spaces, Riemannian cubics and their variants Riemannian cubic splines have been widely used in engineering, computer science and applied mathematics. However, effectively constructing Riemannian cubics and cubic splines is a long-standing and hard problem. This paper proposes new frameworks for constructing Riemannian cubics, based on the variational principle and the shooting method. By adding some interior conditions, piecewise Riemannian cubics joining consecutive conditions can be found by shooting method. Then, we minimize the difference of covariant accelerations, second-order covariant accelerations at junctions by Riemannian-gradient-based methods such as Riemannian gradient descent, Riemannian conjugate gradient descent, and Riemannian limited-memory BFGS. Based on the obtained Riemannian cubics, Riemannian cubic splines can similarly be approximated by minimizing the difference of covariant accelerations at interior points by gradient methods. Numerical experiments on the unit sphere $\mathbb{S}^2$ test the effectiveness and efficiency of the proposed methods.

The main novelty of the proposed methods is that we take advantage of the variational principle and geometric properties of cubic polynomial (for finding local Riemannian cubics). Although numerical experiments show our methods are efficient for determining Riemannian cubics and cubic splines, the efficiency could be further improved if more efficient Riemannian gradient-based methods could be used, which could be our future work. We underline that alternatively minimizing the cost function (1.6) on manifolds and tangent spaces may reduce the efficiency of the proposed methods. Future work may include developing the tangent bundle factory in the Manopt package for such kind of optimization problems. Further, the convergence analysis of Riemannian gradient descent, Riemannian conjugate gradient method and Riemannian BFGS for general or specific tasks has been studied in the literature, such as [24, 26–28]. Motivated by the existing theoretical analysis, we may consider the convergence of our proposed algorithms in the future work.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This research activity was supported by the National Natural Science Foundation of China (Grant No. 12401504). The authors wish to gratefully thank the anonymous reviewers whose invaluable comments and suggestions helped improving the quality of this presentation.

Conflict of interest

Simone Fiori is a guest editor for Electronic Research Archive and was not involved in the editorial review or the decision to publish this article. The authors declare there is no conflict of interest.

References

1. S. A. Gabriel, J. T. Kajiya, Spline interpolation in curved space, State of the Art in Image Synthesis, SIGGRAPH'85 Course Notes, 1–14, 1985. Available from: https://archive.computerhistory.org/resources/access/text/2023/05/102724881-05-11-acc.pdf?utm_source
2. L. Noakes, G. Heinzinger, B. Paden, Cubic splines on curved spaces, *IMA J. Math. Control Inf.*, **6** (1989), 465–473. https://doi.org/10.1093/imamci/6.4.465
3. D. C. Brody, D. D. Holm, D. M. Meier, Quantum splines, *Phys. Rev. Lett.*, **109** (2012), 100501. https://doi.org/10.1103/PhysRevLett.109.100501
4. K. Kim, I. L. Dryden, H. Le, K. E. Severn, Smoothing splines on Riemannian manifolds, with applications to 3D shape space, *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **83** (2021), 108–132. https://doi.org/10.1111/rssb.12402
5. N. Singh, F. Vialard, M. Niethammer, Splines for diffeomorphisms, *Med. Image Anal.*, **25** (2015), 56–71. https://doi.org/10.1016/j.media.2015.04.012
6. P. T. Fletcher, Geodesic regression and the theory of least squares on Riemannian manifolds, *Int. J. Comput. Vision*, **105** (2013), 171–185. https://doi.org/10.1007/s11263-012-0591-y
7. J. Aubray, F. Nicol, Polynomial regression on Lie groups and application to SE(3), *Entropy*, **26** (2004), 825. https://doi.org/10.3390/e26100825
8. M. Hanik, E. Nava-Yazdani, C. von Tycowicz, De Casteljau's algorithm in geometric data analysis: Theory and application, *Comput. Aided Geom. Des.*, **110** (2024), 102288. https://doi.org/10.1016/j.cagd.2024.102288
9. E. Zhang, L. Noakes, The cubic de Casteljau construction and Riemannian cubics, *Comput. Aided Geom. Des.*, **75** (2019), 101789. https://doi.org/10.1016/j.cagd.2019.101789
10. J. Hinkle, P. Muralidharan, P. T. Fletcher, S. Joshi, Polynomial regression on Riemannian manifolds, in *Computer Vision-ECCV 2012*, **7574** (2012), 1–14. https://doi.org/10.1007/978-3-642-33712-3_1
11. C. Belta, V. Kumar, An SVD-based projection method for interpolation on SE(3), *IEEE Trans. Rob. Autom.*, **18** (2002), 334–345. https://doi.org/10.1109/TRA.2002.1019463
12. P. S. V. S. S. Kumar, R. Padhi, Minimum jerk based guidance for autonomous soft-landing of quadrotor, *IFAC-PapersOnLine*, **57** (2024), 107–112. https://doi.org/10.1016/j.ifacol.2024.05.019
13. S. Fiori, A coordinate-free variational approach to fourth-order dynamical systems on manifolds: A system and control theoretic viewpoint, *Mathematics*, **12** (2024), 428. https://doi.org/10.3390/math12030428
14. M. Camarinha, F. Silva Leite, P. Crouch, Existence and uniqueness for Riemannian cubics with boundary conditions, in *CONTROLO 2020*, **695** (2020), 322–331. https://doi.org/10.1007/978-3-030-58653-9_31
15. M. Camarinha, F. Silva Leite, P. Crouch, Riemannian cubics close to geodesics at the boundaries, *J. Geom. Mech.*, **14** (2022), 545–558. https://doi.org/10.3934/jgm.2022003
16. R. Giambò, F. Giannoni, P. Piccione, Optimal control on Riemannian manifolds by interpolation, *Math. Control Signals Syst.*, **16** (2004), 278–296. https://doi.org/10.1007/s00498-003-0139-3

17. H. B. Keller, *Numerical Methods for Two-Point Boundary-Value Problems*, Dover Publications, 2018.

18. L. Noakes, Approximating near-geodesic natural cubic splines, *Commun. Math. Sci.*, **12** (2014), 1409–1425. https://doi.org/10.4310/CMS.2014.v12.n8.a2

19. P. Balseiro, T. J. Stuchi, A. Cabrera, J. Koiller, About simple variational splines from the Hamiltonian viewpoint, *J. Geom. Mech.*, **9** (2017), 257–290. https://doi.org/10.3934/jgm.2017011

20. B. Heeren, M. Rumpf, B. Wirth, Variational time discretization of Riemannian splines, *IMA J. Numer. Anal.*, **39** (2019), 61–104. https://doi.org/10.1093/imanum/drx077

21. L. Machado, F. Silva Leite, K. Krakowski, Higher-order smoothing splines versus least squares problems on Riemannian manifolds, *J. Dyn. Control Syst.*, **16** (2010), 121–148. https://doi.org/10.1007/s10883-010-9080-1

22. T. T. Tran, I. Adouani, C. Samir, Computing regularized splines in the Riemannian manifold of probability measures, *Math. Modell. Numer. Anal.*, **59** (2025), 73–99. https://doi.org/10.1051/m2an/2024056

23. N. Boumal, B. Mishra, P. Absil, R. Sepulchre, Manopt, a Matlab toolbox for optimization on manifolds, *J. Mach. Learn. Res.*, **15** (2014), 1455–1459,

24. B. Afsari, R. Tron, R. Vidal, On the convergence of gradient descent for finding the Riemannian center of mass, *SIAM J. Control Optim.*, **51** (2013), 2230–2260. https://doi.org/10.1137/12086282X

25. C. Samir, P. Absil, A. Srivastava, E. Klassen, A gradient-descent method for curve fitting on Riemannian manifolds, *Found. Comput. Math.*, **12** (2012), 49–73. https://doi.org/10.1007/s10208-011-9091-7

26. X. Zhu, A Riemannian conjugate gradient method for optimization on the Stiefel manifold, *Comput. Optim. Appl.*, **67** (2017), 73–110. https://doi.org/10.1007/s10589-016-9883-4

27. H. Sato, Riemannian conjugate gradient methods: General framework and specific algorithms with convergence analyses, *SIAM J. Optim.*, **32** (2022), 2690–2717. https://doi.org/10.1137/21M1464178

28. W. Huang, P. Absil, K. A. Gallivan, A Riemannian BFGS method for nonconvex optimization problems, in *Numerical Mathematics and Advanced Applications ENUMATH 2015*, **112** (2016), 627–634. https://doi.org/10.1007/978-3-319-39929-4_60

29. X. Yuan, W. Huang, P. Absil, K. A. Gallivan, A Riemannian limited-memory BFGS algorithm for computing the matrix geometric mean, *Proc. Comput. Sci.*, **80** (2016), 2147–2157. https://doi.org/10.1016/j.procs.2016.05.534

30. M. P. Do Carmo, *Riemannian Geometry*, 1^{st} edition, Birkhäuser, 1992.

31. P. Crouch, F. Silva Leite, The dynamic interpolation problem: on Riemannian manifolds, Lie groups, and symmetric spaces, *J. Dyn. Control Syst.*, **1** (1995), 177–202. https://doi.org/10.1007/BF02254638

32. S. Fiori, Manifold calculus in system theory and control-Fundamentals and first-order systems, *Symmetry*, **13** (2021), 2092. https://doi.org/10.3390/sym13112092

33. S. Fiori, Manifold calculus in system theory and control-Second order structures and systems, *Symmetry*, **14** (2022), 1144. https://doi.org/10.3390/sym14061144

Appendix

Calculations of Riemannian gradients

For $0 \leq i \leq n-1$, let $x^{(i)}$ be the Riemannian cubic joining the boundary conditions (z_i, v_i) (at $t = t_i$) and (z_{i+1}, v_{i+1}) (at $t = t_{i+1}$). This appendix presents a full list of the Riemannian gradients of $\nabla_t \dot{x}^{(i)}(t_i^+)$, $\nabla_t^2 \dot{x}^{(i)}(t_i^+)$, $\nabla_t \dot{x}^{(i)}(t_{i+1}^-)$ and $\nabla_t^2 \dot{x}^{(i)}(t_{i+1}^-)$ with respect to boundary values $z_i, v_i, z_{i+1}, v_{i+1}$.

Let Γ be the Christoffel symbol associated with the Levi-Civita connection ∇, $\mathcal{P}$ the projection of vectors to the tangent space of the Riemannian manifold M. We write u_i, w_i, u_{i+1}, w_{i+1} for $\ddot{x}^{(i)}(t_i^+)$, $\dddot{x}^{(i)}(t_i^+)$, $\ddot{x}^{(i)}(t_{i+1}^-)$, $\dddot{x}^{(i)}(t_{i+1}^-)$, respectively. Then, we have

$$\mathcal{P}\left(\frac{\partial}{\partial z_i}\nabla_t \dot{x}(t_i^+)\right) = \mathcal{P}\left(\frac{\partial u_i}{\partial z_i} + d\Gamma_{z_i}(e)(v_i, v_i)\right),$$
$$\mathcal{P}\left(\frac{\partial}{\partial v_i}\nabla_t \dot{x}(t_i^+)\right) = \mathcal{P}\left(\frac{\partial u_i}{\partial v_i} + 2\Gamma_{z_i}(v_i, e)\right),$$

$$\mathcal{P}\left(\frac{\partial}{\partial z_{i+1}}\nabla_t \dot{x}(t_i^+)\right) = \mathcal{P}\left(\frac{\partial u_i}{\partial z_{i+1}}\right),$$
$$\mathcal{P}\left(\frac{\partial}{\partial v_{i+1}}\nabla_t \dot{x}(t_i^+)\right) = \mathcal{P}\left(\frac{\partial u_i}{\partial v_{i+1}}\right),$$

$$\mathcal{P}\left(\frac{\partial}{\partial z_i}\nabla_t^2 \dot{x}(t_i^+)\right) = \mathcal{P}\left(\frac{\partial w_i}{\partial z_i} + 3d\Gamma_{z_i}(e)(v_i, u_i) + 3\Gamma_{z_i}\left(v_i, \frac{\partial u_i}{\partial z_i}\right) + d^2\Gamma_{z_i}(v_i, e)(v_i, v_i)\right.$$
$$\left. + d\Gamma_{z_i}(e)(v_i, \Gamma_{z_i}(v_i, v_i)) + \Gamma_{z_i}(v_i, d\Gamma_{z_i}(e)(v_i, v_i))\right),$$
$$\mathcal{P}\left(\frac{\partial}{\partial v_i}\nabla_t^2 \dot{x}(t_i^+)\right) = \mathcal{P}\left(\frac{\partial w_i}{\partial v_i} + 3\Gamma_{z_i}(e, u_i) + 3\Gamma_{z_i}\left(v_i, \frac{\partial u_i}{\partial v_i}\right) + d\Gamma_{z_i}(e)(v_i, v_i)\right.$$
$$\left. + 2d\Gamma_{z_i}(v_i)(v_i, e) + \Gamma_{z_i}(e, \Gamma_{z_i}(v_i, v_i)) + 2\Gamma_{z_i}(v_i, \Gamma_{z_i}(v_i, e))\right),$$

$$\mathcal{P}\left(\frac{\partial}{\partial z_{i+1}}\nabla_t^2 \dot{x}(t_i^+)\right) = \mathcal{P}\left(\frac{\partial w_i}{\partial z_{i+1}} + 3\Gamma_{z_i}\left(v_i, \frac{\partial u_i}{\partial z_{i+1}}\right)\right),$$
$$\mathcal{P}\left(\frac{\partial}{\partial v_{i+1}}\nabla_t^2 \dot{x}(t_i^+)\right) = \mathcal{P}\left(\frac{\partial w_i}{\partial v_{i+1}} + 3\Gamma_{z_i}\left(v_i, \frac{\partial u_i}{\partial v_{i+1}}\right)\right),$$

$$\mathcal{P}\left(\frac{\partial}{\partial z_i}\nabla_t \dot{x}(t_{i+1}^-)\right) = \mathcal{P}\left(\frac{\partial u_{i+1}}{\partial z_i}\right),$$
$$\mathcal{P}\left(\frac{\partial}{\partial v_i}\nabla_t \dot{x}(t_{i+1}^-)\right) = \mathcal{P}\left(\frac{\partial u_{i+1}}{\partial v_i}\right),$$

$$\mathcal{P}\left(\frac{\partial}{\partial z_{i+1}}\nabla_t\dot{x}(t_{i+1}^-)\right) = \mathcal{P}\left(\frac{\partial u_{i+1}}{\partial z_{i+1}} + d\Gamma_{z_{i+1}}(e)(v_{i+1}, v_{i+1})\right),$$
$$\mathcal{P}\left(\frac{\partial}{\partial v_{i+1}}\nabla_t\dot{x}(t_{i+1}^-)\right) = \mathcal{P}\left(\frac{\partial u_{i+1}}{\partial v_{i+1}} + 2\Gamma_{z_{i+1}}(v_{i+1}, e)\right),$$

$$\mathcal{P}\left(\frac{\partial}{\partial z_i}\nabla_t^2\dot{x}(t_{i+1}^-)\right) = \mathcal{P}\left(\frac{\partial w_{i+1}}{\partial z_i} + 3\Gamma_{z_{i+1}}\left(v_{i+1}, \frac{\partial u_{i+1}}{\partial z_i}\right)\right),$$
$$\mathcal{P}\left(\frac{\partial}{\partial v_i}\nabla_t^2\dot{x}(t_{i+1}^-)\right) = \mathcal{P}\left(\frac{\partial w_{i+1}}{\partial v_i} + 3\Gamma_{z_{i+1}}\left(v_{i+1}, \frac{\partial u_{i+1}}{\partial v_i}\right)\right),$$

$$\begin{aligned}\mathcal{P}\left(\frac{\partial}{\partial z_{i+1}}\nabla_t^2\dot{x}(t_{i+1}^-)\right) &= \mathcal{P}\left(\frac{\partial w_{i+1}}{\partial z_{i+1}} + 3d\Gamma_{z_{i+1}}(e)(v_{i+1}, u_{i+1}) + 3\Gamma_{z_{i+1}}\left(v_{i+1}, \frac{\partial u_{i+1}}{\partial z_{i+1}}\right)\right.\\ &+d^2\Gamma_{z_{i+1}}(v_{i+1}, e)(v_{i+1}, v_{i+1}) + d\Gamma_{z_{i+1}}(e)(v_{i+1}, \Gamma_{z_{i+1}}(v_{i+1}, v_{i+1}))\\ &\left.+\Gamma_{z_{i+1}}(v_{i+1}, d\Gamma_{z_{i+1}}(e)(v_{i+1}, v_{i+1}))\right),\end{aligned}$$

$$\begin{aligned}\mathcal{P}\left(\frac{\partial}{\partial v_{i+1}}\nabla_t^2\dot{x}(t_{i+1}^-)\right) &= \mathcal{P}\left(\frac{\partial w_{i+1}}{\partial v_{i+1}} + 3\Gamma_{z_{i+1}}(e, u_{i+1}) + 3\Gamma_{z_{i+1}}\left(v_{i+1}, \frac{\partial u_{i+1}}{\partial v_{i+1}}\right)\right.\\ &+d\Gamma_{z_{i+1}}(e)(v_{i+1}, v_{i+1}) + 2d\Gamma_{z_{i+1}}(v_{i+1})(v_{i+1}, e) + \Gamma_{z_{i+1}}(e, \Gamma_{z_{i+1}}(v_{i+1}, v_{i+1}))\\ &\left.+2\Gamma_{z_{i+1}}(v_{i+1}, \Gamma_{z_{i+1}}(v_{i+1}, e))\right).\end{aligned}$$

AIMS Press