Computing the Riemannian Logarithm on the Stiefel manifold

Metrics, methods and performance

Zimmermann, Ralf; Hüper, Knut

Published in:
SIAM Journal on Scientific Computing

DOI:
10.1137/21M1425426

Publication date:
2022

Document version:
Final published version

Document license:
Unspecified

Citation for pulished version (APA):
Zimmermann, R., & Hüper, K. (2022). Computing the Riemannian Logarithm on the Stiefel manifold: Metrics, methods and performance. *SIAM Journal on Scientific Computing*, *43*(2), 953-980. https://doi.org/10.1137/21M1425426

SIAM J. MATRIX ANAL. APPL.
Vol. 43, No. 2, pp. 953–980

COMPUTING THE RIEMANNIAN LOGARITHM ON THE STIEFEL MANIFOLD: METRICS, METHODS, AND PERFORMANCE*

RALF ZIMMERMANN† AND KNUT HÜPER‡

Abstract. We address the problem of computing Riemannian normal coordinates on the real, compact Stiefel manifold of orthonormal frames. The Riemannian normal coordinates are based on the so-called Riemannian exponential and the associated Riemannian logarithm map and enable one to transfer almost any computational procedure to the realm of the Stiefel manifold. To compute the Riemannian logarithm is to solve the (local) geodesic endpoint problem. Instead of restricting the consideration to geodesics with respect to a single selected metric, we consider a family of Riemannian metrics introduced by Hüper, Markina, and Silva Leite that includes the Euclidean and the canonical metric as prominent examples. As main contributions, we provide (1) a unified, structured, reduced formula for the Stiefel geodesics. The formula is unified in the sense that it works for the full family of metrics under consideration. It is structured in the sense that it relies on matrix exponentials of skew-symmetric matrices exclusively. It is reduced in relation to the dimension of the matrices of which matrix exponentials have to be calculated. We provide (2) a unified method to tackle the geodesic endpoint problem numerically, and (3) we improve the existing Riemannian log algorithm under the canonical metric in terms of the computational efficiency. The findings are illustrated by means of numerical examples, where the novel algorithms prove to be the most efficient methods known to this date.

Key words. Stiefel manifold, Riemannian logarithm, geodesic endpoint problem, Riemannian computing

AMS subject classifications. 15A16, 15B10, 33B30, 33F05, 53-04, 65F60

DOI. 10.1137/21M1425426

1. Introduction.

Riemannian computing methods have established themselves as important tools in a large variety of applications, including computer vision, machine learning, and optimization; see [1, 2, 3, 11, 12, 21, 27, 28] and the anthologies [22, 32]. They also are gaining increasing attention in statistics and data science [26] and in numerical methods for differential equations [4, 8, 14, 19, 42].

One way to enable the practical execution of data processing methods on a curved manifold $\mathcal{M}$ is via working in local coordinates. This holds among others for basic tasks like averaging, clustering, interpolation, and optimization. Of special importance are the *Riemannian normal coordinates*, as they are *radially isometric* [20, section 5]. The Riemannian normal coordinates rely on the *Riemannian exponential* and the *Riemannian logarithm*, which are local diffeomorphisms: The exponential at a manifold location $p \in \mathcal{M}$ sends a tangent vector v (i.e., the velocity vector of a manifold curve) to the endpoint $q = c(1)$ of a geodesic curve c that starts from $p = c(0)$ with velocity $v = \dot{c}(0)$. The Riemannian logarithm at p maps a manifold location $q \in \mathcal{M}$ to the starting velocity vector v of a geodesic c that connects $p = c(0)$ and $q = c(1)$. Figure 1 illustrates this process. Hence, the Riemannian logarithm is associated with the *geodesic endpoint problem*:

*Received by the editors June 7, 2021; accepted for publication (in revised form) by B. Vandereycken February 16, 2022; published electronically June 24, 2022.

https://doi.org/10.1137/21M1425426

Funding: The work of the second author was supported by German Federal Ministry of Education and Research BMBF Project 05M20WWA: Verbundprojekt 05M2020 - DyCA.

†Department of Mathematics and Computer Science, University of Southern Denmark (SDU), Odense, Denmark (zimmermann@imada.sdu.dk).

‡Institute of Mathematics, Julius-Maximilians-Universität, Würzburg, Germany (hueper@mathematik.uni-wuerzburg.de).

"Given $p, q \in \mathcal{M}$, find a geodesic arc that connects p and q."

In this work, we tackle the local geodesic endpoint problem on the Stiefel manifold of orthonormal frames. Geodesics depend on the way the length of velocity vectors of curves are measured and thus on the Riemannian metric. Popular choices for metrics on the Stiefel manifold are the *Euclidean metric* and the *canonical metric*. These will be detailed in section 2. Rather than restricting the considerations to either of these two, we work with the one-parameter family of metrics that is introduced in [17]. This family contains the Euclidean and the canonical Stiefel metric as special cases.

Original contributions.

- We start from the results of [11] and [18] and derive a unified formula for the Stiefel geodesics and thus for the Stiefel exponential. Here, unified is to be understood in the sense that the formula works for all metrics in the one-parameter family under consideration. Moreover it features the same skew-symmetric structure as exhibited by the canonical geodesics and comes at roughly the same computational costs.
- We provide new theoretical insights on the structure of the matrices that come into consideration as candidate solutions for the Stiefel geodesic endpoint problem. For rectangular matrices of dimensions $(n \times p)$, $n \gg p$, this reduces the endpoint problem to finding suitable $(p \times p)$-orthogonal matrices.
- We provide efficient algorithms for computing the Riemannian logarithm in a unified way for the full one-parameter family of metrics of [17].
- For the special case of the canonical metric, the endpoint problem features a simplified structure that was exploited in [39]. We refine this approach and thus accelerate the existing method.
- We juxtapose the various methods by means of numerical experiments, where the new methods prove to outperform both their predecessors from [7, 39] as well as Newton-based approaches.

Related work. The reference [28, section 5] tackles the local geodesic endpoint problem for the canonical metric via a Riemannian optimization approach; [39] also works in the setting of the canonical metric and provides a matrix-algebraic algorithm based on the Baker–Campbell–Hausdorff (BCH) formula with guaranteed local linear convergence. An algorithm for computing the Stiefel logarithm for the Euclidean metric is considered in [7] and is based on the general "shooting method"; see [31, section 6.5]. The thesis [34] considers single shooting and multiple shooting methods based on Newton's method to solve the geodesic endpoint problem under the canonical metric. It also features an algorithm for computing global Stiefel geodesics that is based on the "leapfrog method" of [25] for general manifolds; see also the associated preprint [35]. This approach requires methods that tackle the geodesic endpoint problem locally, i.e., for input points that are close enough to each other, as building blocks. The notion of being "close enough" depends on geometric quantities (like the injectivity radius) but also on the numerical algorithm that is applied to the problem.

Notational specifics. For $p \in \mathbb{N}$, the $(p \times p)$-identity matrix is denoted by $I_p \in \mathbb{R}^{p \times p}$, or simply I, if the dimension is clear. The $(p \times p)$-orthogonal group, i.e., the set of all square orthogonal matrices, is denoted by

$$O(p) = \{\phi \in \mathbb{R}^{p \times p} | \phi^T \phi = \phi \phi^T = I_p\}.$$

The standard matrix exponential and matrix logarithm are denoted by

$$\exp_m(X) := \sum_{j=0}^{\infty} \frac{X^j}{j!}, \quad \log_m(I + X) := \sum_{j=1}^{\infty} (-1)^{j+1} \frac{X^j}{j}.$$

The sets of symmetric and skew-symmetric $(p \times p)$-matrices are $\mathrm{sym}(p) = \{A \in \mathbb{R}^{p\times p} | A^T = A\}$ and $\mathrm{skew}(p) = \{A \in \mathbb{R}^{p\times p} | A^T = -A\}$, respectively. Overloading this notation, $\mathrm{sym}(A) = \frac{1}{2}(A + A^T), \mathrm{skew}(A) = \frac{1}{2}(A - A^T)$ denote the symmetric and skew-symmetric parts of a matrix A.

Unless stated otherwise, when we employ the QR-decomposition of a rectangular matrix $A \in \mathbb{R}^{n\times p}$, we implicitly assume that $n \geq p$ and work with the "compact" QR-decomposition $A = QR$, with $Q \in \mathbb{R}^{n\times p}$, tall rectangular and $R \in \mathbb{R}^{p\times p}$ triangular.

2. The Stiefel manifold. This section reviews the essential aspects of Stiefel manifolds in regard to numerical, matrix-algorithmic applications. For additional background, see [2, 11, 42]. For featured applications, see, e.g., [5, 9, 16, 37].

The *Stiefel manifold* $St(n, p)$ is the set of rectangular, column-orthonormal n-by-p matrices,

$$St(n, p) := \{U \in \mathbb{R}^{n\times p} |\ U^T U = I_p\}, \quad p \leq n.$$

Observe that this matrix set is the preimage $St(n, p) = F^{-1}(0)$ of the function $F : \mathbb{R}^{n\times p} \to \mathrm{sym}(p), Y \mapsto Y^T Y - I_p$. By the regular value theorem, it is a differentiable manifold of dimension $np - \frac{1}{2}p(p + 1)$.

The *tangent space* $T_U St(n, p)$ at $U \in St(n, p)$ is represented as

$$T_U St(n, p) = \left\{\Delta \in \mathbb{R}^{n\times p} |\quad U^T \Delta \in \mathrm{skew}(p)\right\}.$$

For brevity, we will often write T_U instead of $T_U St(n, p)$. Every tangent vector $\Delta \in T_U$ may be written as

$$\Delta = UA + (I - UU^T)T, \qquad A \in \mathrm{skew}(p), \quad T \in \mathbb{R}^{n\times p} \text{ arbitrary}, \tag{1}$$

$$\Delta = UA + U^\perp H, \qquad A \in \mathrm{skew}(p), \quad H \in \mathbb{R}^{(n-p)\times p} \text{ arbitrary}. \tag{2}$$

In the latter case, $U^\perp \in St(n, n - p)$ is an orthonormal completion such that the matrix $\left(U \mid U^\perp\right)$ is orthogonal. Any matrix $W \in \mathbb{R}^{n\times p}$ can be projected onto T_U by

$$\Pi_U(W) = W - U\,\mathrm{sym}(U^T W); \tag{3}$$

see [11, equations (2.3), (2.4)].

In order to turn $St(n, p)$ into a *Riemannian manifold* [20], a metric, i.e., an inner product $\langle\cdot, \cdot\rangle_U$ with associated norm $\|\cdot\|_U = \sqrt{\langle\cdot, \cdot\rangle_U}$ on the tangent spaces T_U, must be defined for all $U \in St(n, p)$. Given a metric, the length of a curve $C : [a, b] \to St(n, p)$ is $L(C) := \int_a^b \|\dot{C}(t)\|_{C(t)} dt$. Candidates for length-minimizing curves are called *geodesics* and are locally uniquely determined by an ordinary initial value problem when specifying a starting point $C(0)$ and a starting velocity $\dot{C}(t)$ [20, section 6]. It is obvious that geodesics depend on the underlying Riemannian metric.

Geodesics give rise to the Riemannian exponential map. On the Stiefel manifold, the Riemannian exponential at a base point $U \in St(n, p)$ sends a Stiefel tangent vector Δ to the endpoint $C(1) = \widetilde{U} \in St(n, p)$ of a geodesic $t \mapsto C_{U,\Delta}(t)$ that starts from $C(0) = U$ with velocity vector $\dot{C}(0) = \Delta$,

$$\mathrm{Exp}_U(\Delta) := C_{U,\Delta}(1).$$

As a consequence, $\mathrm{Exp}_U(t\Delta) = C_{U,\Delta}(t)$ for $t \in [0, 1]$. Knowing the Riemannian exponential is knowing the geodesics and vice versa.

The Riemannian exponential is locally invertible. The inverse is called the *Riemannian logarithm* and is denoted

$$\mathrm{Log}_U : St(n, p) \ni \widetilde{U} \mapsto \mathrm{Log}_U(\widetilde{U}) := \mathrm{Exp}_U^{-1}(\widetilde{U}) \in T_U St(n, p). \tag{4}$$

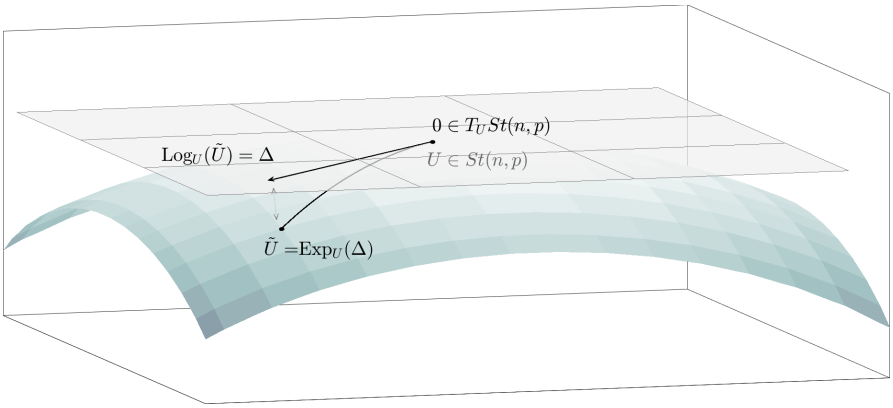

FIG. 1. *(cf. section 2) Conceptual visualization of the Stiefel manifold $St(n,p)$ (curved surface) and associated tangent space $T_U St(n,p)$ (shaded plane). The Riemannian exponential sends a tangent vector Δ from $T_U St(n,p)$ to the endpoint $\widetilde{U}$ of the geodesic $C_{U,\Delta} : [0,1] \to St(n,p)$ with initial values $C_{U,\Delta}(0) = U, \dot{C}_{U,\Delta}(0) = \Delta$. The Riemannian logarithm inverts this process.*

More precisely, $\mathrm{Log}_U(\widetilde{U})$ is well defined for all $\widetilde{U}$ within the *injectivity radius* $r_{St}(U)$ of $St(n,p)$ at U. The injectivity radius at U is the Riemannian distance of U to its cut locus C_U. The cut locus, in turn, is the set of points beyond which the geodesics starting from U cease to be length-minimizing [10, p. 271]. Combined, the Riemannian logarithm and exponential provide the *Riemannian normal coordinates*, which allow us to map data points back and forth between the curved manifold and the flat tangent space; see Figure 1. This is crucial for all data processing operations on manifolds (optimization, interpolation, averaging, clustering...). The Riemannian normal coordinates are special in that they are length-preserving along geodesic rays; one speaks of radial isometries.

The Euclidean and the canonical metric and generalizations. Let $U \in St(n,p)$ and let $\Delta = UA + U^{\perp}H$, $\widetilde{\Delta} = U\widetilde{A} + U^{\perp}\widetilde{H} \in T_U$. Here and in the following, $A = U^T\Delta, \widetilde{A} = U^T\widetilde{\Delta} \in \mathrm{skew}(p)$. There are two standard metrics on the Stiefel manifold.

The *Euclidean metric* on T_U is the one inherited from the ambient $\mathbb{R}^{n\times p}$:

$$\langle \Delta, \widetilde{\Delta} \rangle_U^e = \mathrm{tr}(\Delta^T\widetilde{\Delta}) = \mathrm{tr}\, A^T\widetilde{A} + \mathrm{tr}\, H^T\widetilde{H}.$$

The *canonical metric* on T_U is derived from the quotient representation $St(n,p) = O(n)/(O(n-p))$ of the Stiefel manifold (see [11]) and reads

$$\langle \Delta, \widetilde{\Delta} \rangle_U^c = \mathrm{tr}\left(\Delta^T\left(I - \frac{1}{2}UU^T\right)\widetilde{\Delta}\right) = \frac{1}{2}\,\mathrm{tr}\, A^T\widetilde{A} + \mathrm{tr}\, H^T\widetilde{H}.$$

Let $A = (a_{ij})_{i,j\le p}$ and $H = (h_{ij})_{i\le n; j\le p}$. The Euclidean metric corresponds to measuring tangent vectors $\Delta = UA + U^{\perp}H$ in the Frobenius matrix norm

$$\sqrt{\langle \Delta, \Delta \rangle_U^e} = \|\Delta\|_F = \sqrt{\|A\|_F^2 + \|H\|_F^2} = \sqrt{2\sum_{i<j} a_{ij}^2 + \sum_{i,j} h_{ij}^2},$$

while the canonical metric yields

$$\sqrt{\langle \Delta, \Delta \rangle_U^c} = \sqrt{\frac{1}{2}\|A\|_F^2 + \|H\|_F^2} = \sqrt{\sum_{i<j} a_{ij}^2 + \sum_{i,j} h_{ij}^2}.$$

In this sense, the Euclidean metric disregards the skew-symmetry of A and the independent entries $a_{ij}, i < j$ are counted twice, as was observed in [11, section 2.4].

The work [17] recognizes the Euclidean and the canonical metric as special cases of a one-parameter family of inner products

$$\langle \Delta, \widetilde{\Delta} \rangle_U^\alpha = \operatorname{tr}\left(\Delta^T \left(I - \frac{2\alpha+1}{2(\alpha+1)} UU^T\right)\widetilde{\Delta}\right) = \frac{1}{2(\alpha+1)} \operatorname{tr} A^T \widetilde{A} + \operatorname{tr} H^T \widetilde{H} \tag{5}$$

for $\alpha \in \mathbb{R} \setminus \{-1\}$.[1] For $\alpha = -\frac{1}{2}$ and $\alpha = 0$, the Euclidean and the canonical metric are recovered, respectively. As can be seen from (5), the metric parameter α controls how much weight is put on the $\frac{1}{2}(p-1)p$ degrees of freedom in the matrix $A \in \operatorname{skew}(p)$ relative to the $(n-p)p$ degrees of freedom in $H \in \mathbb{R}^{(n-p)\times p}$. Perfect balance is at $\alpha = 0$, which corresponds to the canonical metric.

Calculating the Riemannian Stiefel exponential: The state of the art. A closed-form expression for the Stiefel exponential w.r.t. the Euclidean metric is derived in [11, section 2.2.2],

$$\widetilde{U} = \operatorname{Exp}_U^e(\Delta) = \begin{pmatrix} U & \Delta \end{pmatrix} \operatorname{exp}_m \begin{pmatrix} A & -\Delta^T\Delta \\ I_p & A \end{pmatrix} \begin{pmatrix} I_p \\ 0 \end{pmatrix} \operatorname{exp}_m(-A). \tag{6}$$

In [18], an alternative formula is obtained:

$$\widetilde{U} = \operatorname{Exp}_U^e(\Delta) = \operatorname{exp}_m\left(\Delta U^T - U\Delta^T\right) U \operatorname{exp}_m(-A). \tag{7}$$

The advantage is that in this form, the Stiefel exponential features only matrix exponentials of skew-symmetric matrices. The downside is that $\Delta U^T - U\Delta^T \in \operatorname{skew}(n)$ and working with $(n \times n)$-matrices might be prohibitively expensive in large-scale applications, where $n \gg p$.

In [17], the formula (7) is generalized to the exponential for all α-metrics of (5),

$$\widetilde{U} = \operatorname{Exp}_U^\alpha(\Delta) = \operatorname{exp}_m\left(-\frac{2\alpha+1}{\alpha+1} UAU^T + \Delta U^T - U\Delta^T\right) U \operatorname{exp}_m\left(\frac{\alpha}{\alpha+1} A\right). \tag{8}$$

An algorithm for computing the Stiefel exponential w.r.t. the canonical metric was derived in [11, section 2.4.2]: Given U, Δ, first compute a compact QR-decomposition $QR = (I - UU^T)\Delta$ with $Q \in St(n,p), R \in \mathbb{R}^{p\times p}$. Then form

$$\widetilde{U} = \operatorname{Exp}_U^c(\Delta) = \begin{pmatrix} U & Q \end{pmatrix} \operatorname{exp}_m \begin{pmatrix} A & -R^T \\ R & 0 \end{pmatrix} \begin{pmatrix} I_p \\ 0 \end{pmatrix}, \tag{9}$$

where again $A = U^T\Delta \in \operatorname{skew}(p)$. Using this form becomes efficient if $p < \frac{n}{2}$.

Calculating the Riemannian Stiefel logarithm: State of the art. Computing the Riemannian logarithm corresponds to solving the *geodesic endpoint problem* locally:

[1] For a deeper reason why $\alpha = -1$ has to be excluded, see [17].

Given $U, \widetilde{U} \in St(n,p)$ find a starting velocity $\Delta \in T_U$ such that

$$\mathrm{Exp}_U(\Delta) = \widetilde{U} \quad \left(\Leftrightarrow \Delta = \mathrm{Log}_U(\widetilde{U})\right).$$

A generic method for solving the geodesic endpoint problem on any manifold is the shooting method; see [31, section 6.5]. This method is considered in [7] to compute the Stiefel logarithm. For the specific case of $St(4,2)$ it features in [33]. The generic principle of the shooting method is as follows:

- Find an initial guess $\Delta_0 \in T_U$.
- "Shoot" a geodesic in the direction of Δ_0, i.e., compute $\widetilde{U}_0 := \mathrm{Exp}_U(\Delta_0)$.
- Measure the gap f_g between $\widetilde{U}_0$ and the actual target $\widetilde{U}$ as a function of Δ.
- Use information on the gap to update $\Delta_1 \leftarrow \Delta_0$ and repeat.

There are many options to implement these steps. In [7], the initial guess is chosen as $\Delta_0 = \Pi_U(U - \widetilde{U}) \in T_U$. The gap is measured in the ambient space $\mathbb{R}^{n\times p}$ as $f_g(\Delta) = \|\widetilde{U}_0 - \widetilde{U}\|_F^2 = \|\mathrm{Exp}_U(\Delta) - \widetilde{U}\|_F^2$. A natural choice for updating the shooting direction Δ is a gradient descent based on this gap function. An alternative is to tackle the matrix root finding problem $\mathrm{Exp}_U(\Delta) - \widetilde{U} = 0$ with the Newton method. This approach is pursued in [34, section 2.3]. However, both of these approaches require in particular the derivative of the matrix exponential, which is expensive to obtain, see [15, section 10.6] for the general formulas and [33, Proposition 12], [40, Lemma 5] for precise applications to the Stiefel exponential.

The reference [7] proposes a method that avoids calculating derivatives. The idea is to project the gap vector $\mathrm{Exp}_U(\Delta) - \widetilde{U}$ from $\mathbb{R}^{n\times p}$ onto $T_{\widetilde{U}}St(n,p)$ and to parallel-translate the result along the geodesic $t \to \mathrm{Exp}_U(t\Delta)$ back to $T_USt(n,p)$, where the update of the shooting direction Δ is then performed. This process is detailed in Algorithm 1. For a higher computational efficiency, only an approximation of the

Algorithm 1. Shooting method, adapted from [7, Algorithm 1]

Input: Stiefel matrices $U, \widetilde{U} \in St(n,p)$, convergence threshold $\epsilon > 0$, time step array $T = \{t_0, t_1, \ldots, t_m\}$, with $t_0 = 0, t_m = 1.0$.

1: $\gamma \leftarrow \|\widetilde{U} - U\|$ {compute gap between base point and target}
2: $\Delta \leftarrow \gamma \frac{\Pi_U(\widetilde{U})}{\|\Pi_U(\widetilde{U})\|}$ {project gap vector $\widetilde{U} - U$ onto T_U, preserve length}
3: **while** $\gamma > \epsilon$ **do**
4: **for** $j = 1, \ldots, m$ **do**
5: $\widetilde{U}^s(j) \leftarrow \mathrm{Exp}_U(t_j\Delta)$ {discrete representation of geodesic}
6: **end for**
7: $\Delta^s \leftarrow \widetilde{U}^s(m) - \widetilde{U}$ {current gap vector: compare endpoint of geodesic to $\widetilde{U}$}
8: $\gamma \leftarrow \|\Delta^s\|$
9: **for** $j = m, \ldots, 0$ **do**
10: $\Delta^s \leftarrow \gamma \frac{\Pi_{\widetilde{U}^s(j)}(\Delta^s)}{\|\Pi_{\widetilde{U}^s(j)}(\Delta^s)\|}$ {initial projection plus approximate parallel transport}
11: **end for**
12: update $\Delta \leftarrow \Delta - \Delta^s$
13: **end while**

Output: Δ

Note: The first inner iteration in the loop in steps 9–11 projects the gap vector of step 7 onto $T_{\widetilde{U}^s(m)}$. The remaining inner iterations in this loop realize an approximation of the parallel transport along the discretized geodesic $t \mapsto \mathrm{Exp}_U(t\Delta)$.

parallel transport is realized on equidistant time steps $0 = t_0, t_1, \ldots, t_m = 1$ in the unit interval $[0, 1]$. The more time steps there are, the more accurate will be the result of the parallel transport. However, this does not help if the tangent information that is to be transported is of poor quality in the first place.

The "multiple shooting" as featured in [34, section 2.4] shares the idea of dissecting the geodesic lines under consideration into m segments. Yet in this method, one actually solves subproblems on the segments with Newton's method.

A Stiefel log algorithm that is tailored for the canonical metric and features guaranteed local linear convergence is given in [39]. This approach will be explained and enhanced in subsection 3.4.

3. Fast computational schemes for solving the geodesic endpoint problem. In this section, we first prepare the grounds for a unified framework to tackle the local geodesic endpoint problem on the Stiefel manifold under the one-parameter family of metrics. Then, we introduce practical numerical algorithms for computing the associated Stiefel logarithm, which improve on the state of the art in terms of the computational efficiency.

3.1. A reduced formula for the Stiefel exponential. As a starting point for efficient computational schemes, we derive an alternative expression for the α-metric Stiefel exponential that combines the advantages of (6) and (8) and represents a considerable computational reduction if $n \gg p$. To this end, assume that $p \leq \frac{n}{2}$. Let $U \in St(n,p)$ be given and take any suitable column-orthonormal extension $U^\perp \in St(n-p,p)$, i.e., any $U^\perp$ such that $\Phi = \left(U \mid U^\perp\right) \in O(n)$ is square and orthogonal. Observe that $\Phi^T U = \left(\begin{smallmatrix} I_p \\ 0 \end{smallmatrix}\right)$. With $\nu = \frac{2\alpha+1}{\alpha+1}$, $\mu = \frac{\alpha}{\alpha+1} = \nu - 1$, we rewrite (8):

$$\begin{aligned}
\operatorname{Exp}_U^\alpha(\Delta) &= \Phi\Phi^T \exp_m\left(-\nu U A U^T + \Delta U^T - U\Delta^T\right) \Phi\Phi^T U \exp_m(\mu A) \\
&= \Phi \exp_m\left(-\nu \Phi^T U A U^T \Phi + \Phi^T\left(\Delta U^T - U\Delta^T\right)\Phi\right) \Phi^T U \exp_m(\mu A) \\
&= \Phi \exp_m \begin{pmatrix} -\nu A + U^T\Delta - \Delta^T U & -\Delta^T U^\perp \\ (U^\perp)^T\Delta & 0 \end{pmatrix} \begin{pmatrix} I_p \\ 0 \end{pmatrix} \exp_m(\mu A) \\
&= \left(U \mid U^\perp\right) \exp_m \begin{pmatrix} (2-\nu)A & -H^T \\ H & 0 \end{pmatrix} \begin{pmatrix} I \\ 0 \end{pmatrix} \exp_m(\mu A), \quad H = (U^\perp)^T\Delta. \qquad (10)
\end{aligned}$$

Let $H = \widetilde{Q}\left(\begin{smallmatrix} B \\ 0 \end{smallmatrix}\right)$ with $\widetilde{Q} = (\widetilde{Q}_p \mid \widetilde{Q}_{n-2p}) \in O(n-p)$ be a (full) QR-decomposition of H. (Only the orthonormality of $\widetilde{Q}$ matters; B need not be triangular for the following considerations.) We can factor

$$\begin{pmatrix} (2-\nu)A & -H^T \\ H & 0 \end{pmatrix} = \begin{pmatrix} I_p & 0 \\ 0 & \widetilde{Q} \end{pmatrix} \exp_m \begin{pmatrix} (2-\nu)A & -B^T & 0 \\ B & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} I_p & 0 \\ 0 & \widetilde{Q}^T \end{pmatrix},$$

which yields $\operatorname{Exp}_U^\alpha(\Delta) = (U \mid U^\perp\widetilde{Q}_p) \exp_m\left(\begin{smallmatrix} (2-\nu)A & -B^T \\ B & 0 \end{smallmatrix}\right)\left(\begin{smallmatrix} \exp_m(\mu A) \\ 0 \end{smallmatrix}\right)$. Note that $U^\perp H = U^\perp(U^\perp)^T\Delta = (I - UU^T)\Delta$. Hence, instead of computing a QR-decomposition of $U^\perp H$, we can directly compute a compact QR-decomposition $(I - UU^T)\Delta = QB$ with $Q \in St(n,p)$. (Again, no special structure is required for $B \in \mathbb{R}^{p\times p}$.) This leads to the following proposition.

PROPOSITION 1. *Let $\alpha \neq -1$. For $U \in St(n,p)$, $\Delta \in T_U St(n,p)$ the Stiefel exponential under the α-metric of* (5) *reads*

$$\text{(11)}\qquad \mathrm{Exp}_U^{\alpha}(\Delta) = \begin{pmatrix} U & Q \end{pmatrix} \exp_m \begin{pmatrix} \frac{1}{\alpha+1} A & -B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} I_p \\ 0 \end{pmatrix} \exp_m \left(\frac{\alpha}{\alpha+1} A \right),$$

where $A = U^T\Delta \in \mathrm{skew}(p)$ *and* $QB = (I - UU^T)\Delta \in \mathbb{R}^{n\times p}$ *is any matrix decomposition with with* $Q \in St(n,p)$ *and* $B \in \mathbb{R}^{p\times p}$.

As with (9), this form features only standard matrix exponentials of skew-symmetric matrices of size p rather than n. With Proposition 1, the Stiefel exponential is computable for all metrics in the α-family (5) in $\mathcal{O}(np^2)$ flops. A more detailed look at the calculations reveals that the formula (11) remains valid also if $p > \frac{n}{2}$. However, in this case, it represents in fact an increase in dimension for the main matrix exponential rather than a reduction when compared to (8). The formula (11) continues to hold if the orthogonal component $(I-UU^T)\Delta$ of the tangent vector is rank-deficient. In this case, it is understood that the QR-decomposition $QB = (I-UU^T)\Delta$ is arranged such that $QB = (Q_r \mid Q_{p-r})\left(\begin{smallmatrix} B_r \\ 0 \end{smallmatrix}\right)$, where $B_r \in \mathbb{R}^{r\times p}$ and $\mathrm{rank}(B_r) = \mathrm{rank}((I-UU^T)\Delta) = r$ and $B_r = 0$ in the extreme case of $0 = (I - UU^T)\Delta$.

Before we continue with the main body of this work, we note an interesting aside.

LEMMA 2. *For* $A \in \mathrm{skew}(p)$, $H \in \mathbb{R}^{(n-p)\times p}$, *it holds that*

$$\begin{pmatrix} I_p & A \\ 0 & H \end{pmatrix} \exp_m \begin{pmatrix} A & A^2 - H^TH \\ I_p & A \end{pmatrix} \begin{pmatrix} I_p \\ 0 \end{pmatrix} = \exp_m \begin{pmatrix} 2A & -H^T \\ H & 0 \end{pmatrix} \begin{pmatrix} I_p \\ 0 \end{pmatrix}.$$

Proof. Fix an orthogonal matrix $\begin{pmatrix} U & U^{\perp} \end{pmatrix} \in O(n)$ and construct $\Delta = UA + U^{\perp}H$. Then, the lemma is a consequence of the fact that both (6) and (10) with $\alpha = -\frac{1}{2}$ are valid expressions of the (unique) Stiefel matrix exponential $\mathrm{Exp}_U^e(\Delta)$. The underlying geodesics $t \mapsto \mathrm{Exp}_U^e(t\Delta)$ satisfy the same initial value problem and thus coincide; see [20, Theorem 4.27, p. 103]. ∎

3.2. The Stiefel log matrix equations. We continue to work under the assumption that $p \leq \frac{n}{2}$ with the setting of $n \gg p$ in mind for practical big data applications. In this section, we show that, in essence, computing the Stiefel logarithm for all α-metrics corresponds to solving a nonlinear matrix equation, where the unknown is a $2p \times 2p$ skew-symmetric matrix.

Let $U \in St(n,p)$ and let $\widetilde{U} \in St(n,p)$ be within the injectivity radius $r_{St}(U)$ of $St(n,p)$ at U. Let $\Delta \in T_U$ be such that $\mathrm{Exp}_U^{\alpha}(\Delta) = \widetilde{U}$. Then, $\widetilde{U}$ has a representation

$$\widetilde{U} = UM + QN, \quad M, N \in \mathbb{R}^{n\times p}, \quad M^TM + N^TN = I_p,$$

where $Q \in St(n,p)$ is the "Q-factor" of a compact QR-decomposition $QB = (I - UU^T)\Delta$. For the canonical metric ($\alpha = 0$), this is immediately clear from (9). For the full family of α-metrics, this follows from (11). More precisely, M, N form the first block column of an orthogonal matrix

$$\text{(12)}\qquad \begin{pmatrix} M & X \\ N & Y \end{pmatrix} = \exp_m \begin{pmatrix} \frac{1}{\alpha+1} A & -B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \exp_m(\frac{\alpha}{\alpha+1} A) & 0 \\ 0 & I_p \end{pmatrix} \in SO(2p).$$

Recall $A = U^T\Delta \in \mathrm{skew}(p)$ and that $\alpha = 0$ and $\alpha = -\frac{1}{2}$ reproduce the canonical and the Euclidean metric, respectively. For brevity, write again

$$\mu = \frac{\alpha}{\alpha+1}, \quad \alpha \neq -1.$$

Now, we aim at finding the inverse $(\mathrm{Exp}_U^\alpha)^{-1}(\widetilde{U}) = \Delta$ with Δ as unknown. We can still obtain M and N from the given data points $U, \widetilde{U}$ via

$$M = U^T\widetilde{U}, \quad QN = (I - UU^T)\widetilde{U}.$$

For the latter equation, any matrix decomposition that represents $(I - UU^T)\widetilde{U}$ via a "subspace factor" Q with orthonormal columns and a corresponding "coordinates factor" N is suitable in order to obey the constraint that M and N be blocks of an orthogonal matrix, i.e., $I = M^TM + N^TN$. This incorporates some ambiguity. If $(I - UU^T)\widetilde{U}$ has full rank p, then Q and N are unique up to a rotation/reflection $\Phi \in O(p)$. More precisely, for $\widetilde{Q}\widetilde{N} = (I - UU^T)\widetilde{U} = QN$, it holds that $Q = \widetilde{Q}\Phi, N = \Phi^T\widetilde{N}$ with $\Phi = \widetilde{Q}^TQ$. If $\mathrm{rank}(I - UU^T)\widetilde{U} = r < p$, then we may assume that

$$(I - UU^T)\widetilde{U} = QN = \begin{pmatrix} Q_r & | & Q_{p-r} \end{pmatrix} \begin{pmatrix} N_r \\ 0 \end{pmatrix}, \tag{13}$$

where $Q_r \in St(n, r), Q_{p-r} \in St(n, p-r), N_r \in \mathbb{R}^{p\times r}$. This determines Q_r, N_r uniquely up to a rotation/reflection $\Phi_r \in O(r)$. Yet, the column block Q_{p-r} may be an arbitrary orthonormal extension.

Once Q, M, N are chosen and fixed, we can compute matrix blocks X_0, Y_0 such that they form an orthogonal completion $V = \left(\begin{smallmatrix} M & X_0 \\ N & Y_0 \end{smallmatrix}\right) \in O(2p)$. The restriction of the exponential map to the skew-symmetric matrices

$$\exp_m|_{\mathrm{skew}(p)} : \mathrm{skew}(p) \to SO(p)$$

is surjective [13, section 3.11, Theorem 9]. Hence, (12) requires selecting X_0, Y_0 such that $\det(V) = +1$.[2] Again, such a completion is not unique and it cannot be expected that the such found X_0, Y_0 align with the structure of (12). Yet, there is an orientation preserving orthogonal matrix $\Phi = \exp_m(C), C \in \mathrm{skew}(p)$ such that $\left(\begin{smallmatrix} X_0 \\ Y_0 \end{smallmatrix}\right)\Phi = \left(\begin{smallmatrix} X \\ Y \end{smallmatrix}\right)$ is the sought-after completion. In summary, computing the Stiefel logarithm boils down to solving the following nonlinear matrix equation:

$$\text{solve} \quad F\begin{pmatrix} A & -B^T \\ B & C \end{pmatrix} = \begin{pmatrix} M & X_0 \\ N & Y_0 \end{pmatrix}, \quad \text{for } F : \mathrm{skew}(2p) \to SO(2p), \tag{14}$$

$$F\begin{pmatrix} A & -B^T \\ B & C \end{pmatrix} = \exp_m\begin{pmatrix} (1-\mu)A & -B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} \exp_m(\mu A) & 0 \\ 0 & \exp_m(-C) \end{pmatrix}, \mu = \frac{\alpha}{\alpha+1}.$$

The sought-after tangent vector is then $\Delta = UA + QB \in T_U$. It is worth mentioning that any ambiguity in Q has no impact on the final Δ, not even in the case where the component $(I - UU^T)\widetilde{U}$ of $\widetilde{U}$ does not feature full rank. This is confirmed in the next theorem, which can be considered as a generalization of [39, Theorem 3.1].

THEOREM 3. *Let $U, \widetilde{U} \in St(n, p)$ with $\mathrm{dist}(U, \widetilde{U}) < r_{St}(U)$ and let*

$$U^T\widetilde{U} = M, \quad (I - UU^T)\widetilde{U} = QN = \begin{pmatrix} Q_r & Q_{p-r} \end{pmatrix}\begin{pmatrix} N_r \\ 0 \end{pmatrix}, \quad r \le p,$$

with $Q \in St(n, p)$ and $r = \mathrm{rank}(N_r) = \mathrm{rank}(N)$. Then $\Delta = \mathrm{Log}_U^\alpha(\widetilde{U})$ features a representation

$$\Delta = UA + QB = UA + \begin{pmatrix} Q_r & Q_{p-r} \end{pmatrix}\begin{pmatrix} B_r \\ 0 \end{pmatrix}, \quad A \in \mathrm{skew}(p), \quad B_r \in \mathbb{R}^{r\times p}.$$

Here, A, B are components of a skew-symmetric block matrix that solves (14).

[2] As an aside, this is why the example S1 from the supplements of [39] fails.

Proof. Let $M, N \in \mathbb{R}^{p\times p}$, with $N = \left(\begin{smallmatrix} N_r \\ 0 \end{smallmatrix}\right), N_r \in \mathbb{R}^{r\times p}$ of full rank r constructed as stated above. We can restrict the considerations to block extensions X_0, Y_0 to an orthogonal matrix of the form

$$V = \begin{pmatrix} M & X_0 \\ N & Y_0 \end{pmatrix} = \left(\begin{array}{c|c|c} M & X_r^0 & 0 \\ \hline N_r & Y_r^0 & 0 \\ \hline 0 & 0 & I_{p-r} \end{array}\right) \in SO(2p),$$

where $X_r^0 \in \mathbb{R}^{p\times r}, Y_r^0 \in \mathbb{R}^{r\times r}$ are obtained, say, via the Gram–Schmidt method.

Let $\left(\begin{smallmatrix} A & -B^T \\ B & C \end{smallmatrix}\right)$ be a solution to (14) that preserves the above structure of V so that in particular

$$\left(\begin{array}{c|c} X_r^0 & 0 \\ \hline Y_r^0 & 0 \\ \hline 0 & I_{p-r} \end{array}\right) \exp_m(C) = \left(\begin{array}{c|c} X_r & 0 \\ \hline Y_r & 0 \\ \hline 0 & I_{p-r} \end{array}\right).$$

This entails $\exp_m(C) = \Phi = \left(\begin{smallmatrix} \Phi_r & 0 \\ 0 & I_{p-r} \end{smallmatrix}\right) \in SO(p)$. Then, for all α-metrics, it holds with $\mu = \frac{\alpha}{\alpha+1}$ that

(15)
$$\log_m\left(\left(\begin{array}{c|c|c} M & X_r^0 & 0 \\ \hline N_r & Y_r^0 & 0 \\ \hline 0 & 0 & I_{p-r} \end{array}\right)\left(\begin{array}{c|c|c} \exp_m(-\mu A) & 0 & 0 \\ \hline 0 & \Phi_r & 0 \\ \hline 0 & 0 & I_{p-r} \end{array}\right)\right) = \begin{pmatrix} (1-\mu)A & -B^T \\ B & 0 \end{pmatrix}.$$

Because the matrix logarithm acts blockwise on block-diagonal matrices, the block structure of the matrix on the left-hand side entails a corresponding block structure for the right-hand side of (15). In particular, necessarily $B = \left(\begin{smallmatrix} B_r \\ 0 \end{smallmatrix}\right)$ with $\operatorname{rank}(B_r) = r$.

Now, define $\Delta = UA + Q_r B_r$ and evaluate the Stiefel exponential to check if $\operatorname{Exp}_U^\alpha(\Delta) = \widetilde{U}$. According to (11), the procedure requires computing $U^T\Delta = A$, as well as an orthogonal decomposition

$$(I - UU^T)\Delta = \left(\widehat{Q}_r \mid \widehat{Q}_{p-r}\right)\begin{pmatrix} \widehat{B}_r \\ 0 \end{pmatrix}.$$

The matrix factors $\widehat{Q}_r$ and $\widehat{B}_r$ are not necessarily equal to Q_r and B_r, respectively. Yet, by construction, it holds that $\widehat{Q}_r\widehat{B}_r = (I - UU^T)\Delta = (I - UU^T)Q_rB_r = Q_rB_r$. Since $Q_r, \widehat{Q}_r$ span the same column-space, $S := Q_r^T\widehat{Q}_r$ is orthogonal and $\widehat{Q}_r = Q_rS, \widehat{B}_r = S^TB_r$. The Stiefel exponential produces

$$\begin{aligned}
\operatorname{Exp}_U^\alpha(\Delta) &= \left(U \mid \widehat{Q}_r \mid \widehat{Q}_{p-r}\right)\exp_m\begin{pmatrix} (1-\mu)A & -\widehat{B}_r^T & 0 \\ \widehat{B}_r & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} \exp_m(\mu A) \\ 0 \\ 0 \end{pmatrix} \\
&= \left(U \mid \widehat{Q}_r\right)\exp_m\begin{pmatrix} (1-\mu)A & -\widehat{B}_r^T \\ \widehat{B}_r & 0 \end{pmatrix}\begin{pmatrix} \exp_m(\mu A) \\ 0 \end{pmatrix} \\
&= \left(U \mid \widehat{Q}_rS\right)\exp_m\begin{pmatrix} (1-\mu)A & -\widehat{B}_r^TS \\ S^T\widehat{B}_r & 0 \end{pmatrix}\begin{pmatrix} \exp_m(\mu A) \\ 0 \end{pmatrix} \\
&= \left(U \mid Q_r\right)\exp_m\begin{pmatrix} (1-\mu)A & -B_r^T \\ B_r & 0 \end{pmatrix}\begin{pmatrix} \exp_m(\mu A) \\ 0 \end{pmatrix} = \left(U \mid Q_r\right)\begin{pmatrix} M \\ N_r \end{pmatrix} = \widetilde{U}
\end{aligned}$$

according to (15). ∎

Remark 4 (the search space for the Stiefel log algorithm). For Stiefel points U and $\widetilde{U} = \operatorname{Exp}_U^\alpha(\Delta)$ within the injectivity radius at U, Theorem 3 shows that both the location $\widetilde{U}$ and the tangent vector Δ are in the same matrix space

$$\mathcal{S} = \{UV | V \in \mathbb{R}^{p\times p}\} \oplus \{QW | W \in \mathbb{R}^{p\times p}\} \subset \mathbb{R}^{n\times p}. \tag{16}$$

This fact can be exploited in numerical schemes for computing the Stiefel logarithm. A direct consequence is that any numerical algorithm can focus on finding the missing factors $A, B \in \mathbb{R}^{p\times p}$.

3.3. A *p*-shooting method tailored to the Stiefel logarithm. In this section, we customize the generic iterative shooting method of Algorithm 1 for solving the geodesic endpoint problem. With large "tall-and-skinny" matrices and big data applications in mind, our contribution is to modify the required calculations such that the loop iterations are tailored for the use of (11) and (16) and work exclusively with $2p \times 2p$ matrices rather than with $n \times p$ matrices. This leads to considerable savings, if $n \gg p$.

Given $U, \widetilde{U} \in St(n,p)$, the starting point is the representation $\widetilde{U} = UU^T\widetilde{U} + (I - UU^T)\widetilde{U} = U\widehat{M} + Q\widehat{N}$, where $\widehat{M} = U^T\widetilde{U} \in \mathbb{R}^{p\times p}$ and $Q\widehat{N} = (I - UU^T)\widetilde{U} \in \mathbb{R}^{n\times p}$ is a compact QR-decomposition. The essential observation is that all iterates Δ, Δ^s of Algorithm 1 actually remain in the matrix space $\mathcal{S}$ of (16) that is spanned by the fixed U and Q. According to Theorem 3 and Remark 4, $\mathcal{S}$ also contains the sought-after solution.

PROPOSITION 5. *Let $U, \widetilde{U} \in St(n,p)$ and let $Q\widehat{N} = (I - UU^T)\widetilde{U} \in \mathbb{R}^{n\times p}$ be a compact QR-decomposition. All tangent matrices Δ produced iteratively by Algorithm* 1 *and all the update matrices Δ^s that are the final outcome of one pass through the while-loop in Algorithm* 1 *allow for a representation of the form*

$$\Delta = UA + QR, \quad \Delta^s = UA^s + QR^s, \quad A, A^s \in \mathrm{skew}(p), R, R^s \in \mathbb{R}^{p\times p}.$$

Remark. It is important to emphasize that it is *the same* Q-factor that works for all the iterates Δ, Δ^s. This enables us to restrict the update to the A- and R-factors. The intermediate matrices Δ^s under the while-loop are also contained in the matrix space $\mathcal{S}$ of Remark 4, but they are of the form $\Delta^s = UX^s + QR^s$ with $X^s \notin \mathrm{skew}(p)$ in general.

Proof. The initial Δ_0 is obtained from the projection

$$\begin{aligned}\Delta_0 = \Pi_U(\widetilde{U} - U) = \Pi_U(\widetilde{U}) &= U\widehat{M} + Q\widehat{N} - U\,\mathrm{sym}(\widehat{M}) \\ &= U\,\mathrm{skew}(\widehat{M}) + Q\widehat{N} =: UA_0 + QR_0.\end{aligned}$$

The first update Δ^s is obtained by projecting the gap vector $\widetilde{U}_0^s - U$ onto $T_{\widetilde{U}}$ and then further onto T_U by following the geodesic "backward in time"; see steps 9–11 in Algorithm 1. At every time instant t, it holds that

$$\widetilde{U}^s(t) = \mathrm{Exp}_U^\alpha(t\Delta_0) = \begin{pmatrix} U & | & Q \end{pmatrix} \begin{pmatrix} M_t & X_t \\ N_t & Y_t \end{pmatrix} \begin{pmatrix} I_p \\ 0 \end{pmatrix} = UM_t + QN_t,$$

where $\left(\begin{smallmatrix} M_t & X_t \\ N_t & Y_t \end{smallmatrix}\right)$ is evaluated according to (12) with inputs $A = tA_0$, $B = tR_0$.

If any matrix of the form $W = UX + QY$ is projected onto the tangent space at any $UM_t + QN_t$, the result is

$$\begin{aligned}\Pi_{UM_t+QN_t}(W) &= U\left(X - M_t\,\mathrm{sym}\left(M_t^TX + N_t^TY\right)\right) + Q\left(Y - N_t\,\mathrm{sym}\left(M_t^TX + N_t^TY\right)\right) \\ &= UX^\Pi + QY^\Pi.\end{aligned}$$

Hence, all operations performed in Algorithm 1 take place in the matrix space $\mathcal{S} = \{UX + QY | X, Y \in \mathbb{R}^{p\times p}\}$ of (16). ∎

Algorithm 2. p-shooting method

Input: Stiefel matrices $U, \widetilde{U} \in St(n,p)$, convergence threshold $\epsilon > 0$, array $T = [t_0, t_1, \ldots, t_m], t_0 = 0, t_m = 1.0$ (discretized unit interval), metric parameter α.

1: $\widehat{M} = U^T \widetilde{U}$

2: $Q\widehat{N} = \widetilde{U} - U\widehat{M}$ {compact QR-decomposition}

3: $\gamma \leftarrow \sqrt{\|\widehat{M} - I_p\|^2 + \|\widehat{N}\|^2}$ {note $\widetilde{U} - U = U(\widehat{M} - I_p) + Q\widehat{N}$}

4: $A \leftarrow \frac{\gamma \operatorname{skew}(\widehat{M})}{\sqrt{\|\operatorname{skew}(\widehat{M})\|^2 + \|\widehat{N}\|^2}}, \quad R \leftarrow \frac{\gamma \widehat{N}}{\sqrt{\|\operatorname{skew}(\widehat{M})\|^2 + \|\widehat{N}\|^2}},$

5:

6: **while** $\gamma > \epsilon$ **do**

7: **for** $j = 1, \ldots, m$ **do**

8: $\begin{pmatrix} M(t_j) \\ N(t_j) \end{pmatrix} \leftarrow \exp_m\left(t_j \begin{pmatrix} \frac{1}{\alpha+1} A & -R^T \\ R & 0 \end{pmatrix}\right) \begin{pmatrix} \exp_m\left(t_j \left(\frac{\alpha}{\alpha+1} A\right)\right) \\ 0 \end{pmatrix}$ {cf. (12)}

9: **end for** {p-factor representation of geodesic $U^s(t_j) = UM(t_j) + QN(t_j)$}

10: $A^s \leftarrow M(t_m) - \widehat{M}$, $R^s \leftarrow N(t_m) - \widehat{M}$ {p-factors of current gap vector}

11: $\gamma \leftarrow \sqrt{\|A^s\|^2 + \|R^s\|^2}$

12: **for** $j = m, \ldots, 0$ **do**

13: $[A^s, R^s] = \text{ParaTrans\_pFactors}(M(t_j), N(t_j), A^s, R^s, \gamma)$ {initial projection plus approximate parallel transport}

14: **end for**

15: update $A \leftarrow A - A^s, \quad R \leftarrow R - R^s$ {updated $\Delta = UA + QR$}

16: **end while**

Output: $\Delta = UA + QR$

With Proposition 5, the computational costs associated with the shooting method can be reduced considerably. This gives rise to Algorithm 2. A few remarks on this procedure are in order: Observe that step 8 of Algorithm 2 is the only step that depends on the chosen metric and thus makes the only difference when computing the Stiefel logarithm for the canonical, the Euclidean, or any other α-metric. The subroutine ParaTrans_pFactors$(M(t), N(t), A^s, R^s, \gamma)$ that appears in step 13 of Algorithm 2 realizes an approximation of the parallel transport of $\Delta^s = UA^s + QR^s$ along the geodesic using the unique representation with the p-factors A^s, R^s. This subroutine is detailed in Algorithm 3. A numerical experiment that illustrates the performance of Algorithm 2 is given in section 4.

3.4. An improved algebraic Stiefel logarithm for the canonical metric. Let $U, \widetilde{U} = UM + QN \in St(n,p)$ with $M = U^T\widetilde{U}, QN = (I - UU^T)\widetilde{U}$ and an orthonormal completion $\left(\begin{smallmatrix} M & X_0 \\ N & Y_0 \end{smallmatrix}\right) \in SO(2p)$ as introduced in the previous sections. In [39], it has been shown that for solving (14) in the case of the canonical metric, it is sufficient to find $\Gamma \in \operatorname{skew}(p)$ such that

$$(17) \qquad \begin{pmatrix} 0 \mid I_p \end{pmatrix} \log_m\left(\begin{pmatrix} M & X_0 \\ N & Y_0 \end{pmatrix} \begin{pmatrix} I_p & 0 \\ 0 & \exp_m(\Gamma) \end{pmatrix}\right) \begin{pmatrix} 0 \\ I_p \end{pmatrix} = 0 \in \mathbb{R}^{p \times p}.$$

With $V = \left(\begin{smallmatrix} M & X_0 \\ N & Y_0 \end{smallmatrix}\right) = \exp_m\left(\begin{smallmatrix} A_0 & -B_0^T \\ B_0 & C_0 \end{smallmatrix}\right)$ and $W = \exp_m\left(\begin{smallmatrix} 0 & 0 \\ 0 & \Gamma \end{smallmatrix}\right)$, (17) requires finding Γ such that the lower p-by-p diagonal block of the matrix $\log_m(VW)$ vanishes. Let $[V, W] = VW - WV$ denote the matrix commutator. The BCH series for the matrix logarithm is

Algorithm 3. ParaTrans_pFactors: Map the tangent vector $\Delta = UA_1 + QR_1 \in T_{U_1}$ to the tangent space T_{U_2}, preserve the length

Input: $M_2, N_2, A_1, R_1 \in \mathbb{R}^{p\times p}$, $\gamma > 0$

{here, M_2, N_2 represent $U_2 = UM_2+QN_2 \in St(n,p)$, A_1, R_1 represent $\Delta = UA_1+QR_1 \in T_{U_1}$.}

1: $S \leftarrow \mathrm{sym}(M_2^T A_1 + N_2^T R_1)$
2: $A_2 = A_1 - M_2 S$, $R_2 = R_1 - N_2 S$
3: $l = \sqrt{\|A_2\|^2 + \|R_2\|^2}$
4: **if** $l > \epsilon$ **then**
5: $A_2 = \frac{\gamma}{l} A_2$, $R_2 = \frac{\gamma}{l} R_2$ {rescale to original length}
6: **else**
7: $A_2 = 0$, $R_2 = 0$.
8: **end if**

Output: A_2, R_2

This algorithm executes the same operation as in step 10 of Algorithm 1 but on the representative $p \times p$ matrix factors.

$$\log_m(VW) = \log_m(V) + \log_m(W) + \frac{1}{2}[\log_m(V), \log_m(W)]$$
$$+ \frac{1}{12}\left([\log_m(V), [\log_m(V), \log_m(W)]] + [\log_m(W), [\log_m(W), \log_m(V)]]\right) + \cdots;$$

see [29, section 1.3, p. 22]. The algorithm [39, Algorithm 1] and the associated convergence analysis rely on the fact that with the choice of $\Gamma_0 = -C_0$, the lower p-by-p block of $\log_m(VW)$ vanishes up to terms of third order in the BCH series. The algorithm iterates on this observation and produces the sequence

$$\begin{pmatrix} A_{k+1} & -B_{k+1}^T \\ B_{k+1} & C_{k+1} \end{pmatrix} := \log_m\left(\exp_m\begin{pmatrix} A_k & -B_k^T \\ B_k & C_k \end{pmatrix}\exp_m\begin{pmatrix} 0 & 0 \\ 0 & \Gamma_k \end{pmatrix}\right) \tag{18}$$

with $\Gamma_k = -C_k$. It is guaranteed that $\|C_k\| \to 0$ for $k \to \infty$ at a linear rate as long as the input points $U, \widetilde{U}$ are close enough; see [39, Theorem 4.1].

In fact, up to commutator products of order three or higher, the BCH series expansion of the lower diagonal block of $\log_m(V_k W)$ is

$$C_k + \Gamma + \frac{1}{2}(C_k\Gamma - \Gamma C_k)$$
$$+\frac{1}{12}\left(C_k^2\Gamma + \Gamma C_k^2 + C_k\Gamma^2 + \Gamma^2 C_k - \left(B_k B_k^T\Gamma + \Gamma B_k B_k^T\right) - 2\left(C_k\Gamma C_k + \Gamma C_k\Gamma\right)\right)$$
$$= C_k + \Gamma + \frac{1}{2}(C_k\Gamma - \Gamma C_k) - \frac{1}{12}\left(B_k B_k^T\Gamma + \Gamma B_k B_k^T\right) + \text{h.o.t.},$$

where "h.o.t." comprises the higher-order terms in the BCH series. Ignoring the quadratic terms $\Gamma C_k, C_k\Gamma$ and the higher ones and setting the above expression to zero yields a symmetric Sylvester equation for Γ:

$$C_k = S_k\Gamma + \Gamma S_k \quad \text{with } S_k := \frac{1}{12}B_k B_k^T - \frac{1}{2}I_p. \tag{19}$$

A sufficient criterion that guarantees a unique solution is that $\|B_k\|_2 < \sqrt{6}$, since in this case $\frac{1}{6}\|B_k B_k^T\|_2 < 1$, which entails that all eigenvalues of S_k are strictly negative.

This in turn yields that S_k and $-S_k$ have disjoint spectra, which ensures the unique solvability of (19) [6, section VII.2]. When selecting Γ_k as the solution to (19) at each iteration k, then the lower p-by-p diagonal block of the BCH series vanishes up to fourth-order commutator terms and terms that are quadratic in C_k and Γ_k. This idea was first presented in [41].

Remark 6. This choice does not cancel all terms that are of first order in Γ in the BCH series of $\log_m(VW)$. The series is ordered by the degree of nested commutator brackets. In the jth-order commutator term, "words" formed by j "letters" of the two-letter alphabet $\{\log_m(V), \log_m(W)\}$ appear, where each of $\log_m(V), \log_m(W)$ appears at least once; see [36, 24, 38]. This means that no matter where we cut off the tail of the BCH series, there remain terms that are linear in $\|\Gamma\|$ (or $\|C_k\|$ for that matter). Expanding this series for $\log_m(VW)$ in powers of W as in [23, section 8.1] will not solve the issue, because the collection of all terms that are of first order in $\|W\|$ constitutes an infinite series by itself that needs to be truncated in numerical applications.

In the numerical implementation of Algorithm 4, we introduce a Boolean "Flag_Sylv on/off" to switch between the original version [39, Algorithm 1], which works with $\Gamma_k = -C_k$, and the "Sylvester-enhancement," which selects Γ_k as the solution to (19). A detailed convergence analysis can be conducted as in [39] but is not worthwhile in the context of this work. Yet, the next proposition allows us to compare the asymptotic convergence rate of Algorithm 4 with "Flag_Sylv" switched off, which is [39, Algorithm 1] and Algorithm 4 with "Flag_Sylv" switched on, which is based on solving (19).

PROPOSITION 7. *Suppose that the input data $U \neq \widetilde{U} \in St(n,p)$ are such that Algorithm 4 converges with "Flag_Sylv" switched on. Assume further that there is a*

Algorithm 4. Improved algebraic Stiefel logarithm, canonical metric ($\alpha = 0$).

Input: $U, \widetilde{U} \in St(n,p)$, $\epsilon > 0$ convergence threshold, Boolean "Flag_Sylv on/off"

$M := U^T\widetilde{U} \in \mathbb{R}^{p\times p}$

$QN := \widetilde{U} - UM \in \mathbb{R}^{n\times p}$ {compact QR-decomp.}

$V_0 := \begin{pmatrix} M & X_0 \\ N & Y_0 \end{pmatrix} \in O_{2p\times 2p}$ {orthonormal completion}

for $k = 0, 1, 2, \ldots$ **do**

  $\begin{pmatrix} A_k & -B_k^T \\ B_k & C_k \end{pmatrix} := \log_m(V_k)$ {matrix log, A_k, C_k skew}

  if $\|C_k\|_2 \leq \epsilon$ **then**

    break

  end if

  if Flag_Sylv **then**

    $S_k := \frac{1}{12}B_kB_k^T - \frac{1}{2}I_p$

    solve $C_k = S_k\Gamma + \Gamma S_k$ for Γ {sym. Sylvester equation}

  else

    $\Gamma := -C_k$ {cancel first term in BCH series}

  end if

  $\Phi_k := \exp_m(\Gamma)$ {matrix exp, Φ_k orthogonal}

  $V_{k+1} := V_kW_k$, where $W_k := \begin{pmatrix} I_p & 0 \\ 0 & \Phi_k \end{pmatrix}$ {update}

end for

Output: $\Delta := \mathrm{Log}_U^{St}(\widetilde{U}) = UA_k + QB_k \in T_USt(n,p)$

bound $0 < \delta < 1$ *such that for* $\log_m(V_k) = \left(\begin{smallmatrix} A_k & -B_k^T \\ B_k & C_k \end{smallmatrix}\right)$, *it holds that* $\|\log_m(V_k)\|_2 < \delta$ *throughout the algorithm's iteration loop.*[3] *Then, for* k *large enough, it holds that*

$$\|C_{k+1}\|_2 \leq \left(\frac{6}{6-\delta^2}\frac{\delta^4}{1-\delta} + \mathcal{O}(\|C_k\|_2)\right)\|C_k\|_2.$$

This implies the asymptotic convergence rate of Algorithm 4 *with "Flag_Sylv" switched on for* $k \to \infty$. *This compares to the asymptotic rate of* [39, *Algorithm* 1], *which according to* [39, *equation* (12)] *is bounded by*

$$\|C_{k+1}\|_2 \leq \left(\frac{1}{6}\delta^2 + \frac{\delta^4}{1-\delta} + \mathcal{O}(\|C_k\|_2)\right)\|C_k\|_2.$$

Proof. Since $\|B_k\|_2 \leq \|\log_m(V_k)\| < \delta < 1$, the matrices $S_k = \frac{1}{12}B_kB_k^T - \frac{1}{2}I_p$ are negative definite with largest eigenvalue bounded by $-\frac{1}{2} + \frac{\delta^2}{12} = -\frac{6-\delta^2}{12}$. Hence, the spectra of the symmetric matrices S_k and $-S_k$ are separated by a vertical strip of width $\frac{6-\delta^2}{6}$ in the complex plain. Applying [6, Theorem VII.2.12] to the Sylvester equation $S_k\Gamma + \Gamma(-S_k) = C_k$ yields $\|\Gamma\|_2 \leq \frac{6}{6-\delta^2}\|C_k\|_2$. Calculating C_{k+1} according to (18) but with this choice of Γ shows that all terms up to order four in the BCH series are at least quadratic in $\|C_k\|_2$:

$$\begin{aligned} C_{k+1} = {} & \frac{1}{2}[C_k,\Gamma] + \frac{1}{12}([C_k^2,\Gamma] + [C_k,\Gamma^2]) - 2(C_k\Gamma C_k + \Gamma C_k\Gamma) \\ & + \frac{1}{24}\left([B_kB_k^T,\Gamma^2] - (C_k^2\Gamma^2 + \Gamma^2C_k^2) + 2[C_k\Gamma C_k,\Gamma]\right) + \text{h.o.t.}(5), \end{aligned}$$

where h.o.t.(5) are the terms of fifth order and higher in the BCH series. From $U \neq \widetilde{U}$, we get $\lim_{k\to\infty}\left(\begin{smallmatrix} A_k & -B_k^T \\ B_k & C_k \end{smallmatrix}\right) \neq 0$. Hence, for k large enough, it holds that $\|\Gamma\|_2 \leq \frac{6}{6-\delta^2}\|C_k\|_2 \leq \|\left(\begin{smallmatrix} A_k & -B_k^T \\ B_k & C_k \end{smallmatrix}\right)\|_2 = \|\log_m(V_k)\|_2$. Because C_{k+1} is but the lower diagonal subblock of $\log_m(V_kW)$, the higher-order terms are bounded by $\|\text{h.o.t}(5)\|_2 \leq \sum_{l=5}^{\infty}\|\log_m(V_k)\|_2^{l-1}\|\Gamma\|_2 \leq \frac{6}{6-\delta^2}\|C_k\|_2\frac{\delta^4}{1-\delta}$; see [39, Lemma A.1]. The claim is now a straightforward consequence. ∎

Proposition 7 shows that with smaller values of δ, the Sylvester approach becomes more and more favorable. The actual value of δ depends on how close the inputs $U, \widetilde{U}$ are. For $\delta \approx 0.7147$ the bounds for the asymptotic convergence rates are the same for both the approaches "Flag_Sylv on/off"; for $\delta \approx 0.3286$ the rate of the Sylvester-based method is improved a factor of 2 and by a factor of 10 for $\delta \approx 0.1270$. Note that in both cases, the bounds overestimate the true convergence rates.

The main computational effort of Algorithm 4 is in the computation of the matrix logarithm in step 5. This can be achieved efficently (and without resorting to complex numbers arithmetics) by first computing a real Schur form. Assuming that the principal matrix logarithm is properly defined, the Schur form of an orthogonal matrix is block-diagonal with blocks (1) of size (1×1) and (2×2)-blocks of the form $\left(\begin{smallmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{smallmatrix}\right)$, the matrix logarithm of such a block being $\left(\begin{smallmatrix} 0 & -\varphi \\ \varphi & 0 \end{smallmatrix}\right)$. This is exploited in the actual implementation.

[3] For the original Stiefel log algorithm, conditions for the existence of such a global bound δ are established in [39, Lemma 4.4]. Similar techniques apply in the present context.

3.5. A geodesic Newton method for the Stiefel logarithm. The nonlinear matrix equation (14) can be cast in the following form:

$$V^T F \begin{pmatrix} A & -B^T \\ B & C \end{pmatrix} = I, \quad \text{where } V = \begin{pmatrix} M & X_0 \\ N & Y_0 \end{pmatrix} \in SO(2p).$$

For brevity, we introduce

$$\widehat{F} : \operatorname{skew}(2p) \to SO(2p), \quad S \mapsto V^T F(S).$$

The parameter domain $\operatorname{skew}(2p)$ has a vector space structure that allows us to employ Euclidean techniques. The co-domain, however, is the Lie group $SO(2p)$. Ignoring the structure of $SO(2p)$, the classical Newton method can be applied to the root finding problem $V^T F(S) - I = 0$. Given a starting point S_0, the Newton method requires solving the Taylor-linearized problem

$$I = \widehat{F}(S) + D\widehat{F}_S(H) \approx \widehat{F}(S + H)$$

so that the update H is determined by the linear system $D\widehat{F}_S(H) = I - \widehat{F}(S)$. Yet, the left-hand side and the right-hand side are not compatible, as $\widehat{F}(S) \in SO(2p)$ and $D\widehat{F}_S(H) \in T_{\widehat{F}(S)} SO(2p) = \widehat{F}(S) \operatorname{skew}(2p)$.

A remedy is to replace the Euclidean first-order Taylor approximation, which can be thought of as progressing along a straight line, by moving along a geodesic in the same direction. This leads to a first-order approximation that preserves the Riemannian structure

$$\widehat{F}(S + H) \approx \operatorname{Exp}_{\widehat{F}(S)}(D\widehat{F}_S(H)).$$

On $SO(2p)$, the geodesic that starts from $\widehat{F}(S)$ in the direction of $D\widehat{F}_S(H)$ is

$$\operatorname{Exp}_{\widehat{F}(S)}(D\widehat{F}_S(H)) = \widehat{F}(S) \exp_m(\widehat{F}(S)^T D\widehat{F}_S(H));$$

see, e.g., [13]. The first-order approximation to (14) becomes

$$I = \widehat{F}(S) \exp_m(\widehat{F}(S)^T D\widehat{F}_S(H)) \quad \Leftrightarrow \quad \underbrace{\widehat{F}(S)^T D\widehat{F}_S(H)}_{\text{skew}} = \underbrace{\log_m(\widehat{F}(S)^T)}_{\text{skew}}. \tag{20}$$

The left-hand side is a linear operator

$$L_S : \operatorname{skew}(2p) \to \operatorname{skew}(2p), \quad H \mapsto \widehat{F}(S)^T D\widehat{F}_S(H).$$

In summary, this leads to the iterative scheme stated in Algorithm 5. Upon convergence, $S_k = \left(\begin{smallmatrix} A_k & -B_k^T \\ B_k & C_k \end{smallmatrix}\right)$ is found and the output $\Delta := \operatorname{Log}_U^{St}(\widetilde{U}) = UA_k + QB_k \in T_U$ is formed with U, Q as in Algorithm 4. Computationally, the expensive part is to evaluate the operator L_S. For convenience, let us restrict to the Euclidean metric and write

$$S = \begin{pmatrix} A & -B^T \\ B & C \end{pmatrix}, \quad S_{\text{tri}} = \begin{pmatrix} 2A & -B^T \\ B & 0 \end{pmatrix}, \quad S_{\text{di}} = \begin{pmatrix} A & 0 \\ 0 & C \end{pmatrix},$$

and likewise for $H \in \operatorname{skew}(2p)$. It holds that

$$\begin{aligned} L_S(H) &= \widehat{F}(S)^T D\widehat{F}_S(H) = F(S)^T V V^T DF_S(H) \\ &= F(S)^T \left(D(\exp_m)_{S_{\text{tri}}}(H_{\text{tri}}) \exp_m(-S_{\text{di}}) + \exp_m(S_{\text{tri}}) D(\exp_m)_{S_{\text{di}}}(-H_{\text{di}}) \right). \end{aligned} \tag{21}$$

Algorithm 5. GeoNewton for solving (14)

```
k = 0
while ‖log_m(F̂(S_k)^T)‖_F > τ do
    solve L_{S_k}(H_k) = log_m(F̂(S_k)^T)
    update S_{k+1} = S_k + H_k
    k = k + 1
end while
```

In an implementation, we do not actually form this operator, but implement its action on a matrix H. This is then used in a matrix-free version of the GMRES algorithm [30] (as preinstalled in MATLAB, version R2019b). For the practical evaluation of $D(\exp_m)_S(H)$, we use Mathias' theorem [15, Theorem 3.6], which simultaneously gives $\exp_m(S)$. An equivalent alternative to (14) is to solve

$$0 = \log_m(F(S)) - \log_m(V). \tag{22}$$

In this form, both the unknown S and the output $\log_m(F(S)) - \log_m(V)$ are in the vector space skew$(2p)$ so that this equation is amenable to being treated with the classical Newton method. However, computing or approximating the derivative remains the computational bottleneck. The associated linear operator now involves differentials of both the matrix exponential and the matrix logarithm. We tackle this with the same strategy as above by relying on Mathias' theorem and the matrix-free GMRES method. The matrix functions $\exp_m(S)$ and $\log_m(S)$ may be approximated via the Cayley transformation $\mathrm{Cay}(S) = (I - \frac{1}{2}S)^{-1}(I + \frac{1}{2}S)$ and its inverse; likewise the differentials of $D(\exp_m)_S(H)$ and $D(\log_m)_S(H)$ may be approximated with the differentials of the corresponding Cayley transformations.

We mention these approaches for the sake of completeness and include the algorithms based on (20) and (22) in the algorithmic competition. However, the numerical experiments show that none of the above approaches can compete, either with the p-shooting method Algorithm 2 or with the algebraic Stiefel log algorithm Algorithm 4. This also holds when the Cayley transformations are used to replace $\exp_m$ and $\log_m$. Therefore, we omit a detailed discussion.

4. Numerical experiments. In this section, we conduct various numerical experiments for assessing the performance of the proposed approaches to solve the local geodesic endpoint problem. All experiments are performed with MATLAB R2019b on a Linux 64bit HP notebook with four Intel Core i7-5600U 2.60GHz CPUs.[4]

4.1. The geodesic endpoint problem for the canonical metric. We create two points $U, \widetilde{U}$ pseudorandomly on $St(n,p)$, but such that they are a prescribed geodesic distance apart. More precisely, we construct U from a QR-decomposition of a random $(n \times p)$-matrix with entries sampled from the uniform distribution. Then, we create a random tangent vector $\Delta = UA + (I - UU^T)T$, where $A \in \mathbb{R}^{p\times p}$ is random but skew and $T \in \mathbb{R}^{n\times p}$ is random. Then, Δ is scaled to the prescribed length according to the Riemannian metric and $\widetilde{U} \in St(n,p)$ is obtained as $\widetilde{U} = \mathrm{Exp}_U(\Delta)$.

Then, the various Stiefel-log algorithms are applied to compute the reconstructed tangent vector Δ_{rec} and the absolute accuracy is checked in the infinity-matrix norm. In summary, we perform the following three steps:

$$(a)\quad \widetilde{U} \leftarrow \mathrm{Exp}_U(\Delta), \quad (b)\quad \Delta_{\mathrm{rec}} \leftarrow \mathrm{Log}_U(\widetilde{U}), \quad (c)\quad \text{compute } \|\Delta - \Delta_{\mathrm{rec}}\|_\infty.$$

[4]The MATLAB code and an accompanying Python implementation are available at https://github.com/RalfZimmermannSDU/RiemannStiefelLog/tree/main/Stiefel_log_general_metric/.

We perform 10 runs with random data, record the numerical accuracy, the iteration count, and the computation time, and average over the results. In order to evaluate the logarithm, we use the following methods:

- Algorithm 4 as in [39],
- Algorithm 4 enhanced by the Sylvester equation (19) as detailed in subsection 3.4,
- Algorithm 4 enhanced by (19) and with the Cayley transformation replacing the matrix exponential in step 15,
- Algorithm 2 on two time steps $\{0.0, 1.0\}$ in the unit interval $[0.1]$,
- Algorithm 2 on four time steps $\{0.0, 0.\bar{3}, 0.\bar{6}, 1.0\}$ of the unit interval $[0, 1]$,
- Algorithm 1 on two time steps $\{0.0, 1.0\}$ in the unit interval $[0.1]$,
- the single shooting method of [34, section 2.3] that is based on a Newton root finding problem under the canonical metric.

In each case, the convergence threshold is set to $\tau = 10^{-11}$. Table 1 displays the results for data on $St(n = 2000, p = 500)$ with $\mathrm{dist}(U, \widetilde{U}) = 5\pi$, for data on $St(n = 120, p = 30)$ with $\mathrm{dist}(U, \widetilde{U}) = \pi$, and for data on $St(n = 12, p = 3)$ with $\mathrm{dist}(U, \widetilde{U}) = 0.95\pi$.

TABLE 1
Numerical performance for the cases considered in subsection 4.1.

(Section 4.1) **Test Case 1:** random data $n = 2000$, $p = 500$, canonical metric, 5 runs			
$\mathrm{dist}(U, \widetilde{U}) = 5\pi$			
Method	av. rel. error $\|\Delta - \Delta_{rec}\|_\infty$	av. iter. count	av. time
Alg. 4	$0.50 \cdot 10^{-11}$	13.0	13.00s
Alg. 4+Sylv.	$0.29 \cdot 10^{-12}$	7.0	8.29s
Alg. 4+Sylv.+Cay.	$0.29 \cdot 10^{-12}$	7.0	8.27s
Alg. 2 on 2 steps	$0.49 \cdot 10^{-11}$	41.0	25.53s
Alg. 2 on 4 steps	$0.61 \cdot 10^{-11}$	35.0	53.89s
Alg. 1 on 2 steps	$0.51 \cdot 10^{-11}$	39.8	34.06s
Single shooting [34, section 2.3]	unfeasible due to memory overflow		

(Section 4.1) **Test Case 2:** random data $n = 120$, $p = 30$, canonical metric, 10 runs			
$\mathrm{dist}(U, \widetilde{U}) = \pi$			
Method	av. rel. error $\|\Delta - \Delta_{rec}\|_\infty$	av. iter. count	av. time
Alg. 4	$0.226 \cdot 10^{-11}$	10.2	0.027s
Alg. 4+Sylv.	$0.159 \cdot 10^{-11}$	5.0	0.018s
Alg. 4+Sylv.+Cay.	$0.160 \cdot 10^{-11}$	5.0	0.018s
Alg. 2 on 2 steps	$0.291 \cdot 10^{-11}$	26.8	0.033s
Alg. 2 on 4 steps	$0.193 \cdot 10^{-11}$	24.7	0.064s
Alg. 1 on 2 steps	$0.281 \cdot 10^{-11}$	26.4	0.032s
Single shooting [34, section 2.3]	$0.58 \cdot 10^{-14}$ (results for 1 run.)	5	524.6s

(Section 4.1) **Test Case 3:** random data $n = 12$, $p = 3$, canonical metric, 100 runs			
$\mathrm{dist}(U, \widetilde{U}) = 0.95\pi$ (averaging only over the converged runs)			
Method	av. rel. error $\|\Delta - \Delta_{rec}\|_\infty$	av. iter. count	av. time
Alg. 4	$0.62 \cdot 10^{-10}$ (1 run not conv'd)	120.3	0.046s
Alg. 4+Sylv.	$0.50 \cdot 10^{-10}$ (1 run not conv'd)	41.1	0.023s
Alg. 4+Sylv.+Cay.	$0.53 \cdot 10^{-10}$ (1 run not conv'd)	41.3	0.021s
Alg. 2 on 2 steps	(all 100 runs not conv'd)	–	–
Alg. 2 on 4 steps	$0.80 \cdot 10^{-10}$ (all 100 runs conv'd.)	212.2	0.031s
Alg. 1 on 2 steps	(all 100 runs not conv'd)	–	–
Single shooting [34, section 2.3]	$0.42 \cdot 10^{-14}$ (41 runs not conv'd.)	7.93	0.021s

Example plots of the convergence histories are shown in Figures SM1 and SM2 of the supplements, respectively.

The table shows that Algorithm 4 with "Flag_Sylv on" exhibits the best performance for the cases considered in regard to the computation time. In terms of the numerical accuracy it is outranked by the Newton-based single shooting method of [34, section 2.3], provided that the latter converges. It can also be seen that a subdivision of the interval $[0, 1]$ is not required for Algorithm 2 in order to converge for the larger data sets under consideration. The dimensions and distance for the low-dimensional data set on $St(12, 3)$ are chosen as in [35], where the global geodesic endpoint problem is considered. Recall that the estimated injectivity radius of $St(n, p)$ under the canonical metric is at least 0.89π and most likely not larger; see [28]. In fact, the methods considered here are local by nature. As expected, in the experiments, we observe convergence in some cases and divergence in other cases; see Table 1 for details. Algorithm 2 on four time steps in $[0, 1]$ converges in all cases considered, while it diverges in all cases, if only the boundary points of $[0, 1]$ are considered in the discrete parallel transport.

4.2. The geodesic endpoint problem for the Euclidean metric. In this section, we repeat the experiments of subsection 4.1 with exactly the same set-up, but for the Euclidean metric. Since the algebraic Stiefel log algorithm Algorithm 4 is not available for metrics other than the canonical one, we juxtapose the following methods:

- Algorithm 5, "GeoNewton," the geodesic Newton method for (14);
- Algorithm "EucNewton," based on the classical Newton method for solving (22);
- Algorithm 2 on two time steps $\{0.0, 1.0\}$ of the unit interval $[0.1]$;
- Algorithm 2 on four time steps $\{0.0, 0.\bar{3}, 0.\bar{6}, 1.0\}$ of the unit interval $[0, 1]$.

In each case, the convergence threshold is set to $\tau = 10^{-11}$. Table 2 displays the results for data on $St(n = 120, p = 30)$ with $\mathrm{dist}(U, \widetilde{U}) = \pi$ and for data on $St(n = 2000, p = 500)$ with $\mathrm{dist}(U, \widetilde{U}) = 5\pi$. In the latter case, only one random run is performed due to the large computation times for the Newton methods. Example plots of the convergence histories are shown in Figures SM4 and SM5 of the supplements, respectively.

TABLE 2
Numerical performance for the cases considered in subsection 4.2.

(Section 4.2) **Test Case 1:** random data $n = 2000$, $p = 500$, Euclidean metric, 1 run $\mathrm{dist}(U, \widetilde{U}) = 5\pi$			
Method	rel. error $\|\Delta - \Delta_{rec}\|_\infty$	iter. count	time
Alg. 5 GeoNewton	$0.61 \cdot 10^{-11}$	13	958.2s
Alg. EucNewton	$0.23 \cdot 10^{-11}$	6	890.1s
Alg. 2 on 2 steps	$0.26 \cdot 10^{-11}$	20	10.5s
Alg. 2 on 4 steps	$0.36 \cdot 10^{-11}$	11	14.8s

(Section 4.2) **Test Case 2:** random data $n = 120$, $p = 30$, Euclidean metric, 10 runs $\mathrm{dist}(U, \widetilde{U}) = \pi$			
Method	av. rel. error $\|\Delta - \Delta_{rec}\|_\infty$	av. iter. count	av. time
Alg. 5 GeoNewton	$0.11 \cdot 10^{-11}$	8.0	0.34s
Alg. EucNewton	$0.071 \cdot 10^{-11}$	5.0	0.85s
Alg. 2 on 2 steps	$0.078 \cdot 10^{-11}$	13.1	0.016s
Alg. 2 on 4 steps	$0.12 \cdot 10^{-11}$	9.0	0.030s

The table shows that Algorithm 2 without a subdivision of $[0,1]$ exhibits the best performance in terms of the computation time for the cases considered here. For the test case with data on $St(2000,500)$, the method is ca. 90 times faster than the Newton-based methods and also much more memory efficient, since no large linear operators need to be constructed. These computation times are representative for other values of α in the family of Riemannian metrics.

4.3. Investigations on the parametric dependencies. In this section, the performance of the proposed methods for computing $\mathrm{Log}_U^\alpha(\widetilde{U})$ with $U,\widetilde{U} \in St(n,p)$ is investigated under changes in the metric parameter α, the Riemannian distance of the input points $\mathrm{dist}(U,\widetilde{U})$ as well as the matrix dimensions n and p.

We start with assessing the performance of Algorithm 2 for metric parameters $\alpha \in (-1,\infty)$, where the associated metric is Riemannian. (For $\alpha \in (-\infty,-1)$, the metric becomes pseudo-Riemannian; see [17, section 5.5].) To this end, we fix the dimensions $n = 200$, $p = 50$ and construct pseudorandom data $U \in St(200,50)$, $\Delta_0 \in T_U St(200,50)$ as described in subsection 4.1. We discretize the parameter interval $[-0.9, 5.0]$ with equidistant steps of size 0.05. For each value $\alpha \in \{-0.9 + 0.05j | j = 0,\ldots,118\}$, we compute $\Delta(\alpha) = \frac{d}{\|\Delta_0\|_\alpha}\Delta_0$, so that $\Delta(\alpha)$ is normalized to a length of d according to the α-metric. As a distance factor, we use $d = 0.5\pi$. Then, we set $\widetilde{U} = \mathrm{Exp}_U^\alpha(\Delta(\alpha))$ with the Stiefel exponential computed according to (11). In this way, a Stiefel data pair $U,\widetilde{U}$ with $\mathrm{dist}_\alpha(U,\widetilde{U}) = d = 0.5\pi$ is obtained. We apply Algorithm 2 to compute $\mathrm{Log}_U^\alpha(\widetilde{U})$ up to a convergence threshold of $\tau = 10^{-11}$ and record the wall clock computation time as well as the iteration count. The results are displayed in Figure 2. The supplements feature an analog experiment with data on $St(2000,200)$; see Figure SM6.

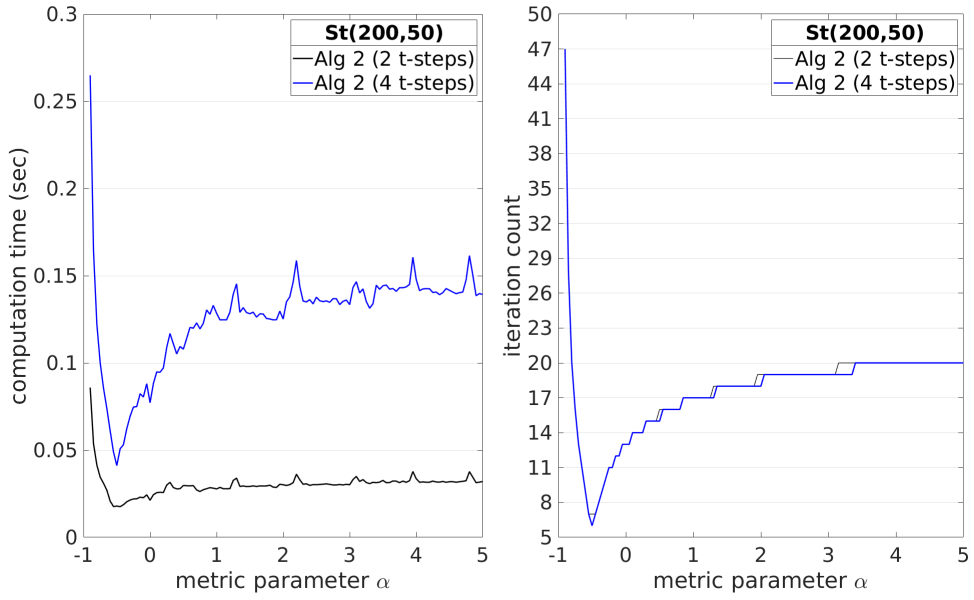

FIG. 2. *(cf. subsection 4.3) Wall clock computation time in seconds versus α (left) and iteration count until convergence versus α (right) for computing $\mathrm{Log}_U^\alpha(\widetilde{U})$ with Algorithm 2. The trial parameter set is $\{\alpha = -0.9 + 0.05j \mid j = 0,\ldots,118\}$. The data points $U,\widetilde{U} \in St(200,50)$ are at a Riemannian α-distance of $\mathrm{dist}_\alpha(U,\widetilde{U}) = 0.5\pi$. Timing results are averaged over 100 runs.*

In both test cases, the iteration count is minimal precisely at $\alpha = -\frac{1}{2}$, which corresponds to the Euclidean metric. We conjecture that the dependency of the iteration count on the metric parameter α is curvature-related. Recall that the curvature of a Riemannian manifold depends on the metric. Furthermore, observe that in the flat matrix space $\mathbb{R}^{n\times p}$, the shooting method produces the correct solution after one single iteration, because the geodesics are straight lines so that $\mathrm{Exp}_U(t\Delta) = U + t\Delta$. Following this line of reasoning, the experimental results suggest that at least locally around the generic pseudorandom data points, $(St(n,p), \langle\cdot,\cdot\rangle^\alpha)$ is least curved for the induced Euclidean metric, where $\alpha = -\frac{1}{2}$.

Next, we assess the dependency of Algorithms 2 and 4 on the distance of the input parameters $U, \widetilde{U}$ under the canonical metric. As in subsection 4.1, we construct $U \in St(n,p)$ and $\Delta_0 \in T_U St(n,p)$. Then, we scale $\Delta(d) = \frac{d}{\|\Delta_0\|_0}\Delta_0$ and set $\widetilde{U} = \mathrm{Exp}^0_U(\Delta(d))$ so that by construction, $\mathrm{dist}(U,\widetilde{U}) = d$ with respect to the canonical metric ($\alpha = 0$). We consider distance factors of $d = 0.5\pi, 1.0\pi, \ldots, 4.5\pi$ and measure the wall clock time and the iteration count until the numerical convergence measure drops below a threshold parameter of $\tau = 10^{-11}$. The results are displayed in Figure 3. It can be seen that for Algorithm 4, the iteration count and wall clock time grow moderately with increasing distance. In contrast, time and iteration count associated with Algorithm 2 on two time steps exhibit a strong nonlinear dependency on the distance. For Algorithm 2 on four time steps, the growth in time and number of iterations is roughly linear but with a steeper slope when compared to Algorithm 4.

Last, we expose the dependency of Algorithms 1, 2, and 4 on the dimensions n and p. Again, we resort to the canonical metric ($\alpha = 0$). We fix p to a value of $p = 200$ and vary $n = 1000 \cdot 2^j$ with $j = 1, \ldots, 8$. Pseudorandom data $U, \widetilde{U} \in St(n,p)$ with

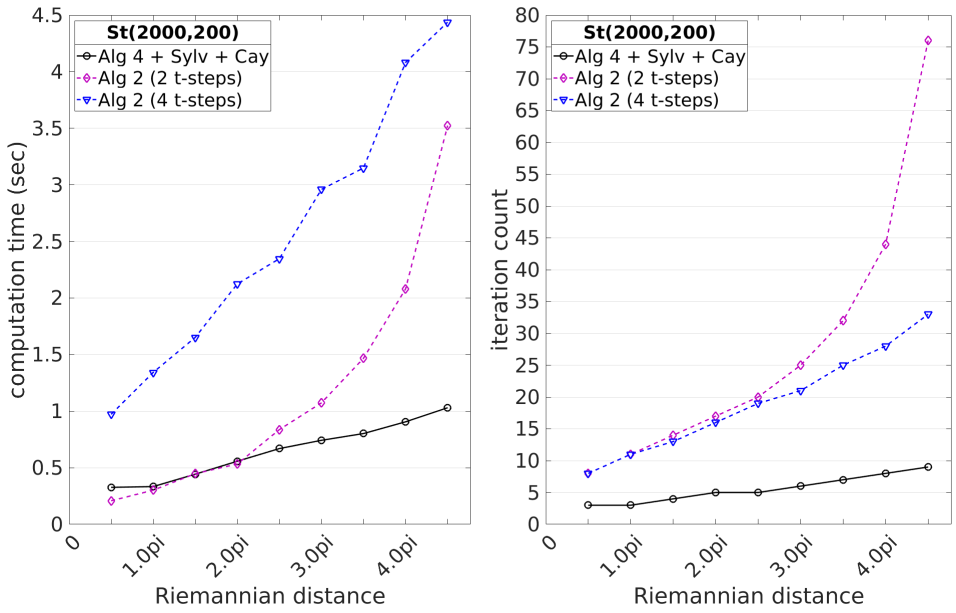

FIG. 3. *(cf. subsection 4.3) Left: Wall clock computation time in seconds versus* $d = \mathrm{dist}(U,\widetilde{U})$. *Right: Iteration count versus* $d = \mathrm{dist}(U,\widetilde{U})$. *The trial parameter set is* $\{d = (0.5 + 0.5j)\cdot\pi |\ j = 0,\ldots,8\}$. *The data points are* $U, \widetilde{U} \in St(2000, 200)$. *Timing results are averaged over* 20 *runs.*

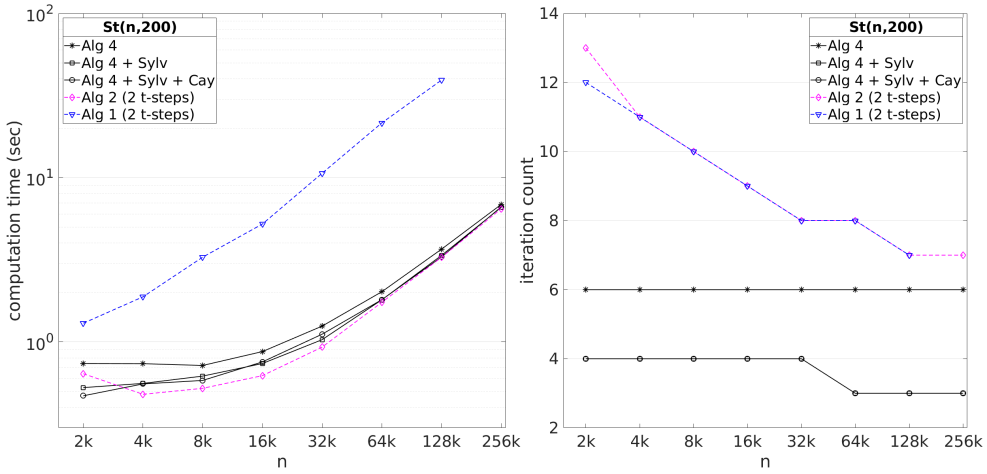

FIG. 4. *(cf. subsection 4.3) Left: Wall clock computation time in seconds versus matrix dimension n on a log-log scale. Right: Iteration count versus matrix dimension n on a log-linear scale. The trial parameter set is $\{n = 1000 \cdot 2^j \mid j = 1, \ldots, 8\}$. The data points $U, \widetilde{U} \in St(n, 200)$ are at a canonical Riemannian distance of $\mathrm{dist}_\alpha(U, \widetilde{U}) = 1.5\pi$. (The case of $St(256k, 200)$ was not treatable with Algorithm 1 on the given laptop computer.)*

TABLE 3
(cf. subsection 4.3) Wallclock computation time for the results displayed in Figure 4.

Method	$n = 8k$	$n = 16k$	$n = 32k$	$n = 64k$	$n = 128k$	$n = 256k$
Alg. 4	0.719s	0.873s	1.25s	2.02s	3.67s	6.87s
Alg. 4+Sylv.	0.620s	0.739s	1.03s	1.80s	3.36s	6.59s
Alg. 4+Sylv.+Cay.	0.582s	0.756s	1.11s	1.81s	3.29s	6.64s
Alg. 2 (2 t-steps)	0.522s	0.623s	0.930s	1.74s	3.27s	6.46s
Alg. 1 (2 t-steps)	3.28s	5.22s	10.63s	21.5s	39.2s	unfeasible

$\mathrm{dist}_0(U, \widetilde{U}) = 1.5\pi$ is constructed as in subsection 4.1. We run the various algorithms with a target accuracy of $\tau = 10^{-10}$ and measure the wall clock computation time as well as the iteration count. Figure 4 displays the results. The associated measurement data for $j = 3, \ldots, 8$ is listed in Table 3. It can be observed that Algorithm 2 and the variants of Algorithm 4 perform comparably, while Algorithm 1 is roughly one order of magnitude slower. This is expected, because the former methods address a matrix problem that scales in the dimension p in the associated loop iterations, but they share matrix multiplications and a QR-decomposition of n-by-p matrices as pre- and postprocessing steps.

We repeat the experiment with fixed $n = 6000$ and $p = 10 \cdot 2^j$ with $j = 1, \ldots, 8$. Figure 5 displays the results. The associated measurement data for $j = 3, \ldots, 8$ is listed in Table 4. It can be observed that for the dimensions tested, Algorithm 2 is fastest until a column-dimension of $p = 320$. Beyond this point, Algorithm 4 with the Sylvester enhancement takes the lead. For $p = 2560$, the wall clock runtime of Algorithm 4 is a factor of 0.82 smaller than the runtime of Algorithm 2.

4.4. The impact of the metric on interpolating matrix factorizations. In this subsection, we consider the practical problems of interpolating the QR-decomposition and the SVD of time-dependent matrix curves. Let $n > m > p$. For a compact QR-decomposition $Y = QR$ of a rectangular matrix $Y \in \mathbb{R}^{n \times p}$, it holds

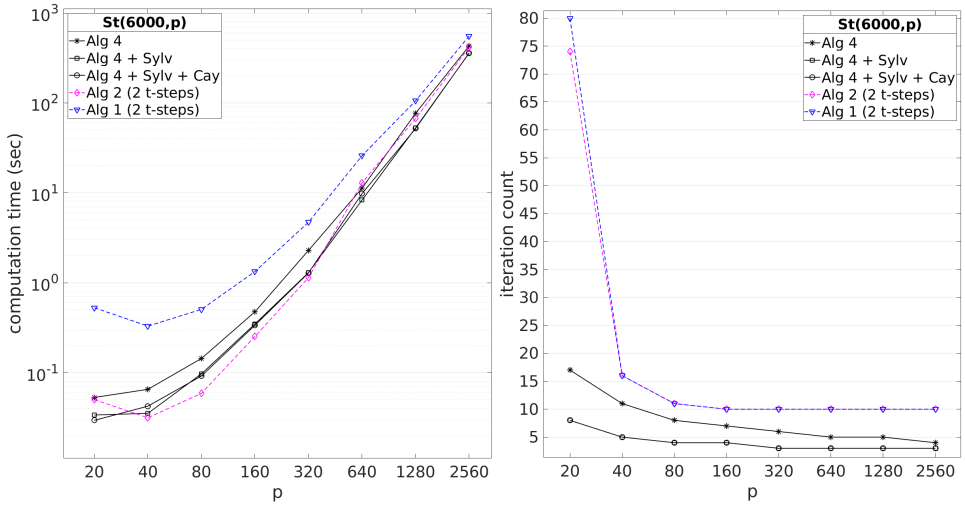

FIG. 5. *(cf. subsection 4.3) Left: Wall clock computation time in seconds versus matrix dimension p on a log-log scale. Right: Iteration count versus matrix dimension p on a log-linear scale. The trial parameter set is* $\{p = 10 \cdot 2^j |\ j = 1, \ldots, 8\}$. *The data points* $U, \widetilde{U} \in St(6000, p)$ *are at a canonical Riemannian distance of* $\mathrm{dist}_\alpha(U, \widetilde{U}) = 1.5\pi$.

TABLE 4
(cf. subsection 4.3) Wall clock computation time for the results displayed in Figure 5.

Method	$p = 80$	$p = 160$	$p = 320$	$p = 640$	$p = 1280$	$p = 2560$
Alg. 4	0.134s	0.474s	2.12s	11.44s	87.31s	423.3s
Alg. 4+Sylv.	0.088s	0.343s	1.27s	7.68s	53.81s	363.0s
Alg. 4+Sylv.+Cay.	0.086s	0.342s	1.28s	7.55s	52.80s	355.1s
Alg. 2 (2 t-steps)	0.076s	0.226s	1.20s	9.87s	61.78s	432.3s
Alg. 1 (2 t-steps)	0.477s	1.336s	4.82s	23.28s	112.6s	564.0s

that $Q \in St(n, p)$. If the SVD is used to produce the best rank-p approximation of a given matrix $\mathbb{R}^{n\times m} \ni Y \approx U_p \Sigma_p V_p^T$, then $U_p \in St(n, p), V_p \in St(m, p)$. Hence, in both applications, interpolation of matrix sample data on the Stiefel manifold has to be considered. We will investigate how the choice of the metric affects the resulting interpolant.

The standard approach to interpolating manifold-valued data is (1) to select a base point, (2) to apply the Riemannian logarithm to map the data set to the tangent space at the chosen base point, (3) to perform interpolation in the tangent space, (4) to map the results back to the manifold via the Riemannian exponential.

First, we reproduce the example from [40, section 5.2] and consider a cubic matrix polynomial

$$t \mapsto Y(t) = Y_0 + tY_1 + t^2 Y_2 + t^3 Y_3, \quad Y_k \in \mathbb{R}^{n\times p}, \quad n = 500, p = 10.$$

The matrices Y_k are produced as random matrices with entries uniformly sampled from $[0, 1]$ for Y_0 and entries uniformly sampled from $[0, 0.5]$ for Y_1, Y_2 and from $[0, 0.2]$ for Y_3. Then, $Y(t)$ is sampled at five equidistant samples $t_i \in \{-1.1, -0.55, 0.0, 0.55, 1.1\}$. At each sample point t_i, the Q-factor $Q(t_i) \in St(n, p)$ of the QR-decomposition is computed. We employ radial basis function (RBF) interpolation in the tangent space with the cubic RBF and three-point cubic spline interpolation; for details, see [40].

For mapping the sample data back and forth between the Stiefel manifold and its tangent space as sketched in Figure 1, we use the Riemannian exponential and logarithm under the α-metric. We start at $\alpha = -0.8$ and proceed until $\alpha = 2.04$ in steps of $\delta\alpha = 0.02$.

For each value of α, the relative interpolation errors are computed at 101 equidistant instants $t_j = -1.1 + j\delta t \in [-1.1, 1.1]$, $\delta t = 0.022$, $j = 0, 1, \ldots, 100$ in the matrix Frobenius norm

$$e_\alpha(t_j) := \frac{\|Q^*_\alpha(t_j) - Q(t_j)\|_F}{\|Q(t_j)\|_F}.$$

Here, $Q^*_\alpha(t_j)$ denotes the manifold interpolant under the α-metric and $Q(t_j)$ is the reference solution. The associated discrete L_2-norm of the error is computed as $\mathrm{err}_{L_2}(\alpha) = \sqrt{\delta t \sum_{j=0}^{100} e_\alpha(t_j)^2}$. The error graphs $\alpha \mapsto \mathrm{err}_{L_2}(\alpha)$ for both interpolation methods under consideration are displayed in Figure 6. Both of the two discrete arrays that underlie the plotted graphs feature a global minimum at a similar location. For RBF interpolation, the total L_2 error is lowest at $\alpha = 0.28$. For three-point cubic spline interpolation, the global minimum is at $\alpha = 0.26$. It can also be seen that beyond $\alpha \geq -0.1$, the impact of the metric on the error is rather negligible. It should also be emphasized that the largest and the smallest tested errors differ only by a small absolute amount.

As a second test case, we rework the example from [40, section 5.3] and construct a nonlinear matrix function with fixed low rank. We start with a cubic matrix polynomial

$$Y(t) = Y_0 + tY_1 + t^2Y_2 + t^3Y_3, \quad Y_i \in \mathbb{R}^{n\times r}, n = 10\,000, r = 10,$$

with random matrices Y_k with entries uniformly sampled from $[0, 1]$ for Y_0 and from $[0, 0.5]$ for Y_1, Y_2, Y_3. Then, a second matrix polynomial is considered:

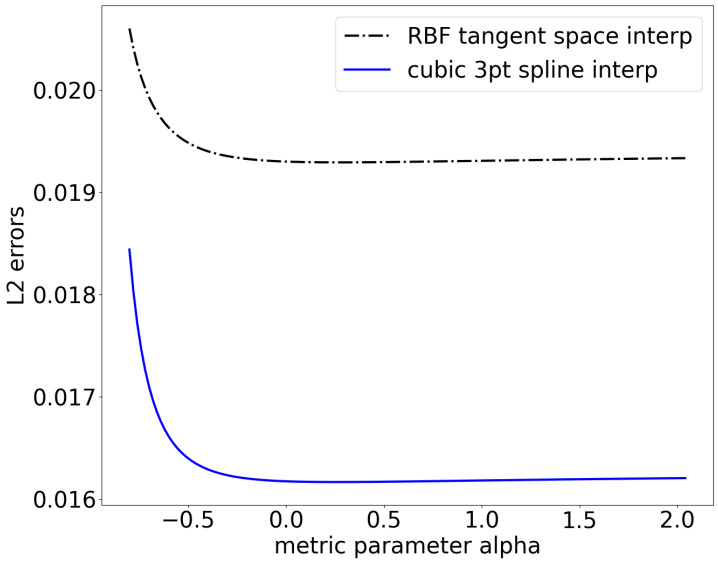

FIG. 6. *(cf. subsection 4.4) Total L_2 errors versus α associated with interpolating the Q-factor of a time-dependent curve of QR-factorizations under varying the α-metric. The tested α-range is $[-0.8, 2.04]$ on steps of size $\delta\alpha = 0.02$.*

$$Z(t) = Z_0 + tZ_1 + t^2 Z_2, \quad Z_i \in \mathbb{R}^{r\times m}, r = 10, m = 300.$$

Here, the entries of Z_0 are sampled uniformly from $[0, 1]$ while the entries of Z_1, Z_2 are sampled uniformly from $[0, 0.5]$. The nonlinear low-rank matrix function is set to be

$$W(t) = Y(t)Z(t) \in \mathbb{R}^{n\times m}.$$

We will conduct cubic Hermite interpolation. To this end, the low-rank SVD

$$W(t) = U_r(t)\Sigma_r(t)V_r(t)^T, \quad U_r(t) \in St(n, r), \;\; V_r(t) \in St(m, r), \;\; \Sigma \in \mathbb{R}^{r\times r},$$

and the associated matrix derivatives are sampled at three Chebychev nodes $t_0 \approx 0.0603, t_1 = 0.45, t_2 = 0.8397$ in the interval $[0.0, 0.9]$. The Hermite sample data set is

$$U_r(t_i), \dot{U}_r(t_i), \quad V_r(t_i), \dot{V}_r(t_i), \quad \Sigma_r(t_i), \dot{\Sigma}_r(t_i), \quad i = 0, 1, 2;$$

see [40, section 5.3] for details.

We conduct Hermite interpolation under the α-metric, starting from $\alpha = -0.75$ and proceeding up to $\alpha = 1.5$ in steps of $\delta\alpha = 0.05$. (For $\alpha < -0.75$ and $\alpha > 1.5$, convergence issues occurred when mapping the sample data set to the tangent space with the Riemannian logarithm.)

For each value of α, the relative interpolation errors are computed at 100 equidistant instants $t_j \in [t_0, t_2]$ in the matrix Frobenius norm as

$$e_\alpha(t_j) = \frac{\|U^*(t_j)\Sigma^*(t_j)(V^*(t_j))^T - W(t_j)\|_F}{\|W(t_j)\|_F},$$

where $U^*(t_j)$, $\Sigma^*(t_j)$, $V^*(t_j)$ are the interpolants of the matrix factors of the low-rank SVD of $W(t_j)$ and $W(t_j)$ is the reference solution. The associated discrete L_2-norm of the error is computed as $\mathrm{err}_{L_2}(\alpha) = \sqrt{\delta t \sum_{j=0}^{99} e_\alpha(t_j)^2}$. The error graph $\alpha \mapsto \mathrm{err}_{L_2}(\alpha)$ is displayed in Figure 7. It can be seen that in the tested α-range, the total L_2 error is largest at the left bound $\alpha = -0.75$ and decreases monotonically with increasing α. The slope flattens considerably for larger values of α.

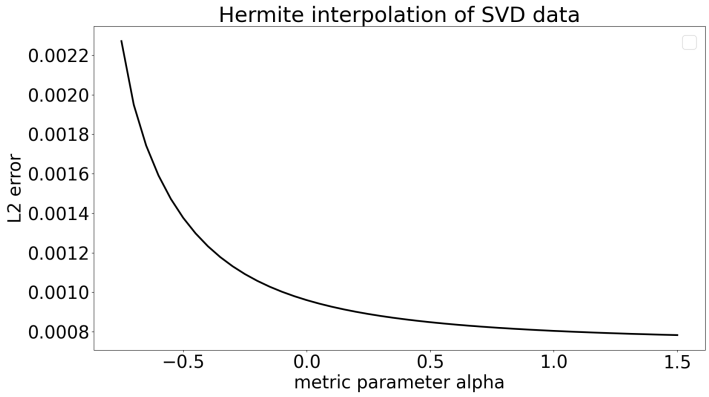

FIG. 7. *(cf. subsection 4.4) Total L_2 errors versus α associated with Hermite-interpolating a time-dependent curve of low-rank SVDs under varying the α-metric. The tested α-range is $[-0.75, 1.5]$ on steps of size $\delta\alpha = 0.05$.*

5. Discussions. For the canonical metric, the numerical experiments identify the Sylvester-enhanced version of Algorithm 4 as the method of choice: The runtime is comparable to Algorithm 2 on two time steps for the smaller test cases and considerably shorter for the test cases in higher dimensions. Moreover, it is more robust with respect to increasing the distance of the input points; see Figure 3.

For all other α-metrics, the p-shooting method Algorithm 2 on two time steps performs best in terms of the runtime. Yet, data on Stiefel manifolds of smaller dimension or data points that are further apart may necessitate in doing more time steps in the inner loops of Algorithm 2. Even though for all test cases considered in this work, it was sufficient to go up to four time steps, Algorithm 2 is a local method and it cannot be expected that increasing the number of inner time steps will make the method converge in all cases. The Euclidean metric ($\alpha = -\frac{1}{2}$) has a special position in the sense that the iteration count of the p-shooting method is lowest for the Stiefel manifold under this choice of metric.

The choice of metric is problem-dependent. We included an experiment on interpolating curves of orthogonal matrix factorizations. When casting this into an interpolation problem on the Stiefel manifold, a metric has to be selected. While the exact curve is completely independent from the selected metric, the interpolation error is not. Hence, if the computational resources allow for it, a parametric study may be beneficial when selecting the metric for a certain application. Otherwise, the canonical metric presents itself as a natural general purpose tool.

Acknowledgment. The authors would like to thank Marco Sutti for sharing the MATLAB code assoicated with [34].

REFERENCES

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *Riemannian geometry of Grassmann manifolds with a view on algorithmic computation*, Acta Appl. Math., 80 (2004), pp. 199–220, https://doi.org/10.1023/B:ACAP.0000013855.14971.91.

[2] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, Princeton, NJ, 2008, http://press.princeton.edu/titles/8586.html.

[3] E. Begelfor and M. Werman, *Affine invariance revisited*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, 2006, pp. 2087–2094, http://doi.ieeecomputersociety.org/10.1109/CVPR.2006.50.

[4] P. Benner, S. Gugercin, and K. Willcox, *A survey of projection-based model reduction methods for parametric dynamical systems*, SIAM Rev., 57 (2015), pp. 483–531, https://doi.org/10.1137/130932715.

[5] A. V. Bernstein and A. P. Kuleshov, *Tangent Bundle Manifold Learning via Grassmann & Stiefel Eigenmaps*, preprint, arXiv:1212.6031, 2012.

[6] R. Bhatia, *Matrix Analysis*, Grad. Texts in Math. 169, Springer-Verlag, New York, 1997.

[7] D. Bryner, *Endpoint geodesics on the Stiefel manifold embedded in Euclidean space*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1139–1159.

[8] E. Celledoni, S. Eidnes, B. Owren, and T. Ringholm, Math. Comput., 89 (2020), pp. 699–716.

[9] R. Chakraborty and B. C. Vemuri, *Statistics on the (Compact) Stiefel Manifold: Theory and Applications*, http://arxiv.org/abs/1708.00045v1, 2017.

[10] M. P. do Carmo, *Riemannian Geometry*, Mathematics: Theory & Applications, Birkhäuser, Boston, 1992, https://books.google.de/books?id=ct91XCWkWEUC.

[11] A. Edelman, T. A. Arias, and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 303–353, https://doi.org/10.1137/S0895479895290954.

[12] K. A. Gallivan, A. Srivastava, X. Liu, and P. Van Dooren, *Efficient algorithms for inferences on Grassmann manifolds*, in Proceedings of the IEEE Workshop on Statistical Signal Processing, 2003, pp. 315–318, https://doi.org/10.1109/SSP.2003.1289408.

[13] R. Godement and U. Ray, *Introduction to the Theory of Lie Groups*, Universitext, Springer, New York, 2017, https://books.google.dk/books?id=ZgnnnAAACAAJ.

[14] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer Ser. Comput. Math. 31, 2nd ed., Springer-Verlag, Berlin, 2006.
[15] N. J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia, 2008.
[16] K. Hüper, M. Kleinsteuber, and F. Silva Leite, *Rolling Stiefel manifolds*, Int. J. Systems Sci., 39 (2008), pp. 881–887, https://doi.org/10.1080/00207720802184717.
[17] K. Hüper, I. Markina, and F. Silva Leite, *A Lagrangian approach to extremal curves on Stiefel manifolds*, J. Geom. Mech., 13 (2021), pp. 55–72.
[18] K. Hüper and F. Ullrich, *Real Stiefel manifolds: An extrinsic point of view*, in Proceedings of the 13th APCA International Conference on Automatic Control and Soft Computing, 2018, pp. 13–18, https://doi.org/10.1109/CONTROLO.2018.8514292.
[19] A. Iserles, H. Z. Munthe-Kaas, S. P. Nørsett, and A. Zanna, *Lie-group methods*, Acta Numer., 9 (2000), 215365, https://doi.org/10.1017/S0962492900002154.
[20] J. M. Lee, *Introduction to Riemannian Manifolds*, Grad. Texts in Math. 176, 2nd ed., Springer, Cham, 2018.
[21] Y. Man Lui, *Advances in matrix manifolds for computer vision*, Image and Vision Computing, 30 (2012), pp. 380–388, http://dx.doi.org/10.1016/j.imavis.2011.08.002.
[22] H. Q. Minh and V. Murino, *Algorithmic Advances in Riemannian Geometry and Applications: For Machine Learning, Computer Vision, Statistics, and Optimization*, Adv. Comput. Vis. Pattern Recognit., Springer, Cham, 2016.
[23] M. Müger, *Notes on the Theorem of Baker-Campbell-Hausdorff-Dynkin*, Lecture Notes, Radboud University, 2020, https://www.math.ru.nl/ mueger/PDF/BCHD.pdf.
[24] M. Newman, S. Wasin, and R. C. Thompson, *Convergence domains for the Campbell-Baker-Hausdorff formula*, Linear Multilinear Algebra, 24 (1989), pp. 301–310.
[25] L. Noakes, *A global algorithm for geodesics*, J. Austral. Math. Soc. Ser. A, 65 (1998), pp. 37–50, https://doi.org/10.1017/S1446788700039380.
[26] V. Patrangenaru and L. Ellingson, *Nonparametric Statistics on Manifolds and Their Applications to Object Data Analysis*, Chapman & Hall/CRC Monogra. Statistics Applied Probability, CRC Press, Boca Raton, FL, 2015, https://books.google.dk/books?id=NtOYCgAAQBAJ.
[27] I. U. Rahman, I. Drori, V. C. Stodden, D. L. Donoho, and P. Schröder, *Multiscale representations for manifold-valued data*, SIAM J. Multiscale Model. Simul., 4 (2005), pp. 1201–1232.
[28] Q. Rentmeesters, *Algorithms for Data Fitting on Some Common Homogeneous Spaces*, Ph.D. thesis, Université Catholique de Louvain, Louvain, Belgium, 2013, http://hdl.handle.net/2078.1/132587.
[29] W. Rossmann, *Lie Groups: An Introduction Through Linear Groups*, Ox. Grad. Texts Math. 5, Oxford University Press, New York, 2006, https://books.google.de/books?id=bAjulQ65W-UC.
[30] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[31] A. Srivastava and E. P. Klassen, *Functional and Shape Data Analysis*, Springer Ser. Statist., Springer-Verlag, New York, 2016, https://doi.org/10.1007/978-1-4939-4020-2.
[32] A. Srivastava and P. K. Turaga, *Riemannian Computing in Computer Vision*, Springer, New York, 2015, https://doi.org/10.1007/978-3-319-22957-7.
[33] G. Sundaramoorthi, A. Mennucci, S. Soatto, and A. Yezzi, *A new geometric metric in the space of curves, and applications to tracking deforming objects by prediction and filtering*, SIAM J. Imaging Sci., 4 (2011), pp. 109–145, https://doi.org/10.1137/090781139.
[34] M. Sutti, *Riemannian Algorithms on the Stiefel and the Fixed-Rank Manifold*, Ph.D. thesis, Université de Genève, 2020, https://archive-ouverte.unige.ch/unige:146438.
[35] M. Sutti and B. Vandereycken, *The Leapfrog Algorithm as Nonlinear Gauss–Seidel*. arXiv:2010.14137v1, 2020.
[36] R. C. Thompson, *Convergence proof for Goldberg's exponential series*, Linear Algebra Appl., 121 (1989), pp. 3–7.
[37] P. K. Turaga, A. Veeraraghavan, and R. Chellappa, *Statistical analysis on Stiefel and Grassmann manifolds with applications in computer vision*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8, https://doi.org/10.1109/CVPR.2008.4587733.
[38] A. Van-Brunt and M. Visser, *Simplifying the Reinsch Algorithm for the Baker-Campbell-Hausdorff Series*, http://arxiv.org/abs/1501.05034, 2015.
[39] R. Zimmermann, *A matrix-algebraic algorithm for the Riemannian logarithm on the Stiefel manifold under the canonical metric*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 322–342, https://doi.org/10.1137/16M1074485.

[40] R. Zimmermann, *Hermite interpolation and data processing errors on Riemannian matrix manifolds*, SIAM J. Sci. Comput., 42 (2020), pp. A2593–A2619, https://doi.org/10.1137/19M1282878.

[41] R. Zimmermann, *A note on rank-one subspace modifications and some remarks on the canonical Stiefel logarithm*, Oberwolfach Rep., 20 (2018), pp. 57–61, https://doi.org/DOI: 10.4171/OWR/2018/20.

[42] R. Zimmermann, *Manifold interpolation*, in System- and Data-Driven Methods and Algorithms, P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. M. Silveira, eds., Model Order Reduction 1, De Gruyter, Boston, 2021, pp. 229–274.