Simplifying Momentum-based Positive-definite Submanifold Optimization with Applications to Deep Learning

Wu Lin [1] **Valentin Duruisseaux** [2] **Melvin Leok** [2] **Frank Nielsen** [3] **Mohammad Emtiyaz Khan** [4] **Mark Schmidt** [1 5]

Abstract

Riemannian submanifold optimization with momentum is computationally challenging because, to ensure that the iterates remain on the submanifold, we often need to solve difficult differential equations. Here, we simplify such difficulties for a class of structured symmetric positive-definite matrices with affine-invariant metric. We do so by proposing a generalized version of the Riemannian normal coordinates that dynamically orthonormalizes the metric and locally converts the problem into an unconstrained problem in the Euclidean space. We use our approach to simplify existing approaches for structured covariances and develop matrix-inverse-free 2^{nd}-order optimizers for deep learning with low numerical precision.

1. Introduction

Estimation of symmetric positive definite (SPD) matrices is important in machine learning and related fields. For example, many optimization methods require it to estimate preconditioning matrices. Approximate inference methods also estimate SPD matrices to obtain Gaussian posterior approximations. Other applications include dictionary learning (Cherian & Sra, 2016), trace regression (Slawski et al., 2015) for kernel matrices, metric learning (Guillaumin et al., 2009), log-det maximization (Wang et al., 2010), Gaussian mixtures (Hosseini & Sra, 2015), and Gaussian graphical models (Makam et al., 2021).

Because the set of SPD matrices forms a Riemannian manifold, one can use Riemannian gradient descent (RGD) for SPD estimation, but this can be computationally infeasible in high-dimensions. This is because RGD often requires inversion of dense matrices (see Table 1). Computation can be reduced by using sparse matrices induced by a submanifold. However, this complicates manifold operations needed for such submanifold optimization. For example, the Riemannian gradient computation may involve another matrix inverse (see Table 1). Other operations needed for Riemannian momentum, such as the Riemannian exponential map and the parallel transport map, also require solving an intractable ordinary differential equation (ODE).

Another idea to develop practical Riemannian methods is to use moving coordinates where a local coordinate is generated, used, and discarded at each iteration. Such approaches can efficiently handle manifold constraints, for example, the recently proposed natural-gradient descent (NGD) method by Lin et al. (2021a). However, it is nontrivial to include metric-aware momentum using moving coordinates in a computationally efficient way. In this paper, we aim to simplify the addition of momentum to such methods and develop efficient momentum-based updates on submanifolds.

We propose special local coordinates for a class of SPD submanifolds with the affine-invariant metric. Our approach avoids the use of global coordinates as well as the computation of Riemannian exponential and transport maps. Instead, we exploit Lie-group structures to obtain efficient structure-preserving updates on submanifolds. Under our local coordinates, all constraints disappear and the metric at evaluation points becomes the standard Euclidean metric. This *metric-preserving* trivialization on a submanifold enables an efficient metric-inverse-free Riemannian momentum update by essentially performing, in the local coordinates, a momentum-based Euclidean gradient descent update.

We extend the structured NGD approach in several ways: (i) we establish its connection to Riemannian methods (Sec. 3.1); (ii) we demystify the construction of coordinates (Sec. 3.2); (iii) we introduce new coordinates for efficient momentum computation (Sec. 3.1); and (iv) we expand its scope to structured SPD matrices where Gaussian and Bayesian assumptions are not needed (Sec. 3.3). By exploiting the submanifold structure of preconditioners, we use our method to develop new inverse-free structured optimizers for deep learning (DL) in low precision settings (Sec. 4).

[1]University of British Columbia, Vancouver, Canada [2]University of California San Diego, San Diego, California, USA [3]Sony Computer Science Laboratories Inc., Tokyo, Japan [4]RIKEN Center for Advanced Intelligence Project, Tokyo, Japan [5]CIFAR AI Chair, Alberta Machine Intelligence Institute, Alberta, Canada. Correspondence to: Wu Lin <yorker.lin@gmail.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

2. Manifold Optimization and its Challenges

Consider a complete manifold $\mathcal{M}$ with a Riemannian metric $\mathbf{F}$ represented by a single global coordinate $\boldsymbol{\tau}$. Under this coordinate system, Riemannian gradient descent (Absil et al., 2009; Bonnabel, 2013) (RGD) is defined as

$$\text{RGD}: \boldsymbol{\tau} \leftarrow \text{RExp}(\boldsymbol{\tau}, -\beta \mathbf{F}^{-1}(\boldsymbol{\tau})\mathbf{g}(\boldsymbol{\tau})), \qquad (1)$$

where $\mathbf{v}(\boldsymbol{\tau}) := \mathbf{F}^{-1}(\boldsymbol{\tau})\mathbf{g}(\boldsymbol{\tau})$ is a Riemannian gradient, $\mathbf{g}(\boldsymbol{\tau})$ is a Euclidean gradient, $\mathbf{F}$ is the metric, β is a stepsize, and $\text{RExp}(\boldsymbol{\tau}, \mathbf{v})$ is the Riemannian exponential map defined by solving a nonlinear (geodesic) ODE (see Appx. C.3). The nonlinearity of the ODE makes it difficult to obtain a closed-form expression for the solution.

To incorporate momentum in RGD, a Riemannian parallel transport map $\hat{T}_{\boldsymbol{\tau}^{(\text{cur})} \to \boldsymbol{\tau}^{(\text{new})}}(\boldsymbol{\nu}^{(\text{cur})})$ is introduced in many works to move the Riemannian vector $\boldsymbol{\nu}^{(\text{cur})}$ computed at point $\boldsymbol{\tau}^{(\text{cur})}$ to point $\boldsymbol{\tau}^{(\text{new})}$ on the manifold. For example, Alimisis et al. (2020) propose the following update using the Riemannian transport map (see Appx. C.4) and Riemannian momentum $\boldsymbol{\nu}^{(\text{cur})}$ with momentum weight α.

$$\begin{aligned} \text{Momentum}: \boldsymbol{\nu}^{(\text{cur})} &\leftarrow \alpha \mathbf{z}^{(\text{cur})} + \beta \mathbf{F}^{-1}(\boldsymbol{\tau}^{(\text{cur})})\mathbf{g}(\boldsymbol{\tau}^{(\text{cur})}), \\ \text{RGD}: \boldsymbol{\tau}^{(\text{new})} &\leftarrow \text{RExp}(\boldsymbol{\tau}^{(\text{cur})}, -\boldsymbol{\nu}^{(\text{cur})}), \\ \text{Transport}: \mathbf{z}^{(\text{new})} &\leftarrow \hat{T}_{\boldsymbol{\tau}^{(\text{cur})} \to \boldsymbol{\tau}^{(\text{new})}}(\boldsymbol{\nu}^{(\text{cur})}). \end{aligned} \qquad (2)$$

Unlike existing works, we suggest using Euclidean momentum and a Euclidean parallel transport map (see Appx. C.5) $T_{\boldsymbol{\tau}^{(\text{cur})} \to \boldsymbol{\tau}^{(\text{new})}}(\mathbf{m}^{(\text{cur})})$ to move the momentum. The use of this Euclidean map is essential for efficient approximations of the transport, as will be discussed in Sec. 3.1. Through this Euclidean map, we obtain an equivalent update of Eq. (2) (shown in Appx. C.6) via Euclidean momentum $\mathbf{m}^{(\text{cur})}$:

$$\begin{aligned} \text{Momentum}: \mathbf{m}^{(\text{cur})} &\leftarrow \alpha \mathbf{w}^{(\text{cur})} + \beta \mathbf{g}(\boldsymbol{\tau}^{(\text{cur})}), \\ \text{RGD}: \boldsymbol{\tau}^{(\text{new})} &\leftarrow \text{RExp}(\boldsymbol{\tau}^{(\text{cur})}, -\mathbf{F}^{-1}(\boldsymbol{\tau}^{(\text{cur})})\mathbf{m}^{(\text{cur})}), \\ \text{Transport}: \mathbf{w}^{(\text{new})} &\leftarrow T_{\boldsymbol{\tau}^{(\text{cur})} \to \boldsymbol{\tau}^{(\text{new})}}(\mathbf{m}^{(\text{cur})}). \end{aligned} \qquad (3)$$

Both the transport maps are defined by solving transport ODEs (defined in Appx. C.4-C.5). However, solving any of the ODEs can be computationally intensive since it is a linear system of differential equations and the solution often involves matrix decomposition.

2.1. Challenges on SPD Manifold and Submanifolds

Consider the k-by-k SPD manifold $\mathcal{M} = \{\boldsymbol{\tau} \in \mathbb{R}^{k \times k} | \boldsymbol{\tau} \succ 0\}$. The affine-invariant metric $\mathbf{F}$ for the SPD manifold (see Theorem 2.10 of Minh & Murino (2017)) is defined as twice the Fisher-Rao metric (see Appx. C.1) for the k-dimensional Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\tau})$ with zero mean and covariance $\boldsymbol{\tau}$, which is the 2nd derivative of a matrix,

$$\mathbf{F}(\boldsymbol{\tau}) = -2\mathbb{E}_{\mathcal{N}(\mathbf{0},\boldsymbol{\tau})}\left[\nabla^2_{\boldsymbol{\tau}} \log \mathcal{N}(\mathbf{0}, \boldsymbol{\tau})\right]. \qquad (4)$$

The Fisher-Rao metric known as the Fisher information matrix is a well-known and useful metric for many machine learning applications. Moreover, the affine-invariant metric (Pennec et al., 2006) is more suitable and useful for SPD matrices compared to the Euclidean metric (Hosseini & Sra, 2015). Thus, we use and preserve the affine-invariant metric.

In the SPD manifold case, the Riemannian maps have a closed-form expression (see Table 1). However, the updates in Eq. (1)-(3) require computing full-rank matrix inverses in the Riemannian maps, so such methods are impractical in high-dimensional cases. We will propose an alternative approach to construct practical Riemannian methods for SPD submanifolds in higher-dimensions.

For many SPD submanifolds with the same (induced) metric, it is nontrivial to implement updates in Eq. (1)- (3) since these needed Riemannian maps often do not admit a simple closed-form expression. For example, consider the following SPD submanifold, which can be used (Calvo & Oller, 1990) to represent a $(k-1)$-dimensional Gaussian with mean $\boldsymbol{\mu}$ and full covariance $\boldsymbol{\Sigma} := \mathbf{V} - \boldsymbol{\mu}\boldsymbol{\mu}^T$,

$$\mathcal{M} = \left\{\boldsymbol{\tau} = \begin{bmatrix} \mathbf{V} & \boldsymbol{\mu} \\ \boldsymbol{\mu}^T & 1 \end{bmatrix} \in \mathbb{R}^{k \times k} \mid \boldsymbol{\tau} \succ 0\right\}.$$

The Riemannian exponential map for this submanifold does not have a simple and closed-form expression (Calvo & Oller, 1991), not to mention other submanifolds induced by structured covariance $\boldsymbol{\Sigma}$. The exponential map also is unknown on the following rank-one SPD submanifold,

$$\mathcal{M} = \left\{\boldsymbol{\tau} = \begin{bmatrix} a^2 & a\mathbf{b}^T \\ a\mathbf{b} & \mathbf{b}\mathbf{b}^T + \text{Diag}(\mathbf{c}^2) \end{bmatrix} \in \mathbb{R}^{k \times k} \mid \boldsymbol{\tau} \succ 0\right\}.$$

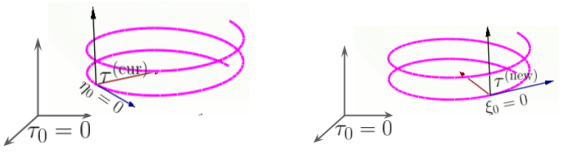
The existing Riemannian maps defined on the full SPD manifold such as the exponential map and the transport maps cannot be used on SPD submanifolds since these maps do not persevere the submanifold structures, and in particular, do not guarantee that their output stays on a given SPD submanifold. To stay on a submanifold, a retraction map using global coordinate $\boldsymbol{\tau}$ is proposed as an approximation of the exponential map for the submanifold. Likewise, a vector transport map is proposed to approximate the Riemannian parallel transport map. However, both retraction and vector transport maps vary from one submanifold to another. It can be difficult to design such maps for a new submanifold. A generic approach to design such maps is to approximate the ODEs. However, it is computationally challenging to even evaluate the ODEs at a point when the global coordinate $\boldsymbol{\tau}$ is used, since this requires computing the Christoffel symbols $\Gamma^a_{cb}(\boldsymbol{\tau})$ (defined in Appx. C.2) arising in the ODEs. These symbols are defined by partial derivatives of the metric $\mathbf{F}(\boldsymbol{\tau})$ in Eq. (4), so their computation involves complicated 3rd order derivatives of a matrix. Furthermore, it is unclear how to efficiently compute a Riemannian gradient $\mathbf{F}^{-1}(\boldsymbol{\tau})\mathbf{g}(\boldsymbol{\tau})$ on a SPD submanifold without explicitly inverting the metric, which can be another computationally intensive operation.

Operations on SPD in global $\boldsymbol{\tau}$	Manifold	Submanifolds
Riemann. gradient $\mathbf{v}$ at $\boldsymbol{\tau}_0$	$\boldsymbol{\tau}_0\mathbf{g}\boldsymbol{\tau}_0$	$\mathbf{F}^{-1}(\boldsymbol{\tau}_0)\mathbf{g}$
Riemann. exponential $\mathrm{RExp}(\boldsymbol{\tau}_0, \mathbf{v})$	$\boldsymbol{\tau}_0^{1/2}\mathrm{Expm}(\boldsymbol{\tau}_0^{-1/2}\mathbf{v}\boldsymbol{\tau}_0^{-1/2})\boldsymbol{\tau}_0^{1/2}$	Unknown
Riemann. transport $\hat{T}_{\boldsymbol{\tau}_0\to\boldsymbol{\tau}_1}(\mathbf{v})$	$\mathbf{E}\mathbf{v}\mathbf{E}^T;\ \mathbf{E} := (\boldsymbol{\tau}_1\boldsymbol{\tau}_0^{-1})^{1/2}$	Unknown
Euclidean transport $T_{\boldsymbol{\tau}_0\to\boldsymbol{\tau}_1}(\mathbf{g})$	$\mathbf{H}\mathbf{g}\mathbf{H}^T;\ \mathbf{H} := \boldsymbol{\tau}_1^{-1}\mathbf{E}\boldsymbol{\tau}_0$	Unknown

Table 1. Manifold operations compatible with affine-invariant metric $\mathbf{F}$, where $\mathrm{Expm}(\mathbf{N}) = \sum_{k=0}^{\infty}\frac{1}{k!}\mathbf{N}^k$ is the matrix exponential function, and $\mathbf{g}$ is a (symmetric) Euclidean gradient at $\boldsymbol{\tau}_0$. On submanifolds, $\boldsymbol{\tau}$ denotes learnable parameters.

	SNC	GNC
$\boldsymbol{\tau}$	$\boldsymbol{\psi}_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta})$	$\boldsymbol{\phi}_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta})$
$\boldsymbol{\tau}^{(\mathrm{cur})}$	$\boldsymbol{\psi}_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta}_0)$	$\boldsymbol{\phi}_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta}_0)$
$\mathbf{F}(\boldsymbol{\eta}_0)$	$\mathbf{I}$	$\mathbf{I}$
$\Gamma^a_{bc}(\boldsymbol{\eta}_0)$	0	can be nonzero

Table 2. Properties of Riemannian normal coordinates $\boldsymbol{\eta}$ defined at $\boldsymbol{\tau}^{(\mathrm{cur})}$, where the original $\boldsymbol{\eta}_0 = \mathbf{0}$ represents $\boldsymbol{\tau}^{(\mathrm{cur})}$, $\boldsymbol{\psi}$ is defined by the Riemannian exponential map, and $\boldsymbol{\phi}$ is a diffeomorphic and isometric map.



(a) Normal coordinate $\boldsymbol{\eta}$ at $\boldsymbol{\tau}^{(\mathrm{cur})}$ (b) Normal coordinate $\boldsymbol{\xi}$ at $\boldsymbol{\tau}^{(\mathrm{new})}$

Figure 1. A (orthonormal) SNC/GNC is generated at each iteration

2.2. Natural-gradient Descent and its Challenges

A practical approach is natural-gradient descent (NGD), which approximates the Riemannian exponential map by ignoring the Christoffel symbols. A NGD update is a linear approximation of the update of Eq. (1) given by

$$\boldsymbol{\tau} \leftarrow \boldsymbol{\tau} - \beta\mathbf{F}^{-1}(\boldsymbol{\tau})\mathbf{g}(\boldsymbol{\tau}). \tag{5}$$

This approximation is also known as the Euclidean retraction map (Jeuris et al., 2012). Unfortunately, NGD in the global coordinate $\boldsymbol{\tau}$ does not guarantee that the update stays on a manifold even in the full SPD case. Moreover, computing Riemannian gradients remains challenging due to the metric inverse. Structured NGD (Lin et al., 2021a) addresses the SPD constraint of Gaussians for Bayesian posterior approximations by performing NGD on local coordinates. Local coordinates could enable efficient Riemannian gradient computation by simplifying the metric inverse computation. However, it is nontrivial to incorporate momentum in structured NGD due to the metric and the use of moving coordinates. We address this issue and develop practical Riemannian momentum methods by using generalized normal coordinates. Using our normal coordinates, we will explain and generalize structured NGD from a manifold optimization perceptive. We further expand the scope of structured NGD to SPD submanifolds by going beyond the Bayesian settings. Our local-parameter approach gives a computationally efficient paradigm for a class of SPD submanifolds where it can be nontrivial to design an efficient retraction map using a global coordinate while keeping the Riemannian gradient computation efficient and inverse-free.

2.3. Standard Normal Coordinates (SNCs)

Defn 1: A metric $\mathbf{F}$ is orthonormal at $\boldsymbol{\eta}_0 = \mathbf{0}$ if $\mathbf{F}(\boldsymbol{\eta}_0) = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

Normal coordinates can simplify calculations in differential geometry and general relativity. However, these coordinates are seldom studied in the optimization literature. Given a reference point $\boldsymbol{\tau}^{(\mathrm{cur})}$, the standard (Riemannian) normal coordinate (SNC) $\boldsymbol{\eta}$ at $\boldsymbol{\tau}^{(\mathrm{cur})}$ is defined as below, where $\boldsymbol{\tau}^{(\mathrm{cur})}$ and $\mathbf{F}^{-1/2}(\boldsymbol{\tau}^{(\mathrm{cur})})$ are treated as constants in coordinate $\boldsymbol{\eta}$:

$$\boldsymbol{\tau} = \boldsymbol{\psi}_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta}) := \mathrm{RExp}(\boldsymbol{\tau}^{(\mathrm{cur})}, \mathbf{F}^{-1/2}(\boldsymbol{\tau}^{(\mathrm{cur})})\boldsymbol{\eta}). \tag{6}$$

$\mathbf{F}^{-1/2}(\boldsymbol{\tau}^{(\mathrm{cur})})$ in Eq. (6) is essential since it orthonormalizes the metric at $\boldsymbol{\eta}_0$ (i.e., $\mathbf{F}(\boldsymbol{\eta}_0) = \mathbf{I}$) and will simplify the Riemannian gradient computation in Eq. (7) and (11).

Lezcano (2019; 2020) consider (non-normal) local coordinates defined by the Riemannian exponential such as $\boldsymbol{\tau} = \mathrm{RExp}(\boldsymbol{\tau}^{(\mathrm{cur})}, \boldsymbol{\lambda})$. However, the metric is not orthonormal in their coordinates (i.e., $\mathbf{F}(\boldsymbol{\lambda}_0) \equiv \mathbf{F}(\boldsymbol{\tau}^{(\mathrm{cur})}) \neq \mathbf{I}$ at $\boldsymbol{\lambda}_0 = \mathbf{0}$). Lezcano (2019; 2020) use an ad-hoc metric $\mathbf{I}$ instead of $\mathbf{F}(\boldsymbol{\lambda}_0)$ at $\boldsymbol{\lambda}_0$. Thus, their approach does not perverse the predefined metric $\mathbf{F}$. Another issue is that the ad-hoc metric in their approach does not obey the metric transform rule needed for the change of local coordinates at each iteration. In Sec. 3, we will fix these two issues via *metric-aware* orthonormalizations and propose *metric-preserving* trivializations even when the Riemannian exponential is unknown.

A SNC has nice computational properties, summarized in Table 2. In coordinate $\boldsymbol{\eta}$, the origin $\boldsymbol{\eta}_0 := \mathbf{0}$ represents the point $\boldsymbol{\tau}^{(\mathrm{cur})} = \boldsymbol{\psi}_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta}_0)$ as illustrated in Fig. 1.

RGD in Eq. (1) can be reexpressed as (Euclidean) gradient descent (GD) in local coordinate $\boldsymbol{\eta}$ since the metric $\mathbf{F}(\boldsymbol{\eta}_0)$ at $\boldsymbol{\eta}_0$ becomes the standard Euclidean metric $\mathbf{I}$:

$$\begin{aligned}\mathrm{GD}: \boldsymbol{\eta}_1 &\leftarrow \boldsymbol{\eta}_0 - \beta\mathbf{F}^{-1}(\boldsymbol{\eta}_0)\mathbf{g}(\boldsymbol{\eta}_0) = \mathbf{0} - \beta\mathbf{I}^{-1}\mathbf{g}(\boldsymbol{\eta}_0),\\ \boldsymbol{\tau}^{(\mathrm{new})} &\leftarrow \boldsymbol{\psi}_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta}_1),\end{aligned} \tag{7}$$

where $\mathbf{g}(\boldsymbol{\eta}_0) = \mathbf{F}^{-1/2}(\boldsymbol{\tau}^{(\mathrm{cur})})\mathbf{g}(\boldsymbol{\tau}^{(\mathrm{cur})})$ is a Euclidean gradient evaluated at $\boldsymbol{\eta}_0$. Since the metric $\mathbf{F}(\boldsymbol{\eta}_0)$ is orthonormal, the GD update is also a NGD update in coordinate $\boldsymbol{\eta}$. The orthonormalization of the metric makes it easy to add momentum into RGD while preserving the metric.

In the SPD case, $\mathbf{F}^{-1/2}(\boldsymbol{\tau})\boldsymbol{\eta} = \boldsymbol{\tau}^{1/2}\boldsymbol{\eta}\boldsymbol{\tau}^{1/2}$, where $\boldsymbol{\eta}$ is a symmetric matrix. Recall that $\mathbf{F}(\boldsymbol{\tau}^{(\mathrm{cur})}) \neq \mathbf{I}$ for any $\boldsymbol{\tau}^{(\mathrm{cur})} \neq \mathbf{I}$. However, by Eq. (4), the metric $\mathbf{F}(\boldsymbol{\eta})$ is orthonormal, at $\boldsymbol{\eta}_0$ associated to $\boldsymbol{\tau}^{(\mathrm{cur})} = \boldsymbol{\psi}_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta}_0)$ as shown below.

$$\mathbf{F}(\boldsymbol{\eta}_0) = -2\mathbb{E}_{\mathcal{N}(\mathbf{0},\boldsymbol{\tau}))}\left[\nabla^2_{\boldsymbol{\eta}}\log\mathcal{N}(\mathbf{0},\boldsymbol{\tau})\right]\big|_{\boldsymbol{\eta}=\boldsymbol{\eta}_0} = \mathbf{I}, \tag{8}$$

where $\boldsymbol{\tau} = \boldsymbol{\psi}_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta})$. By Table 1, $\boldsymbol{\psi}_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta})$ in SNC $\boldsymbol{\eta}$ has a closed-form, where $\mathrm{Expm}(\cdot)$ is the matrix exponential,

$$\boldsymbol{\psi}_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta}) = \left(\boldsymbol{\tau}^{(\mathrm{cur})}\right)^{1/2}\mathrm{Expm}(\boldsymbol{\eta})\left(\boldsymbol{\tau}^{(\mathrm{cur})}\right)^{1/2}. \tag{9}$$

Eq. (9) is only obtainable for SPD manifolds if we use SNC $\boldsymbol{\eta}$, and in particular, the metric must be orthonormal at $\boldsymbol{\eta}_0$.

In the case of SPD submanifolds, it is hard to use a SNC as it relies on the intractable Riemannian exponential map, as seen in Eq. (6). However, we will make use of Eq. (9) to generalize normal coordinates and to reduce the computation cost of Riemannian gradients on SPD submanifolds.

3. Generalized Normal Coordinates (GNCs)

We propose new (metric-aware orthonormal) coordinates to simplify the momentum computation on a (sub)manifold with a predefined metric. Inspired by SNCs, we identify their properties that enable efficient Riemannian optimization, and generalize normal coordinates by defining new coordinates satisfying these properties on the (sub)manifold.

Defn 2: A local coordinate is a generalized (Riemannian) normal coordinate (GNC) at point $\boldsymbol{\tau}^{(\text{cur})}$ denoted by $\boldsymbol{\tau} = \boldsymbol{\phi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta})$ if $\boldsymbol{\phi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta})$ satisfies all the following assumptions.

Assumption 1: *The origin $\boldsymbol{\eta}_0 = \mathbf{0}$ represents point $\boldsymbol{\tau}^{(cur)} = \boldsymbol{\phi}_{\boldsymbol{\tau}^{(cur)}}(\boldsymbol{\eta}_0)$ in coordinate $\boldsymbol{\eta}$.*

Assumption 2: *The metric $\mathbf{F}(\boldsymbol{\eta}_0)$ is orthonormal at $\boldsymbol{\eta}_0$.*

Assumption 3: *The map $\boldsymbol{\phi}_{\boldsymbol{\tau}^{(cur)}}(\boldsymbol{\eta})$ is bijective. $\boldsymbol{\phi}_{\boldsymbol{\tau}^{(cur)}}$ and $\boldsymbol{\phi}^{-1}_{\boldsymbol{\tau}^{(cur)}}$ are smooth (i.e. $\boldsymbol{\phi}_{\boldsymbol{\tau}^{(cur)}}(\boldsymbol{\eta})$ is diffeomorphic).*

Assumption 4: *The parameter space of $\boldsymbol{\eta}$ is a vector space.*

Assumption 1 enables simplification using the chain rule of high-order derivative calculations (i.e., the metric and Christoffel symbols in Appx. E, F) by evaluating at zero, which is useful when computing Christoffel symbols. Assumption 2 enables metric preservation in GD updates by dynamically orthonormalizing the metric. By Assumption 3, (sub)manifold constraints disappear in these coordinates, and Assumption 4 ensures that scalar products and vector additions are well-defined, so that we can perform GD. For example, SNCs satisfy Assumption 1-2 (see Table 2) and Assumption 3 due to the Riemannian exponential. On a complete manifold, SNCs satisfy Assumption 4 (Absil et al., 2009). These assumptions make it easy to design new normal coordinates without computing the Riemannian exponential. Thus, both the metric and the (sub)manifold constraints are trivialized in our coordinates. Assumptions 1-4 together imply that $\boldsymbol{\phi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta})$ is a metric-preserving/isometric map from the tangent space at $\boldsymbol{\tau}^{(\text{cur})}$ with Riemannian metric $\mathbf{F}(\boldsymbol{\tau}^{(\text{cur})})$ to a (local) coordinate space of $\boldsymbol{\eta}$ identified as a tangent space at $\boldsymbol{\eta}_0$ with the Euclidean metric $\mathbf{F}(\boldsymbol{\eta}_0)=\mathbf{I}$ such as a matrix subspace in Sec. 3.3. Our approach differs from Lezcano (2019; 2020); Lin et al. (2021a;b), as the metric is not orthonormal in their coordinates.

We will propose GNCs to work with NGD/GD by satisfying Assumptions 1-4 while reducing the computational cost by ignoring Christoffel symbols (see Table 2). A GD update in a GNC is an approximation of the RGD update in Eq. (7). As will be shown in Sec. 3.3.1, the GD resembles structured NGD in Gaussian cases, both of which ignore the symbols,

$$\begin{aligned}\text{GD}: \boldsymbol{\eta}_1 &\leftarrow \boldsymbol{\eta}_0 - \beta\mathbf{F}^{-1}(\boldsymbol{\eta}_0)\mathbf{g}(\boldsymbol{\eta}_0) = \mathbf{0} - \beta\mathbf{I}^{-1}\mathbf{g}(\boldsymbol{\eta}_0),\\ \boldsymbol{\tau}^{(\text{new})} &\leftarrow \boldsymbol{\phi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta}_1).\end{aligned} \tag{10}$$

In the full SPD case, Lin et al. (2021a) establish a connection between the update in $\boldsymbol{\eta}$ and a retraction map in $\boldsymbol{\tau}$.

3.1. Adding Momentum using GNCs

Since the metric is orthonormal at $\boldsymbol{\eta}_0$, we propose simply adding (Euclidean) momentum in the GD update in our normal coordinates. At each iteration, given the current point $\boldsymbol{\tau}^{(\text{cur})}$ in the global coordinate, we generate a GNC $\boldsymbol{\eta}$ at $\boldsymbol{\tau}^{(\text{cur})}$ and perform the update in coordinate $\boldsymbol{\eta}$, where we first assume momentum $\mathbf{w}^{(\eta_0)}$ is given.

our update with momentum

$$\begin{aligned}\text{Momentum}: \mathbf{m}^{(\eta_0)} &\leftarrow \alpha\mathbf{w}^{(\eta_0)} + \beta\mathbf{g}(\boldsymbol{\eta}_0),\\ \text{GD}: \boldsymbol{\eta}_1 &\leftarrow \boldsymbol{\eta}_0 - \mathbf{F}^{-1}(\boldsymbol{\eta}_0)\mathbf{m}^{(\eta_0)} = \mathbf{0} - \mathbf{I}^{-1}\mathbf{m}^{(\eta_0)},\\ \boldsymbol{\tau}^{(\text{new})} &\leftarrow \boldsymbol{\phi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta}_1),\end{aligned} \tag{11}$$

where α is the momentum weight and β is the stepsize.

We now discuss how to include momentum in moving (local) coordinates $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$, where $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ are GNCs defined at $\boldsymbol{\tau}^{(\text{cur})}$ and $\boldsymbol{\tau}^{(\text{new})}$, respectively. Note that $\boldsymbol{\tau}^{(\text{new})}$ is represented by $\boldsymbol{\eta}_1$ in current coordinate $\boldsymbol{\eta}$ and by $\boldsymbol{\xi}_0$ in new coordinate $\boldsymbol{\xi}$. To compute $\mathbf{w}^{(\xi_0)}$ in coordinate $\boldsymbol{\xi}$ at the next iteration, we perform the following two steps.

Step 1: (In Coordinate $\boldsymbol{\eta}$) We transport momentum $\mathbf{m}^{(\eta_0)}$ at point $\boldsymbol{\tau}^{(\text{cur})}$ via the Euclidean transport map to point $\boldsymbol{\tau}^{(\text{new})}$, which is similar to the transport step in Eq. (3).

Since performing NGD/GD alone ignores Christoffel symbols, we suggest ignoring the symbols in the approximation of the Euclidean transport map $T_{\eta_0\to\eta_1}(\mathbf{m}^{(\eta_0)})$ using coordinate $\boldsymbol{\eta}$. This gives the following update:

momentum transport in $\boldsymbol{\eta}$

$$\text{Transport}: \ \mathbf{w}^{(\eta_1)} \leftarrow \mathbf{m}^{(\eta_0)}. \tag{12}$$

The approximation keeps the dominant term of the map and ignores negligible terms. In a global coordinate, this approximation is known as the Euclidean vector transport map (Jeuris et al., 2012) and the vector-transport-free map (Godaz et al., 2021). A similar approximation is also suggested in Riemannian sampling (Girolami & Calderhead, 2011) to avoid solving an implicit leapfrog update. Using GNCs, we can make a better approximation by adding the second dominant term (see Appx. F) defined by the Christoffel symbols. For example, in SPD cases, the second dominant term (see Eq. (55) in Appx. F) can be explicitly computed and is negligible compared to the first term. The computation can be similarly carried out on submanifolds. Moreover, if GNC $\boldsymbol{\eta}$ is a symmetric matrix as will be shown in Eq. (14), the second dominant term vanishes even if the Christoffel symbols are non-vanishing. On the other hand, it is nontrivial to compute the second term in a global coordinate $\boldsymbol{\tau}$.

Step 2: (At Point $\boldsymbol{\tau}^{(\text{new})}$) We coordinate-transform momentum $\mathbf{w}^{(\eta_1)}$ from coordinate $\boldsymbol{\eta}$ to coordinate $\boldsymbol{\xi}$ and return the transformation as $\mathbf{w}^{(\xi_0)}$, where $\boldsymbol{\eta}_1$ and $\boldsymbol{\xi}_0$ represent $\boldsymbol{\tau}^{(\text{new})}$.

By construction of GNCs (shown in Fig. 1), coordinates $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ represent the global coordinate $\boldsymbol{\tau}$ as $\boldsymbol{\tau} = \phi_{\tau^{(\text{cur})}}(\boldsymbol{\eta}) = \phi_{\tau^{(\text{new})}}(\boldsymbol{\xi})$, where $\boldsymbol{\xi}$ is the GNC associated to $\boldsymbol{\tau}^{(\text{new})}$ at the next iteration. We transform Euclidean momentum $\mathbf{w}^{(\eta_1)}$ as a Euclidean (gradient) vector from coordinate $\boldsymbol{\eta}$ to coordinate $\boldsymbol{\xi}$ via the (Euclidean) chain rule,

momentum coordinate-transform from $\boldsymbol{\eta}$ to $\boldsymbol{\xi}$

$$\left(\mathbf{w}^{(\xi_0)}\right)^T = \left(\mathbf{w}^{(\eta_1)}\right)^T \mathbf{J}(\boldsymbol{\xi}_0); \;\; \mathbf{J}(\boldsymbol{\xi}) := \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\xi}}, \quad (13)$$

where $\boldsymbol{\xi}_0 = \mathbf{0}$, $\boldsymbol{\eta} = \phi^{-1}_{\tau^{(\text{cur})}} \circ \phi_{\tau^{(\text{new})}}(\boldsymbol{\xi})$, and $\mathbf{J}(\boldsymbol{\xi})$ is the Jacobian. Thanks to GNCs, the Jacobian computation can be simplified by evaluating at $\boldsymbol{\xi}_0 = \mathbf{0}$.

We can see that our update in Eq. (11)-(13) is a practical approximation of update (3). The Euclidean transport map is required for the use of the (Euclidean) chain rule and simplification of the Jacobian computation in Eq. (13). Our update shares the same spirit of Cartan's method of moving frames (Ivey & Landsberg, 2003) by using only Euclidean/exterior derivatives. The Christoffel symbols could be computed via a Lie bracket [1] due to Cartan's structure equations and the Maurer-Cartan form (Piuze et al., 2015).

3.2. Designing GNCs for SPD Manifolds

We describe how to design GNCs on SPD manifolds. This procedure explains the construction of existing coordinates in Lin et al. (2021a); Godaz et al. (2021).

To mimic the SNC in Eq. (9), consider the matrix factorization $\boldsymbol{\tau}^{(\text{cur})} = \mathbf{A}^{(\text{cur})}\left(\mathbf{A}^{(\text{cur})}\right)^T$, where $\mathbf{A}^{(\text{cur})}$ is invertible (not a Cholesky). This asymmetric factorization contains a *matrix Lie group structure* in $\mathbf{A}^{(\text{cur})}$ for submanifolds while the symmetric one (i.e., $\left(\boldsymbol{\tau}^{(\text{cur})}\right)^{1/2}$) in Eq. (9) does not. The Christoffel symbols are non-vanishing due to the asymmetry. Nevertheless, this factorization allows us to obtain a coordinate by approximating the map $\psi_{\tau^{(\text{cur})}}$ in SNC as

$$\begin{aligned}\phi_{\tau^{(\text{cur})}}(\boldsymbol{\eta}) :=& \mathbf{A}^{(\text{cur})}\text{Expm}(\boldsymbol{\eta})\left(\mathbf{A}^{(\text{cur})}\right)^T \\ =& \mathbf{A}^{(\text{cur})}\text{Expm}(\tfrac{1}{2}\boldsymbol{\eta})\text{Expm}^T(\tfrac{1}{2}\boldsymbol{\eta})\left(\mathbf{A}^{(\text{cur})}\right)^T = \mathbf{A}\mathbf{A}^T, \quad (14)\end{aligned}$$

where $\mathbf{A} := \mathbf{A}^{(\text{cur})}\text{Expm}(\frac{1}{2}\boldsymbol{\eta})$ and $\boldsymbol{\eta}$ is a symmetric matrix. Factorization $\boldsymbol{\tau} = \mathbf{A}\mathbf{A}^T$ can be non-unique in $\mathbf{A}$. We only require coordinate $\boldsymbol{\eta}$ to be unique instead of $\mathbf{A}$. To satisfy uniqueness in $\boldsymbol{\eta}$ required in Assumption 3, we can restrict $\boldsymbol{\eta}$ to be in a subspace of $\mathbb{R}^{k\times k}$ such as the symmetric matrix space. Coordinate $\boldsymbol{\eta}$ is a GNC at $\boldsymbol{\tau}^{(\text{cur})}$ (shown in Appx. H.1). Using other factorizations, we obtain more GNCs:

- $\phi_{\tau^{(\text{cur})}}(\boldsymbol{\eta}) = \mathbf{B}^{-T}\mathbf{B}^{-1}$, with $\mathbf{B} := \mathbf{B}^{(\text{cur})}\text{Expm}(-\frac{1}{2}\boldsymbol{\eta})$
- $\phi_{\tau^{(\text{cur})}}(\boldsymbol{\eta}) = \mathbf{C}^T\mathbf{C}$, with $\mathbf{C} := \text{Expm}(\frac{1}{2}\boldsymbol{\eta})\mathbf{C}^{(\text{cur})}$

where $\boldsymbol{\eta}$ is a symmetric matrix in both cases. For (unique) Cholesky factor $\mathbf{A}$, we obtain a new GNC by restricting $\boldsymbol{\eta}$ to be a lower-triangular matrix with proper scaling factors.

[1] There is a Lie group structure for the coordinate-transformation in the frame bundle of a general (sub)manifold.

The GNC considered in Eq. (14) is similar to local coordinates in structured NGD, where $\mathbf{A}$ is referred to as an auxiliary coordinate. The authors of structured NGD (Lin et al., 2021a) introduce a similar local coordinate as $\mathbf{A} := \mathbf{A}^{(\text{cur})}\text{Expm}(\boldsymbol{\eta})$ in Gaussian cases without providing the construction and mentioning other local coordinates. The metric is not orthonormal in their coordinates. Our procedure sheds light on the construction and the role of local coordinates in structured NGD. For example, our construction explains why $\mathbf{A} = \mathbf{A}^{(\text{cur})}\text{Expm}(\boldsymbol{\eta})$ instead of $\mathbf{A} = \text{Expm}(\boldsymbol{\eta})\mathbf{A}^{(\text{cur})}$ is used in structured NGD for the factorization $\boldsymbol{\tau} = \mathbf{A}\mathbf{A}^T$. Using $\mathbf{A} = \text{Expm}(\boldsymbol{\eta})\mathbf{A}^{(\text{cur})}$ in structured NGD makes it difficult to compute natural-gradients since the metric is not orthonormal. In contrast, our coordinates explicitly orthonormalize the metric, which makes it easy to compute (natural) gradients and include momentum.

Using the GNC in Eq. (14), our update in Eq. (11)-(13) can be simplified as shown in Appx. H.3.

Godaz et al. (2021) consider a special case for full SPD manifolds with symmetric $\boldsymbol{\eta}$ and a unique factor $\mathbf{A}$. However, their method is non-constructive and limited to SPD manifolds. Our construction generalizes their method by allowing $\boldsymbol{\eta}$ to be an asymmetric matrix, using a non-unique factor $\mathbf{A}$, and extending their method to SPD submanifolds.

3.3. Designing GNCs for SPD Submanifolds

Although SNCs are unknown on SPD submanifolds, GNCs allow us to work on a class of SPD submanifolds by noting that $\mathbf{A}$ is in the general linear group $\text{GL}^{k\times k}$ known as a *matrix Lie group*. Matrix structures are preserved under matrix multiplication and the matrix exponential. The coordinate space of $\boldsymbol{\eta}$ is a subspace of the tangent space of $\text{Expm}(\boldsymbol{\eta})$ at $\text{Expm}(\boldsymbol{\eta}_0) = \mathbf{I}$ known as the *Lie algebra*. Thus, Assumption 4 is satisfied. These observations let us consider the following class of SPD submanifolds[2]:

$$\mathcal{M} = \left\{\boldsymbol{\tau} = \mathbf{A}\mathbf{A}^T \in \mathbb{R}^{k\times k} \mid \mathbf{A} \in \text{ Connected Subgroup of } \text{GL}^{k\times k}\right\}.$$

We give two examples of SPD submanifolds, where we construct GNCs. GNCs can be constructed for other submanifolds considered in Lin et al. (2021a). For example, we will use a Heisenberg SPD submanifold suggested by Lin et al. (2021a) in our experiments. In Sec. 3.3.1, we present our update without the Bayesian and Gaussian assumptions. Our update recovers structured NGD as a special case by making a Gaussian assumption. In Sec. 3.3.2, we demonstrate the scalability of our approach.

3.3.1. Gaussian Family as a SPD submanifold

We will first consider a SPD submanifold in a non-Bayesian and non-Gaussian setting. We then show how our update relates to structured NGD in Gaussian cases where Gaus-

[2] Another Lie group structure is induced by a SPD submanifold.

sian gradient identities are available. Consider the SPD submanifold, where $k = d + 1$,

$$\mathcal{M} = \left\{ \boldsymbol{\tau} = \begin{bmatrix} \mathbf{V} & \boldsymbol{\mu} \\ \boldsymbol{\mu}^T & 1 \end{bmatrix} \in \mathbb{R}^{k\times k} \mid \boldsymbol{\tau} \succ 0 \right\}.$$

Note that $\boldsymbol{\tau}^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}^{-1} & -\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ -\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1} & 1+\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \end{bmatrix}$, for $\boldsymbol{\tau} \in \mathcal{M}$ and $\boldsymbol{\Sigma} := \mathbf{V} - \boldsymbol{\mu}\boldsymbol{\mu}^T$. Thus, $\boldsymbol{\Sigma} \succ 0$ since $\boldsymbol{\tau}^{-1} \succ 0$. Then, letting $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$, $\mathcal{M}$ can be reexpressed as

$$\mathcal{M} = \left\{ \boldsymbol{\tau} = \mathbf{A}\mathbf{A}^T \mid \mathbf{A} := \begin{bmatrix} \mathbf{L} & \boldsymbol{\mu} \\ \mathbf{0} & 1 \end{bmatrix}, \mathbf{L} \in \mathrm{GL}^{d\times d} \right\}.$$

Observe that $\mathbf{A}$ is subgroup of $\mathrm{GL}^{k\times k}$. We construct a GNC $\phi_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta}) = \mathbf{A}\mathbf{A}^T$ similar to the one in Eq. (14), where

$$\mathbf{A} = \mathbf{A}^{(\mathrm{cur})}\mathrm{Expm}\left(\begin{bmatrix} \frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0 \end{bmatrix} \right), \qquad (15)$$

and $\boldsymbol{\eta} = \{\boldsymbol{\eta}_L, \boldsymbol{\eta}_\mu\}$, $\boldsymbol{\eta}_L \in \mathbb{R}^{d\times d}$ is a symmetric matrix so that Assumption 3 is satisfied. The scalars highlighted in red are to satisfy Assumption 2 that the metric is orthonormal.

There is another GNC $\phi_{\boldsymbol{\tau}^{(\mathrm{cur})}}(\boldsymbol{\eta}) = \mathbf{A}\mathbf{A}^T$ where

$$\begin{aligned} \mathbf{A} =& \mathbf{A}^{(\mathrm{cur})} \begin{bmatrix} \mathrm{Expm}(\frac{1}{2}\boldsymbol{\eta}_L) & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 1 \end{bmatrix} \\ =& \begin{bmatrix} \mathbf{L}^{(\mathrm{cur})}\mathrm{Expm}(\frac{1}{2}\boldsymbol{\eta}_L) & \boldsymbol{\mu}^{(\mathrm{cur})} + \frac{1}{\sqrt{2}}\mathbf{L}^{(\mathrm{cur})}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 1 \end{bmatrix}. \end{aligned} \qquad (16)$$

We can obtain Eq. (16) from Eq. (15) (shown in Appx. G.1).

Using the GNC in either Eq. (15) or Eq. (16), the Euclidean gradient needed in Eq. (11) is given by

$$\mathbf{g}(\boldsymbol{\eta}_0) = \{\mathbf{L}^T\mathbf{g}_1\mathbf{L},\ \sqrt{2}\mathbf{L}^T(\mathbf{g}_1\boldsymbol{\mu} + \mathbf{g}_2)\},$$

where $\mathbf{L} = \mathbf{L}^{(\mathrm{cur})}$, $\boldsymbol{\mu} = \boldsymbol{\mu}^{(\mathrm{cur})}$, $\mathbf{g}(\boldsymbol{\tau}^{(\mathrm{cur})}) = \begin{bmatrix} \mathbf{g}_1 & \mathbf{g}_2 \\ \mathbf{g}_2^T & 0 \end{bmatrix}$ is a (symmetric) Euclidean gradient w.r.t. $\boldsymbol{\tau} \in \mathcal{M}$. The vector-Jacobian product in Eq. (13) is easy to compute by using coordinate Eq. (16). Note that the computation of this product does depend on the choice of GNCs.

Our update can be used for SPD estimation such as metric learning, trace regression, and dictionary learning from a deterministic matrix manifold optimization perspective. Neither Bayesian nor Gaussian assumptions are required.

In Gaussian cases, Calvo & Oller (1990) show that a d-dimensional Gaussian $\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ can be reexpressed as an augmented $(d+1)$-dimensional Gaussian $\mathcal{N}(\mathbf{x}|\mathbf{0}, \boldsymbol{\tau})$ with zero mean and covariance $\boldsymbol{\tau}$, where $\mathbf{x}^T = [\mathbf{z}^T, 1]$ and $\boldsymbol{\tau}$ is on this SPD submanifold. Hosseini & Sra (2015) consider this reparametrization for maximum likelihood estimation (MLE) of Gaussian mixture models, but their method is only guaranteed to converge to this particular submanifold at the optimum as they use the Riemannian maps for the corresponding full SPD manifold. On the contrary, our update not only stays on this SPD submanifold at each iteration but is also applicable to other SPD submanifolds. Our approach expands the scope of structured NGD originally proposed as a Bayesian estimator to a maximum likelihood estimator for Gaussian mixtures with *structured covariances* $\boldsymbol{\Sigma}$ where existing methods such as Hosseini & Sra (2015) and the expectation-maximization algorithm cannot be applied.

We now show that structured NGD is a special case of our update. The update of structured NGD (Lin et al., 2021a) for Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with the Fisher-Rao metric is

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \gamma\boldsymbol{\Sigma}\mathbf{g}_\mu, \quad \mathbf{L} \leftarrow \mathbf{L}\mathrm{Expm}(-\gamma\mathbf{L}^T\mathbf{g}_\Sigma\mathbf{L}), \qquad (17)$$

where $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$ and γ is the stepsize for structured NGD.

As shown in Appx. G.2, we can reexpress $\mathbf{g}_1$ and $\mathbf{g}_2$ using Gaussian gradients $\mathbf{g}_\mu$ and $\mathbf{g}_\Sigma$ as $\mathbf{g}_1 = \mathbf{g}_\Sigma$ and $\mathbf{g}_2 = \frac{1}{2}(\mathbf{g}_\mu - 2\mathbf{g}_\Sigma\boldsymbol{\mu})$. Thus, we reexpress $\mathbf{g}(\boldsymbol{\eta}_0)$ as

$$\mathbf{g}(\boldsymbol{\eta}_0) = \{\mathbf{L}^T\mathbf{g}_\Sigma\mathbf{L},\ \frac{1}{\sqrt{2}}\mathbf{L}^T\mathbf{g}_\mu\}. \qquad (18)$$

Since the affine-invariant metric is twice the Fisher-Rao metric (see Eq. (4)) , we set stepsize $\beta = 2\gamma$ and momentum weight $\alpha = 0$. Our update (without momentum) in Eq. (11) recovers structured NGD in Eq. (17) by using the Gaussian gradients in Eq. (18) and coordinate (16).

Using the GNC in (16), our update essentially performs NGD in the expectation parameter space $\{\boldsymbol{\mu}, \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T\}$ (induced by $\boldsymbol{\tau}$) of a Gaussian by considering the Gaussian as an exponential family. Likewise, using another GNC such as $\boldsymbol{\tau} = \mathbf{B}^{-T}\mathbf{B}^{-1}$, our update performs NGD in the natural parameter space $\{-\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}, \boldsymbol{\Sigma}^{-1}\}$ (induced by $\boldsymbol{\tau}^{-1}$), which is the (Bregman) dual space of the expectation parameter space (Khan & Lin, 2017). When $\mathbf{A}$ and $\mathbf{B}^{-T}$ have the same matrix structure, our updates in the parameter spaces agree to first-order $O(\beta)$ with respect to stepsize β due to the linear invariance of NGD.

3.3.2. Rank-one SPD Submanifold

Now, we give an example of sparse SPD matrices to illustrate the usage of our update in high-dimensional problems.

Consider the following SPD submanifold, where $k = d + 1$.

$$\mathcal{M} = \left\{ \boldsymbol{\tau} = \begin{bmatrix} a^2 & a\mathbf{b}^T \\ a\mathbf{b} & \mathbf{b}\mathbf{b}^T + \mathrm{Diag}(\mathbf{c}^2) \end{bmatrix} \in \mathbb{R}^{k\times k} \mid \boldsymbol{\tau} \succ 0 \right\}.$$

To construct GNCs, we reexpress $\mathcal{M}$ as

$$\mathcal{M} = \left\{ \boldsymbol{\tau} = \mathbf{A}\mathbf{A}^T \mid \mathbf{A} := \begin{bmatrix} a & \mathbf{0} \\ \mathbf{b} & \mathrm{Diag}(\mathbf{c}) \end{bmatrix}, a > 0, \mathbf{c} > 0 \right\},$$

where $\mathrm{Diag}(\mathbf{c})$ is in the diagonal subgroup of $\mathrm{GL}^{d\times d}$. Observe that $\mathbf{A}$ is in a subgroup of $\mathrm{GL}^{k\times k}$. Thus, we can construct the following GNC,

$$\boldsymbol{\tau} = \mathbf{A}\mathbf{A}^T; \ \ \mathbf{A} = \mathbf{A}^{(\mathrm{cur})}\mathrm{Expm}(\mathbf{D} \odot \boldsymbol{\eta}),$$

where $\boldsymbol{\eta} = \begin{bmatrix} \eta_a & \mathbf{0} \\ \boldsymbol{\eta}_b & \mathrm{Diag}(\boldsymbol{\eta}_c) \end{bmatrix} \in \mathbb{R}^{k\times k}$, $\odot$ is elementwise product, and the constant matrix $\mathbf{D} = \frac{1}{2}\begin{bmatrix} 1 & \mathbf{0} \\ \sqrt{2}\mathbf{1} & \mathbf{I} \end{bmatrix}$ (with $\mathbf{1}$ denoting a matrix of ones) enforces Assumption 2.

Our Matrix-inverse-free Update

1: Each T iterations, update $\mathbf{m}_K, \mathbf{m}_C, \mathbf{K}, \mathbf{C}$
Obtain $\boldsymbol{\mu}_{AA} \otimes \boldsymbol{\mu}_{GG}$ to approximate $\nabla^2_\mu \ell(\boldsymbol{\mu})$
$\mathbf{m}_K \leftarrow \alpha_1 \mathbf{m}_K + \frac{\beta_1}{2d}(\mathrm{Tr}(\mathbf{H}_C)\mathbf{H}_K + c^2\mathbf{K}^T\mathbf{K} - d\mathbf{I}_p)$
$\mathbf{m}_C \leftarrow \alpha_1 \mathbf{m}_C + \frac{\beta_1}{2p}(\mathrm{Tr}(\mathbf{H}_K)\mathbf{H}_C + \kappa^2\mathbf{C}^T\mathbf{C} - p\mathbf{I}_d)$
$\mathbf{K} \leftarrow \mathbf{K}\mathrm{Expm}(-\mathbf{m}_K) \approx \mathbf{K}(\mathbf{I}_p - \mathbf{m}_K)$
$\mathbf{C} \leftarrow \mathbf{C}\mathrm{Expm}(-\mathbf{m}_C) \approx \mathbf{C}(\mathbf{I}_d - \mathbf{m}_C)$
2: $\mathbf{M}_\mu \leftarrow \alpha_2 \mathbf{M}_\mu + \mathbf{C}\mathbf{C}^T\mathrm{vec}^{-1}(\nabla_\mu \ell(\boldsymbol{\mu}))\mathbf{K}\mathbf{K}^T$
3: $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_2(\mathrm{vec}(\mathbf{M}_\mu) + \gamma\nabla_\mu(\boldsymbol{\mu}))$

KFAC Optimizer

1: Each T iterations, update $(\mathbf{K}\mathbf{K}^\mathbf{T})^{-1}, (\mathbf{C}\mathbf{C}^\mathbf{T})^{-1}, \mathbf{K}\mathbf{K}^\mathbf{T}, \mathbf{C}\mathbf{C}^\mathbf{T}$
Obtain $\boldsymbol{\mu}_{AA} \otimes \boldsymbol{\mu}_{GG}$ to approximate $\nabla^2_\mu \ell(\boldsymbol{\mu})$
$(\mathbf{K}\mathbf{K}^\mathbf{T})^{-1} \leftarrow \theta\,(\mathbf{K}\mathbf{K}^\mathbf{T})^{-1} + (1-\theta)\boldsymbol{\mu}_{AA}$
$(\mathbf{C}\mathbf{C}^\mathbf{T})^{-1} \leftarrow \theta\,(\mathbf{C}\mathbf{C}^\mathbf{T})^{-1} + (1-\theta)\boldsymbol{\mu}_{GG}$
$\mathbf{K}\mathbf{K}^\mathbf{T} \leftarrow ((\mathbf{K}\mathbf{K}^\mathbf{T})^{-1} + \lambda\mathbf{I}_p)^{-1}$
$\mathbf{C}\mathbf{C}^\mathbf{T} \leftarrow ((\mathbf{C}\mathbf{C}^\mathbf{T})^{-1} + \lambda\mathbf{I}_d)^{-1}$
2: $\mathbf{M}_\mu \leftarrow \alpha_2 \mathbf{M}_\mu + \mathbf{C}\mathbf{C}^\mathbf{T}\mathrm{vec}^{-1}(\nabla_\mu \ell(\boldsymbol{\mu}))\mathbf{K}\mathbf{K}^\mathbf{T}$
3: $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta_2(\mathrm{vec}(\mathbf{M}_\mu) + \gamma\nabla_\mu(\boldsymbol{\mu}))$

Figure 2. In our update, we denote $\mathbf{H}_K := \mathbf{K}^T\boldsymbol{\mu}_{AA}\mathbf{K}$, $\mathbf{H}_C := \mathbf{C}^T\boldsymbol{\mu}_{GG}\mathbf{C}$, $\kappa^2 := \lambda\mathrm{Tr}(\mathbf{K}^T\mathbf{K})$, and $c^2 := \lambda\mathrm{Tr}(\mathbf{C}^T\mathbf{C})$, where $\mathrm{vec}^{-1}(\boldsymbol{\mu}) \in \mathbb{R}^{d\times p}$, $\mathbf{C} \in \mathbb{R}^{d\times d}$, $\mathbf{K} \in \mathbb{R}^{p\times p}$. Note that we merge factors $\frac{1}{2\sqrt{d}}$ and $\frac{1}{2\sqrt{p}}$ in Eq. (23) into the updates in $\mathbf{m}_K$ and $\mathbf{m}_C$, respectively (see Eq. (86) in Appx. I for a justification). We use the linear truncation of the matrix exponential function. Our update does not require explicit matrix inverses. We can also pre-compute $\mathbf{C}\mathbf{C}^T$ and $\mathbf{K}\mathbf{K}^T$ when $\mathbf{T} > 1$. In KFAC, a damping term $\lambda\mathbf{I}$ is introduced to handle the singularity of $(\mathbf{K}\mathbf{K}^\mathbf{T})^{-1}$ and $(\mathbf{C}\mathbf{C}^\mathbf{T})^{-1}$. We introduce a similar damping term in κ^2 and c^2 (see Appx. I for a derivation) to improve numerical stability. Our update and KFAC include momentum weight α_2 for layer-wise NN weights $\boldsymbol{\mu}$ and (L2) weight decay γ. In our update, we also introduce momentum weight α_1 in the SPD preconditioner. Our update can use a larger stepsize β_2 than KFAC.

The Euclidean gradient $\mathbf{g}(\boldsymbol{\eta}_0)$ in Eq. (11) can be efficiently computed via sparse Euclidean gradients w.r.t. $\mathbf{A}$. It can also be computed via dense Euclidean gradients w.r.t. $\boldsymbol{\tau}$ as

$$\mathbf{g}(\boldsymbol{\eta}_0) = \begin{bmatrix} a(ag_1 + 2\mathbf{g}_2^T\mathbf{b}) + \mathbf{b}^T\mathbf{f} & \mathbf{0} \\ \sqrt{2}\mathbf{c} \odot (a\mathbf{g}_2 + \mathbf{f}) & \mathrm{Diag}(\mathbf{c}^2 \odot \mathrm{diag}(\mathbf{G}_3)) \end{bmatrix},$$

where $\mathbf{f} = \mathbf{G}_3\mathbf{b}$ and $\mathbf{g}(\boldsymbol{\tau}^{(\mathrm{cur})}) = \begin{bmatrix} g_1 & \mathbf{g}_2^T \\ \mathbf{g}_2 & \mathbf{G}_3 \end{bmatrix}$ is a (symmetric) Euclidean gradient w.r.t. $\boldsymbol{\tau}$. We assume that we can directly compute $\mathbf{f}$ and $\mathrm{diag}(\mathbf{G}_3)$ without computing $\mathbf{G}_3$. The computation of Eq. (13) can be found in Appx. D.

On this submanifold, it is easy to scale up our update (11) to high-dimensional settings, by truncating the matrix exponential function as $\mathrm{Expm}(\mathbf{N}) \approx \mathbf{I} + \mathbf{N} + \frac{1}{2}\mathbf{N}^2$. This truncation preserves the structure and non-singularity of $\mathbf{A}$.

4. Generalization for Deep Learning

It is useful to design preconditioned GD by exploiting the submanifold structure of the preconditioner. We will use that structure to design inverse-free structured matrix optimizers for low-precision floating-point training schemes in DL.

To solve a minimization problem $\min_{\mu\in\mathbb{R}^k} \ell(\boldsymbol{\mu})$, Lin et al. (2021b) propose using structured NGD as preconditioned GD with a SPD preconditioner $\mathbf{S} := \mathbf{B}\mathbf{B}^T$,

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta\mathbf{S}^{-1}\mathbf{g}_\mu, \quad \mathbf{B} \leftarrow \mathbf{B}\mathrm{Expm}\big(\frac{\beta}{2}\mathbf{B}^{-1}\mathbf{g}_{S^{-1}}\mathbf{B}^{-T}\big), \quad (19)$$

where $\mathbf{g}_{S^{-1}} := \frac{1}{2}(\nabla^2_\mu \ell(\boldsymbol{\mu}) - \mathbf{S})$, and $\mathbf{g}_\mu := \nabla_\mu \ell(\boldsymbol{\mu})$. Lin et al. (2021b) show that the update in $\mathbf{B}$ can be re-expressed in terms of $\mathbf{S}$ as $\mathbf{S} \leftarrow (1-\beta)\mathbf{S} + \beta\nabla^2_\mu \ell(\boldsymbol{\mu}) + O(\beta^2)$. Similar to Newton's update, Eq. (19) is linearly (Lie group) invariant in $\boldsymbol{\mu}$. When $\mathbf{S}$ is a SPD submanifold, this update has a structural (Lie subgroup) invariant[3]. However, this update requires a matrix inverse which can be slow and numerically unstable in large-scale training due to the use of low-precision floating-point training schemes.

[3]This is a Lie-group structural invariant in $\boldsymbol{\mu}$ induced by a structured SPD preconditioner (Lin et al., 2021b).

Eq. (19) can be obtained by considering the inverse of the preconditioner $\boldsymbol{\tau} = \mathbf{S}^{-1} = \mathbf{B}^{-T}\mathbf{B}^{-1}$ as a SPD manifold. The update of $\mathbf{B}$ can be obtained by using the GNC in Sec. 3.2 as $\boldsymbol{\tau} = \mathbf{B}^{-T}\mathbf{B}^{-1}$ where $\mathbf{B} = \mathbf{B}^{(\mathrm{cur})}\mathrm{Expm}(-\frac{1}{2}\boldsymbol{\eta})$. The minus sign (see Eq. (73) in Appx. H.3) is canceled out by another minus sign in the GD update of Eq. (11).

We can obtain a matrix-inverse-free update (see Appx. H.3):

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta\mathbf{A}\mathbf{A}^T\mathbf{g}_\mu, \quad (20)$$

by changing the GNC to $\mathbf{S}^{-1} = \boldsymbol{\tau} = \mathbf{A}\mathbf{A}^T$ where $\mathbf{A} = \mathbf{A}^{(\mathrm{cur})}\mathrm{Expm}(\frac{1}{2}\boldsymbol{\eta})$. The update of $\mathbf{A}$ is also matrix-inverse-free (see Sec. 3.2). We can obtain the same update by considering $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ with $\boldsymbol{\Sigma} = \mathbf{S}^{-1}$ as a submanifold in Sec. 3.3.1 since structured NGD is a special case of ours.

To approximate the Hessian $\nabla^2_\mu \ell(\boldsymbol{\mu})$ in DL as required in $\mathbf{g}_{S^{-1}}$, we consider the KFAC approximation (Martens & Grosse, 2015). For simplicity, consider the loss function $\ell(\boldsymbol{\mu})$ defined by one hidden layer of a neural network (NN), where $k = pd$, $\mathrm{vec}^{-1}(\boldsymbol{\mu}) \in \mathbb{R}^{d\times p}$ is a learnable weight matrix, $\mathrm{vec}(\cdot)$ is the vectorization function. In KFAC (summarized in Fig. 2), the Hessian at each layer of a NN with many layers is approximated by a Kronecker-product structure between two dense symmetric positive semi-definite matrices as $\nabla^2_\mu \ell(\boldsymbol{\mu}) \approx \boldsymbol{\mu}_{AA} \otimes \boldsymbol{\mu}_{GG}$, where matrices $\boldsymbol{\mu}_{AA} \in \mathbb{R}^{p\times p}$ and $\boldsymbol{\mu}_{GG} \in \mathbb{R}^{d\times d}$ are computed as suggested by the authors and $\otimes$ denotes the Kronecker product.

We consider a Kronecker-product structured submanifold with $k = pd$ to exploit the structure of the approximation:

$$\mathcal{M} = \Big\{\boldsymbol{\tau} = \mathbf{U} \otimes \mathbf{W} \in \mathbb{R}^{pd\times pd} \mid \boldsymbol{\tau} \succ 0\Big\},$$

where matrices $\mathbf{U} \in \mathbb{R}^{p\times p}$ and $\mathbf{W} \in \mathbb{R}^{d\times d}$ are both SPD. We can reexpress this submanifold as

$$\mathcal{M} = \Big\{\boldsymbol{\tau} = \mathbf{A}\mathbf{A}^T \mid \mathbf{A} := \mathbf{K} \otimes \mathbf{C}\Big\}, \quad (21)$$

where $\mathbf{U} = \mathbf{K}\mathbf{K}^T$, $\mathbf{W} = \mathbf{C}\mathbf{C}^T$, $\mathbf{K} \in \mathbb{R}^{p\times p}$, $\mathbf{C} \in \mathbb{R}^{d\times d}$. $\mathbf{K}$ and $\mathbf{C}$ are (sub)groups of $\mathrm{GL}^{p\times p}$ and $\mathrm{GL}^{d\times d}$, respectively.

By exploiting the Kronecker structure in $\mathbf{A} = \mathbf{K} \otimes \mathbf{C}$, we can reexpress the update of $\boldsymbol{\mu}$ in (20) as:

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \beta \text{vec}\left(\mathbf{C}\mathbf{C}^T \text{vec}^{-1}(\mathbf{g}_\mu)\mathbf{K}\mathbf{K}^T\right). \quad (22)$$

Now, we describe how to update $\mathbf{A} = \mathbf{K} \otimes \mathbf{C}$ by using the structure of the SPD submanifold $\boldsymbol{\tau} = \mathbf{A}\mathbf{A}^T$. Unfortunately, the affine-invariant metric defined in Eq. (4) is singular. To enable the usage of this submanifold, we consider a block-diagonal approximation to update blocks $\mathbf{K}$ and $\mathbf{C}$. Given each block, we construct a normal coordinate to orthonormalize the metric with respect to the block while keeping the other block frozen. Such an approximation of the affine-invariant metric leads to a block-diagonal approximated metric. For blocks $\mathbf{K}$ and $\mathbf{C}$, we consider the following blockwise GNCs $\boldsymbol{\eta}_K$ and $\boldsymbol{\eta}_C$, respectively:

$$\mathbf{A} = \left(\mathbf{K}^{(\text{cur})} \text{Expm}\left(\frac{\boldsymbol{\eta}_K}{2\sqrt{d}}\right)\right) \otimes \mathbf{C}^{(\text{cur})}, \quad \mathbf{A} = \mathbf{K}^{(\text{cur})} \otimes \left(\mathbf{C}^{(\text{cur})} \text{Expm}\left(\frac{\boldsymbol{\eta}_C}{2\sqrt{p}}\right)\right), \quad (23)$$

where both $\boldsymbol{\eta}_K \in \mathbb{R}^{p\times p}$ and $\boldsymbol{\eta}_C \in \mathbb{R}^{d\times d}$ are symmetric matrices and $\mathbf{A}^{(\text{cur})} = \mathbf{K}^{(\text{cur})} \otimes \mathbf{C}^{(\text{cur})}$. The scalars highlighted in red are needed to orthonormalize the block-diagonal metric.

Using these blockwise GNCs, our update is summarized in Fig. 2 (see Appx. I for a derivation), where similar to KFAC, we further introduce momentum weight α_2 for $\boldsymbol{\mu}$, weight decay γ, and damping weight λ. In practice, we truncate the matrix exponential function: the quadratic truncation $\text{Expm}(\mathbf{N}) \approx \mathbf{I} + \mathbf{N} + \frac{1}{2}\mathbf{N}^2$ ensures the non-singularity of $\mathbf{K}$ and $\mathbf{C}$. In our DL experiments, we observed that the linear truncation $\text{Expm}(\mathbf{N}) \approx \mathbf{I} + \mathbf{N}$ also works well.

We can also develop sparse Kronecker updates while original KFAC does not admit a sparse update. For example, our approach allows us to include sparse structures by considering $\mathbf{K}$ and $\mathbf{C}$ with sparse group structures in Eq. (21).

Our approach allows us to go beyond the KFAC approximation by exploiting other structures in the Hessian approximation of $\nabla^2_\mu \ell(\boldsymbol{\mu})$ and developing inverse-free update schemes by using SPD submanifolds to respect the structures.

5. Numerical Results

5.1. Results on Synthetic Examples

To validate our proposed updates, we consider several optimization problems. In the first three examples, we consider manifold optimization on SPD matrices $\mathcal{S}^{k\times k}_{++}$, where the Riemannian maps admit a closed-form expression (shown in Table 1). We evaluate our method on the metric nearness problem considered in Lin et al. (2021a), a log-det optimization problem considered in Han et al. (2021), and a MLE problem of a Gaussian mixture model (GMM) considered in Hosseini & Sra (2015). We consider structured NGD (Lin et al., 2021a), RGD, and existing Riemannian momentum methods such as the ones presented by Ahn & Sra (2020), Alimisis et al. (2020; 2021) as baselines (see Appx. J for our implementation of these methods). All methods are trained using the same stepsize and momentum weight. Our updates and structured NGD can use a larger stepsize than the other methods. The exact Riemannian maps are not numerically stable in high-dimensional settings. From Fig. 3a-3c, we can see that our method performs as well as the Riemannian methods with the exact Riemannian maps in the global coordinate. In the last example, we minimize a 1000-dim Rosenbrock function. We consider the inverse of the preconditioner $\boldsymbol{\tau} = \mathbf{S}^{-1}$ in Eq. (19) as a SPD submanifold. We include momentum in $\boldsymbol{\tau}$ and $\boldsymbol{\mu}$. We compare our update with structured NGD, where both methods use the Heisenberg structure suggested by Lin et al. (2021a) to construct a submanifold. Both methods make use of Hessian information without computing the full Hessian. We consider other baselines: BFGS and Adam. We tune the stepsize for all methods. From Fig. 3d, we can see that adding momentum in the preconditioner could be useful for optimization.

5.2. Results in Deep Learning

To demonstrate our method as a practical Riemannian method in high-dimensional cases, we consider image classification tasks with NN architectures ranging from classical to modern models: VGG-16 (Simonyan & Zisserman, 2014) with the batch normalization, PyramidNet-65 (Han et al., 2017), RegNetX-1.6GF (Radosavovic et al., 2020), RegNetZ-500MF (Dollár et al., 2021), RepVGG-B1G4 (Ding et al., 2021), and ConvMixer-12 (Trockman & Kolter, 2022). We use the KFAC approximation for convolution layers (Grosse & Martens, 2016). Table 6 in Appx. A summarizes the number of learnable parameters. We consider three complex datasets "CIFAR-100", "TinyImageNet-200"[4], and "ImageNet-100"[5]. The hyper-parameter configuration of our update and KFAC can be found in Table 5 in Appx. A. We also consider other baselines such as SGD with momentum, Adam, and AdamW. By default, a L2 weight decay is included in all methods. We also consider a version of SGD and Adam denoted by "No L2", where no weight decay is added. We train all models from scratch for 120 epochs with mini-batch size 128. For all methods, we tune the initial stepsize and then divide the stepsize by 10 every 40 epochs, as suggested by Wilson et al. (2017). We set the weight decay to be 0.01 for all methods. Our method can take a larger stepsize than KFAC for all the NN models. Our method has similar running times as KFAC, as shown in Fig. 4 in the main text, and Fig. 6 in Appx. A. We report the test error rate (i.e., error rate $= 100 -$ accuracy percentage) for all methods in Tables 3 and 4. From Fig. 4, we can see that our method performs better than KFAC and achieves competitive performances among other baselines. More results on other datasets can be found in Fig. 5 in Appx. A.

[4] github.com/tjmoon0104/pytorch-tiny-imagenet
[5] kaggle.com/datasets/ambityga/imagenet100

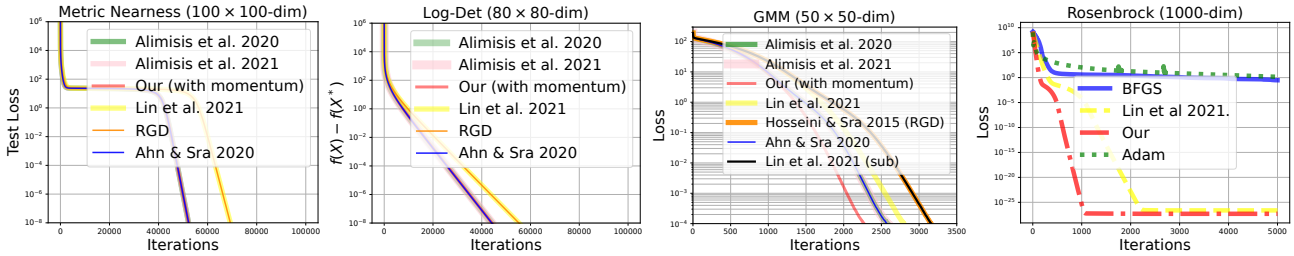


Figure 3. The performance of our updates for optimization problems. Fig. 3a- 3b shows the performance on SPD manifold optimization problems. Our update using approximations of the Riemannian maps achieves a similar performance as existing Riemannian methods using the exact Riemannian maps. Fig. 3c shows the performance on a MLE problem on a Gaussian mixture. The method denoted by "sub" performs updates on a SPD submanifold (see sec. 3.3.1) while the other methods perform updates on a SPD manifold. Note that the loss in Fig. 3c is computed by augmented $(d+1)$-dim Gaussian components suggested by Hosseini & Sra (2015). If we perform updates on the SPD manifold $\mathcal{S}_{++}^{k\times k}$ with $k=d+1$ instead of the submanifold, we cannot obtain the original (non-augmented) d-dim Gaussian components during the iterations since the updates are not guaranteed to stay on the submanifold. Thus, we cannot use the standard MLE loss defined by the d-dim Gaussians. In Fig. 3a- 3c, we use the same stepsize and momentum weight for all methods. Note that our method and Lin et al. (2021a) can use a larger stepsize than the other methods using the exact Riemannian maps. Our method and Lin et al. (2021a) use the quadratic truncation while the other methods use the exact maps. We observe that our method with truncation is more numerically robust than the other methods using the exact maps. Fig. 3d shows the performance using a structured preconditioner to optimize a 1000-dim function, where our update and structured NGD use Hessian information without computing the full Hessian.

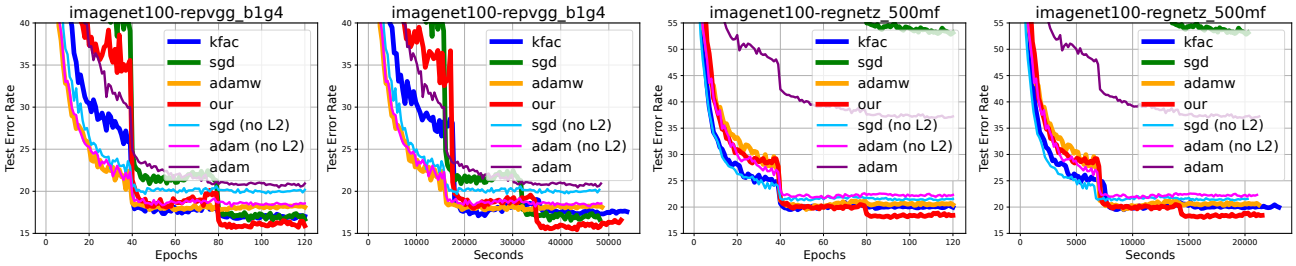


Figure 4. The error curves for optimization in deep NN models. Our updates achieve lower test error rates than the other methods.

Dataset	Method	VGG-16	PyramidNet-65	ConvMixer-12	RegNetX-1.6GF
CIFAR-100	SGD (No L2)	**25.13** (31.01)	30.87 (27.71)	43.13 (30.54)	24.34 (26.67)
	Adam (No L2)	29.60 (31.33)	34.67 (30.20)	38.56 (31.24)	30.09 (26.93)
	AdamW	31.04	28.59	29.46	26.95
	KFAC	27.16	25.92	25.81	23.25
	Ours	25.92	**25.45**	**25.44**	**21.78**
TinyImageNet-200	SGD (No L2)	**39.12** (45.05)	60.36 (42.94)	74.90 (45.26)	41.33 (38.21)
	Adam (No L2)	44.93 (45.34)	63.16 (43.48)	64.34 (46.61)	45.98 (40.89)
	AdamW	45.49	42.27	45.96	40.41
	KFAC	42.20	40.73	43.25	37.01
	Ours	40.03	**40.42**	**41.52**	**34.80**

Table 3. More results about the performance (test error rate) of the methods considered in error curves on the other datasets (shown in Fig. 5 in Appx. A). The results are obtained by averaging over the last 10 iterations.

Dataset	Method	RegNetZ-500MF	RepVGG-B1G4
ImagetNet-100	SGD (No L2)	53.28 (21.45)	17.03 (20.06)
	Adam (No L2)	37.13 (22.18)	20.78 (18.46)
	AdamW	20.58	18.23
	KFAC	20.07	16.97
	Ours	**18.58**	**16.14**

Table 4. Results about the performance (test error rate) of the methods in Fig. 4. The results are obtained by averaging over the last 10 iterations.

6. Conclusion

We propose GNCs to simplify existing Riemannian optimization methods via *metric-preserving* trivializations, which results in practical NGD momentum updates with metric-inverse-free Riemannian/natural gradient computation. Our approach expands the scope of structured NGD to SPD submanifolds arising in machine learning applications and enables the usage of structured NGD beyond Bayesian and Gaussian settings from a manifold optimization perspective. We further develop matrix-inverse-free structured optimizers for DL by exploiting a submanifold structure of SPD preconditioners. An interesting application is to design customized optimizers for a given NN architecture by exploiting a range of submanifolds. Overall, our work provides a new way to design practical manifold optimization methods while taking care of numerical stability in high-dimensional, low numerical precision, and noisy settings.

Acknowledgements

This research was partially supported by the Canada CIFAR AI Chair Program, the NSF under grants CCF-2112665, DMS-1345013, DMS-1813635, and the AFOSR under grant FA9550-18-1-0288.

References

Absil, P.-A., Mahony, R., and Sepulchre, R. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Ahn, K. and Sra, S. From Nesterov's estimate sequence to Riemannian acceleration. In *Conference on Learning Theory*, pp. 84–118. PMLR, 2020.

Alimisis, F., Orvieto, A., Bécigneul, G., and Lucchi, A. A continuous-time perspective for modeling acceleration in Riemannian optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 1297–1307. PMLR, 2020.

Alimisis, F., Orvieto, A., Becigneul, G., and Lucchi, A. Momentum improves optimization on riemannian manifolds. In *International Conference on Artificial Intelligence and Statistics*, pp. 1351–1359. PMLR, 2021.

Bonnabel, S. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58 (9):2217–2229, 2013.

Calvo, M. and Oller, J. M. A distance between multivariate normal distributions based in an embedding into the Siegel group. *Journal of multivariate analysis*, 35(2): 223–242, 1990.

Calvo, M. and Oller, J. M. An explicit solution of information geodesic equations for the multivariate normal model. *Statistics & Risk Modeling*, 9(1-2):119–138, 1991.

Cherian, A. and Sra, S. Riemannian dictionary learning and sparse coding for positive definite matrices. *IEEE transactions on neural networks and learning systems*, 28 (12):2859–2871, 2016.

Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., and Sun, J. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13733–13742, 2021.

Dollár, P., Singh, M., and Girshick, R. Fast and accurate model scaling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 924–932, 2021.

Girolami, M. and Calderhead, B. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

Godaz, R., Ghojogh, B., Hosseini, R., Monsefi, R., Karray, F., and Crowley, M. Vector transport free riemannian lbfgs for optimization on symmetric positive definite matrix manifolds. In *Asian Conference on Machine Learning*, pp. 1–16. PMLR, 2021.

Grosse, R. and Martens, J. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pp. 573–582. PMLR, 2016.

Guillaumin, M., Verbeek, J., and Schmid, C. Is that you? metric learning approaches for face identification. In *2009 IEEE 12th international conference on computer vision*, pp. 498–505. IEEE, 2009.

Han, A., Mishra, B., Jawanpuria, P. K., and Gao, J. On riemannian optimization over positive definite matrices with the bures-wasserstein geometry. *Advances in Neural Information Processing Systems*, 34:8940–8953, 2021.

Han, D., Kim, J., and Kim, J. Deep pyramidal residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5927–5935, 2017.

Hosseini, R. and Sra, S. Matrix manifold optimization for Gaussian mixtures. In *Advances in Neural Information Processing Systems*, pp. 910–918, 2015.

Ivey, T. A. and Landsberg, J. M. *Cartan for beginners: differential geometry via moving frames and exterior differential systems*, volume 61. American Mathematical Society Providence, RI, 2003.

Jeuris, B., Vandebril, R., and Vandereycken, B. A survey and comparison of contemporary algorithms for computing the matrix geometric mean. *Electronic Transactions on Numerical Analysis*, 39(ARTICLE):379–402, 2012.

Khan, M. and Lin, W. Conjugate-computation variational inference: Converting variational inference in non-conjugate models to inferences in conjugate models. In *Artificial Intelligence and Statistics*, pp. 878–887, 2017.

Lezcano, C.-M. Trivializations for gradient-based optimization on manifolds. *Advances in Neural Information Processing Systems*, 32:9157–9168, 2019.

Lezcano, C.-M. Adaptive and momentum methods on manifolds through trivializations. *arXiv preprint arXiv:2010.04617*, 2020.

Lin, W., Nielsen, F., Emtiyaz, K. M., and Schmidt, M. Tractable structured natural-gradient descent using local parameterizations. In *International Conference on Machine Learning*, pp. 6680–6691. PMLR, 2021a.

Lin, W., Nielsen, F., Khan, M. E., and Schmidt, M. Structured second-order methods via natural gradient descent. *arXiv preprint arXiv:2107.10884*, 2021b.

Makam, V., Reichenbach, P., and Seigal, A. Symmetries in directed gaussian graphical models. *arXiv preprint arXiv:2108.10058*, 2021.

Martens, J. and Grosse, R. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pp. 2408–2417, 2015.

Minh, H. Q. and Murino, V. Covariances in computer vision and machine learning. *Synthesis Lectures on Computer Vision*, 7(4):1–170, 2017.

Pennec, X., Fillard, P., and Ayache, N. A Riemannian framework for tensor computing. *International Journal of computer vision*, 66(1):41–66, 2006.

Piuze, E., Sporring, J., and Siddiqi, K. Maurer-cartan forms for fields on surfaces: application to heart fiber geometry. *IEEE transactions on pattern analysis and machine intelligence*, 37(12):2492–2504, 2015.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10428–10436, 2020.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Slawski, M., Li, P., and Hein, M. Regularization-free estimation in trace regression with symmetric positive semidefinite matrices. *Advances in Neural Information Processing Systems*, 28, 2015.

Trockman, A. and Kolter, J. Z. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022.

Wang, C., Sun, D., and Toh, K.-C. Solving log-determinant optimization problems by a newton-cg primal proximal point algorithm. *SIAM Journal on Optimization*, 20(6): 2994–3013, 2010.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems*, 30, 2017.

Appendices

Outline of the Appendix:

- Appendix A contains more numerical results.
- Appendix B summarizes the normal coordinates used in this work.
- The rest of the appendix contains proofs of the claims and derivations for examples considered in the main text.

A. Additional Results

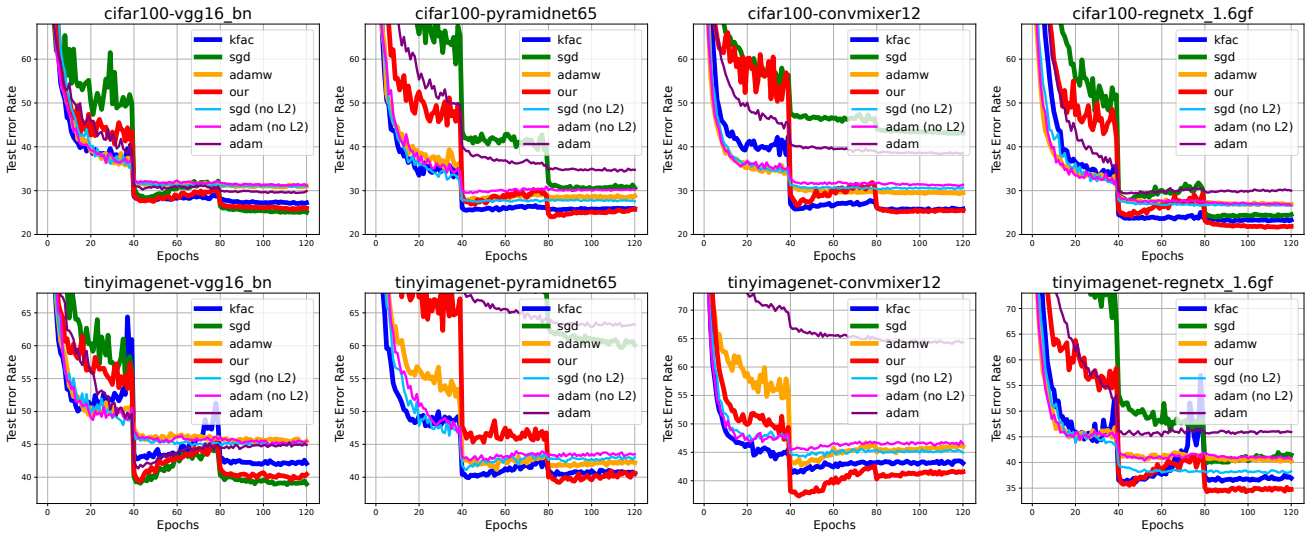


Figure 5. Performance of NN optimizers on more datasets. SGD performs best in the classical model and fairly in the modern models. Our updates achieve competitive test error rates compared to baselines and perform better than KFAC in many cases.

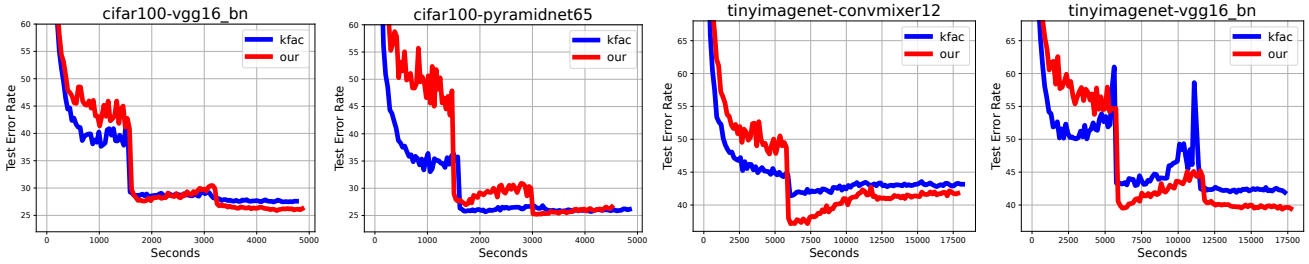


Figure 6. Additional results of our method and KFAC using a new random seed. We use these two methods to train NN models in 120 epochs. We report the performance of the methods in terms of running time.

Hyperparameter	Meaning	Our Method in Fig. 2	KFAC
β_2	Standard stepsize	Tuned	Tuned
α_2	Standard momentum weight	0.9	0.9
γ	(L2) weight decay	0.01	0.01
λ	Damping weight	0.01; 0.005; 0.0005	0.01; 0.005; 0.0005
T	Update frequency	10; 25; 60	10; 25; 60
θ	Moving average in KFAC	NA	0.95
β_1	Stepsize to update our preconditioner	0.01	NA
α_1	Momentum weight to update our preconditioner	0.5	NA

Table 5. Hyperparameter configuration in our update and KFAC. We first choose the damping weight λ based on the performance of KFAC and use the same value in our update. For both methods, we set $\lambda = 0.01, 0.0005, 0.005$ in VGG16, RepVGG-B1G4, and other models, respectively. To reduce the iteration cost of both methods, we update the preconditioner at every $T = 60, 25, 10$ iterations for RepVGG-B1G4, RegNetZ-500MF, and other models, respectively. The value of the hyperparameter θ is chosen as suggested at https://github.com/alecwangcq/KFAC-Pytorch. Since we do not use pre-training, we consider the first 500 iterations as a warm-up period to update our preconditioner by using a smaller stepsize β_1: we set $\beta_1 = 0.0002$ for the first 100 iterations, increase it to $\beta_1 = 0.002$ for the next 400 iterations, and finally fix it to $\beta_1 = 0.01$ for the remaining iterations.

Dataset	VGG-16-BN	PyramidNet-65	ConvMixer-12	RegNetX-1.6GF
CIFAR-100	14,774,436	707,428	911,204	8,368,436
TinyImageNet-200	14,825,736	733,128	936,904	8,459,736

Dataset	RegNetZ-500MF	RepVGG-B1G4
ImageNet-100	6,200,242	38,121,956

Table 6. Number of Learnable Parameters in the NN Models Considered

Dataset	Input Dimension	Number of Classes	Number of Training Points	Number of Test Points
CIFAR-100	32×32	100	50,000	10,000
TinyImageNet-200	64×64	200	100,000	5,000
ImageNet-100	224×224	100	130,000	5,000

Table 7. Statistics of the Datasets

B. Summary of Generalized Normal Coordinates

SPD (sub)manifold $\mathcal{M}$	Name	Our normal coordinate
$\{\boldsymbol{\tau} \in \mathbb{R}^{k\times k} \big\vert \boldsymbol{\tau} \in \mathcal{S}_{++}^{k\times k}\}$ with affine-invariant metric $\mathbf{F}$	Full manifold	See Eq. (14)
$\left\{\boldsymbol{\tau} = \begin{bmatrix} \mathbf{V} & \boldsymbol{\mu} \\ \boldsymbol{\mu}^T & 1 \end{bmatrix} \in \mathbb{R}^{k\times k} \Big\vert \boldsymbol{\tau} \in \mathcal{S}_{++}^{k\times k}\right\}$ with affine-invariant metric $\mathbf{F}$	Submanifold induced by Siegel embedding	See Eq. (15),(16)
$\left\{\boldsymbol{\tau} = \mathbf{U} \otimes \mathbf{W} \in \mathbb{R}^{pd\times pd} \Big\vert \mathbf{U} \in \mathcal{S}_{++}^{p\times p}, \mathbf{W} \in \mathcal{S}_{++}^{d\times d}\right\}$ with an approximated affine-invariant metric $\mathbf{F}$	Kronecker-product submanifold	See Eq. (23)

Table 8. Summary of our normal coordinates, where $\mathcal{S}_{++}$ denotes the set of SPD matrices

C. Background

In this section, we will assume $\boldsymbol{\tau}$ is a (learnable) vector to simplify notations. For a SPD matrix $\mathbf{M} \in \mathcal{S}_{++}^{k\times k}$, we could consider $\boldsymbol{\tau} = \text{vech}(\mathbf{M})$, where $\text{vech}(\mathbf{M})$ returns a $\frac{k(k+1)}{2}$-dimensional array obtained by vectorizing only the lower triangular part of $\mathbf{M}$, which is known as the half-vectorization function.

C.1. Fisher-Rao Metric

Under parametrization $\boldsymbol{\tau}$, the Fisher-Rao Metric is defined as

$$\mathbf{F}(\boldsymbol{\tau}^{(\text{cur})}) := -E_{q(z|\tau)}[\nabla_\tau^2 \log q(\mathbf{z}|\boldsymbol{\tau})]\big|_{\boldsymbol{\tau}=\boldsymbol{\tau}^{(\text{cur})}}, \tag{24}$$

where $q(\mathbf{z}|\boldsymbol{\tau})$ is a probabilistic distribution parameterized by $\boldsymbol{\tau}$, such as a Gaussian distribution with zero mean and covariance $\boldsymbol{\tau}$.

C.2. Christoffel Symbols

Given a Riemannian metric $\mathbf{F}$, the Christoffel symbols of the first kind are defined as

$$\Gamma_{d,ab}(\boldsymbol{\tau}_1) := \tfrac{1}{2}[\partial_a F_{bd}(\boldsymbol{\tau}) + \partial_b F_{ad}(\boldsymbol{\tau}) - \partial_d F_{ab}(\boldsymbol{\tau})]\Big|_{\tau=\tau_1}, \tag{25}$$

where $F_{bd}(\boldsymbol{\tau})$ denotes the (b,d) entry of the metric $\mathbf{F}_\tau$ and ∂_b denotes the partial derivative w.r.t. the b-th entry of $\boldsymbol{\tau}$.

The Christoffel symbols of the second kind are defined as

$$\Gamma^c_{ab}(\boldsymbol{\tau}) := \sum_d F^{cd}(\boldsymbol{\tau})\Gamma_{d,ab}(\boldsymbol{\tau}), \tag{26}$$

where $F^{cd}(\boldsymbol{\tau})$ denotes the (c,d) entry of the inverse $\mathbf{F}^{-1}$ of the metric. Observe that the Christoffel symbols of the second kind involve computing all partial derivatives of the metric $\mathbf{F}$ and the inverse of the metric.

C.3. Riemannian Exponential Map

The Riemannian exponential map is defined via a geodesic, which generalizes the notion of a straight line to a manifold. The geodesic $\mathbf{r}(t)$ satisfies the following second-order nonlinear system of ODEs with initial values $\mathbf{r}(0) = \mathbf{x}$ and $\dot{\mathbf{r}}(0) = \boldsymbol{\nu}$, where $\mathbf{x}$ denotes a point on the manifold and $\boldsymbol{\nu}$ is a Riemannian gradient,

$$\text{geodesic ODE:} \quad \ddot{r}^c(t) + \sum_{a,b} \Gamma^c_{ab}(\mathbf{r}(t))\dot{r}^a(t)\dot{r}^b(t) = 0, \tag{27}$$

where $r^c(t)$ denotes the c-th entry of $\mathbf{r}(t)$.

The Riemannian exponential map is defined as

$$\text{RExp}(\mathbf{x}, \boldsymbol{\nu}) := \mathbf{r}(1), \tag{28}$$

where $\mathbf{x}$ denotes an initial point and $\boldsymbol{\nu}$ is an initial Riemannian gradient so that $\mathbf{r}(0) = \mathbf{x}$ and $\dot{\mathbf{r}}(0) = \boldsymbol{\nu}$.

C.4. Riemannian (Parallel) Transport Map

In a curved space, the transport map along a given curve generalizes the notion of parallel transport. In Riemannian optimization, we consider the transport map along a geodesic curve. Given a geodesic curve $\mathbf{r}(t)$, denote by $\mathbf{v}(t)$ a smooth Riemannian gradient field that satisfies the following first-order linear system of ODEs with initial value $\mathbf{v}(0) = \boldsymbol{\nu}$,

$$\text{transport ODE:} \quad \dot{v}^c(t) + \sum_{a,b} \Gamma^c_{ab}(\mathbf{r}(t)) v^a(t) \dot{r}^b(t) = 0. \tag{29}$$

The transport map $\hat{T}_{\boldsymbol{\tau}^{(\text{cur})} \to \boldsymbol{\tau}^{(\text{new})}}(\boldsymbol{\nu})$ transports the Riemannian gradient $\boldsymbol{\nu}$ at $\boldsymbol{\tau}^{(\text{cur})}$ to $\boldsymbol{\tau}^{(\text{new})}$ as follows,

$$\hat{T}_{\boldsymbol{\tau}^{(\text{cur})} \to \boldsymbol{\tau}^{(\text{new})}}(\boldsymbol{\nu}) := \mathbf{v}(1), \tag{30}$$

where $\mathbf{r}(0) = \boldsymbol{\tau}^{(\text{cur})}$, $\mathbf{r}(1) = \boldsymbol{\tau}^{(\text{new})}$, and $\mathbf{v}(0) = \boldsymbol{\nu}$. It can be computationally challenging to solve this linear ODE due to the presence of the Christoffel symbols.

C.5. Euclidean (Parallel) Transport Map

Given a geodesic curve $\mathbf{r}(t)$, denote by $\boldsymbol{\omega}(t)$ a smooth Euclidean gradient field on manifold $\mathcal{M}$ that satisfies the following first-order linear system of ODEs with initial value $\boldsymbol{\omega}(0) = \mathbf{m}$,

$$\text{transport ODE:} \quad \dot{\omega}_c(t) - \sum_{a,b} \Gamma^a_{cb}(\mathbf{r}(t)) \omega_a(t) \dot{r}^b(t) = 0. \tag{31}$$

The transport map $T_{\boldsymbol{\tau}^{(\text{cur})} \to \boldsymbol{\tau}^{(\text{new})}}(\mathbf{g})$ transports the Euclidean gradient $\mathbf{g}$ at $\boldsymbol{\tau}^{(\text{cur})}$ to $\boldsymbol{\tau}^{(\text{new})}$ as shown below,

$$T_{\boldsymbol{\tau}^{(\text{cur})} \to \boldsymbol{\tau}^{(\text{new})}}(\mathbf{g}) := \boldsymbol{\omega}(1), \tag{32}$$

where $\mathbf{r}(0) = \boldsymbol{\tau}^{(\text{cur})}$, $\mathbf{r}(1) = \boldsymbol{\tau}^{(\text{new})}$, and $\boldsymbol{\omega}(0) = \mathbf{g}$.

Given a Euclidean gradient $\mathbf{g}$ evaluated at $\boldsymbol{\tau}^{(\text{cur})}$, the Riemannian and Euclidean transport maps are related as follows,

$$\hat{T}_{\boldsymbol{\tau}^{(\text{cur})} \to \boldsymbol{\tau}^{(\text{new})}}(\mathbf{F}_\tau^{-1}(\boldsymbol{\tau}^{(\text{cur})})\mathbf{g}) = \mathbf{F}_\tau^{-1}(\boldsymbol{\tau}^{(\text{new})}) T_{\boldsymbol{\tau}^{(\text{cur})} \to \boldsymbol{\tau}^{(\text{new})}}(\mathbf{g}). \tag{33}$$

C.6. Update 3 is Equivalent to Alimisis et al. (2020)

Note that $\mathbf{m}$ and $\mathbf{z}$ are initialized by zero. Due to Eq. (33), updates (2) and (3) are equivalent since $\mathbf{m}^{(\text{cur})} = \mathbf{F}(\boldsymbol{\tau}^{(\text{cur})})\boldsymbol{\nu}^{(\text{cur})}$ and $\mathbf{w}^{(\text{new})} = \mathbf{F}(\boldsymbol{\tau}^{(\text{new})})\mathbf{z}^{(\text{new})}$, where $\mathbf{m}^{(\text{cur})}$ and $\mathbf{w}^{(\text{new})}$ are defined in Eq. (3) while $\boldsymbol{\nu}^{(\text{cur})}$ and $\mathbf{z}^{(\text{new})}$ are defined in Eq. (2)

D. Simplification of the vector-Jacobian Product

The vector-Jacobian product in (13) could, in general, be computed by automatic differentiation. We give two cases for SPD (sub)manifolds where the product can be explicitly simplified. For notation simplicity, we denote $\mathbf{m} = \mathbf{m}^{(\boldsymbol{\eta}_0)}$ and $\mathbf{w} = \mathbf{w}^{(\boldsymbol{\eta}_1)}$. Thus $\mathbf{w} = \mathbf{m}$ when we use the approximation in Eq. (12).

D.1. Symmetric Cases

Suppose that $\boldsymbol{\eta}$ is symmetric, in which case $\mathbf{m}$ is also symmetric due to update (11). We further denote $\mathbf{A}_0 = \mathbf{A}^{(\text{cur})}$ and $\mathbf{A}_1 = \mathbf{A}^{(\text{new})}$, where $\mathbf{A}_1 = \mathbf{A}_0 \text{Expm}(-\frac{1}{2}\mathbf{m}^{(\boldsymbol{\eta}_0)}) = \mathbf{A}_0 \text{Expm}(-\frac{1}{2}\mathbf{m})$.

Recall that $\boldsymbol{\tau} = \mathbf{A}_0 \text{Expm}(\boldsymbol{\eta}) \mathbf{A}_0^T = \mathbf{A}_1 \text{Expm}(\boldsymbol{\xi}) \mathbf{A}_1^T$. Thus, we have

$$\text{Expm}(\boldsymbol{\eta}) = \text{Expm}(-\tfrac{1}{2}\mathbf{m}) \text{Expm}(\boldsymbol{\xi}) \text{Expm}^T(-\tfrac{1}{2}\mathbf{m}) = \text{Expm}(-\tfrac{1}{2}\mathbf{m}) \text{Expm}(\boldsymbol{\xi}) \text{Expm}(-\tfrac{1}{2}\mathbf{m}). \tag{34}$$

By the Baker–Campbell–Hausdorff formula, we have

$$\text{Expm}^{-1}(\text{Expm}(\mathbf{N})\text{Expm}(\mathbf{M})) = \mathbf{N} + \mathbf{M} + \sum_i w_i (\mathbf{M}^{a_i} \mathbf{N} \mathbf{M}^{b_i} - \mathbf{M}^{c_i} \mathbf{N} \mathbf{M}^{d_i}) + O(\mathbf{N}^2), \tag{35}$$

$$\text{Expm}^{-1}(\text{Expm}(\mathbf{M})\text{Expm}(\mathbf{N})) = \mathbf{N} + \mathbf{M} + \sum_i w_i (\mathbf{M}^{a_i} \mathbf{N} \mathbf{M}^{b_i} - \mathbf{M}^{c_i} \mathbf{N} \mathbf{M}^{d_i}) + O(\mathbf{N}^2), \tag{36}$$

where a_i, b_i, c_i, d_i are non-negative integers satisfying $a_i + b_i = c_i + d_i > 0$, and w_i is a coefficient.

Since we evaluate the Jacobian at $\boldsymbol{\xi}_0 = \mathbf{0}$, we can get rid of the higher-order term $O(\boldsymbol{\xi}^2)$, which leads to the following simplification,

$$\boldsymbol{\eta} = \mathrm{Expm}^{-1}(\mathrm{Expm}(-\tfrac{1}{2}\mathbf{m})\mathrm{Expm}(\boldsymbol{\xi})\mathrm{Expm}(-\tfrac{1}{2}\mathbf{m})) = \boldsymbol{\xi} - \mathbf{m} + \sum_i w_i(\mathbf{m}^{a_i}\boldsymbol{\xi}\mathbf{m}^{b_i} - \mathbf{m}^{c_i}\boldsymbol{\xi}\mathbf{m}^{d_i}) + O(\boldsymbol{\xi}^2). \tag{37}$$

Recall that $\mathbf{m} = \mathbf{w}$ is symmetric. The vector-Jacobian product can be simplified as

$$\begin{aligned}
\mathbf{w}^T\mathbf{J}(\boldsymbol{\xi}_0) &= \mathbf{m}^T\big[\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\xi}}\big|_{\xi=\xi_0}\big] \\
&= \nabla_\xi \mathrm{Tr}(\mathbf{m}^T\boldsymbol{\eta})\big|_{\xi=\xi_0} \\
&= \nabla_\xi \mathrm{Tr}\big(\mathbf{m}^T[\boldsymbol{\xi} - \mathbf{m} + \sum_i w_i(\mathbf{m}^{a_i}\boldsymbol{\xi}\mathbf{m}^{b_i} - \mathbf{m}^{c_i}\boldsymbol{\xi}\mathbf{m}^{d_i}) + O(\boldsymbol{\xi}^2)]\big)\big|_{\xi=\xi_0} \\
&= \nabla_\xi \mathrm{Tr}\big(\mathbf{m}^T[\boldsymbol{\xi} + \sum_i w_i(\mathbf{m}^{a_i}\boldsymbol{\xi}\mathbf{m}^{b_i} - \mathbf{m}^{c_i}\boldsymbol{\xi}\mathbf{m}^{d_i})]\big)\big|_{\xi=\xi_0} \\
&= \nabla_\xi \mathrm{Tr}\big(\mathbf{m}[\boldsymbol{\xi} + \sum_i w_i(\mathbf{m}^{a_i}\boldsymbol{\xi}\mathbf{m}^{b_i} - \mathbf{m}^{c_i}\boldsymbol{\xi}\mathbf{m}^{d_i})]\big)\big|_{\xi=\xi_0} \\
&= \nabla_\xi \mathrm{Tr}\big(\mathbf{m}\boldsymbol{\xi} + \sum_i w_i(\mathbf{m}^{a_i+b_i+1}\boldsymbol{\xi} - \mathbf{m}^{c_i+d_i+1}\boldsymbol{\xi})\big)\big|_{\xi=\xi_0} \\
&= \nabla_\xi \mathrm{Tr}(\mathbf{m}\boldsymbol{\xi})\big|_{\xi=\xi_0} \\
&= \mathbf{m} = \mathbf{w}.
\end{aligned} \tag{38}$$

D.2. Triangular Cases

Without loss of generality, we assume $\boldsymbol{\eta}$ is lower-triangular, in which case $\mathbf{m}$ is also lower-triangular due to update (11). Similarly, we denote $\mathbf{A}_0 = \mathbf{A}^{(\mathrm{cur})}$ and $\mathbf{A}_1 = \mathbf{A}^{(\mathrm{new})}$, where $\mathbf{A}_1 = \mathbf{A}_0\mathrm{Expm}(-\mathbf{D}\odot\mathbf{m}^{(\boldsymbol{\eta}_0)}) = \mathbf{A}_0\mathrm{Expm}(-\mathbf{D}\odot\mathbf{m})$, and where $\mathbf{D} = \frac{1}{\sqrt{2}}\mathrm{Tril}(\mathbf{1}) + (\frac{1}{2} - \frac{1}{\sqrt{2}}\mathbf{1})\mathbf{I}$ is chosen so that the metric is orthonormal at $\boldsymbol{\eta}_0$, and $\mathrm{Tril}(\mathbf{1})$ denotes a lower-triangular matrix of ones.

Recall that $\boldsymbol{\tau} = \mathbf{A}_0\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}^T(\mathbf{D}\odot\boldsymbol{\eta})\mathbf{A}_0^T = \mathbf{A}_1\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\xi})\mathrm{Expm}^T(\mathbf{D}\odot\boldsymbol{\xi})\mathbf{A}_1^T$. Thus, we have

$$\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}^T(\mathbf{D}\odot\boldsymbol{\eta}) = \mathrm{Expm}(-\mathbf{D}\odot\mathbf{m})\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\xi})\mathrm{Expm}^T(\mathbf{D}\odot\boldsymbol{\xi})\mathrm{Expm}^T(-\mathbf{D}\odot\mathbf{m}). \tag{39}$$

Since $\boldsymbol{\eta}$, $\mathbf{m}$ and $\boldsymbol{\xi}$ are lower-triangular, $\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\eta})$, $\mathrm{Expm}(-\mathbf{D}\odot\mathbf{m})$, and $\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\xi})$ are also lower-triangular. Moreover, all the eigenvalues of $\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\eta})$, $\mathrm{Expm}(-\mathbf{D}\odot\mathbf{m})$, and $\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\xi})$ are positive.

Note that we make use of the uniqueness of the Cholesky decomposition since $\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\eta})$ can be viewed as a Cholesky factor. Thus, $\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\eta}) = \mathrm{Expm}(-\mathbf{D}\odot\mathbf{m})\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\xi})$.

By the Baker–Campbell–Hausdorff formula, we have

$$\mathrm{Expm}^{-1}\big(\mathrm{Expm}(\mathbf{M})\mathrm{Expm}(\mathbf{N})\big) = \mathbf{M} + \mathbf{N} + \tfrac{1}{2}\langle\mathbf{M},\mathbf{N}\rangle + O(\langle\mathbf{M},\langle\mathbf{M},\mathbf{N}\rangle\rangle) + O(\mathbf{N}^2), \tag{40}$$

where $\langle\mathbf{M},\mathbf{N}\rangle = \mathbf{M}\mathbf{N} - \mathbf{N}\mathbf{M}$ is the Lie bracket. Thus, we have

$$\boldsymbol{\eta} = \big(\mathbf{D}\odot\boldsymbol{\xi} - \mathbf{D}\odot\mathbf{m} + \tfrac{1}{2}\langle-\mathbf{D}\odot\mathbf{m}, \mathbf{D}\odot\boldsymbol{\xi}\rangle\big) \oslash \mathbf{D} + O(\langle\mathbf{m},\langle\mathbf{m},\boldsymbol{\xi}\rangle\rangle + O(\boldsymbol{\xi}^2), \tag{41}$$

where $\oslash$ denotes elementwise division.

We get rid of the higher-order term $O(\boldsymbol{\xi}^2)$ by evaluating $\boldsymbol{\xi} = \boldsymbol{\xi}_0 = \mathbf{0}$.

Recall that $\mathbf{m} = \mathbf{w}$. The vector-Jacobian product can be simplified as

$$\begin{aligned}
\mathbf{w}^T\mathbf{J}(\boldsymbol{\xi}_0) &= \mathbf{m}^T\big[\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\xi}}\big|_{\xi=\xi_0}\big] \\
&= \nabla_\xi \mathrm{Tr}(\mathbf{m}^T\boldsymbol{\eta})\big|_{\xi=\xi_0} \\
&= \nabla_\xi \mathrm{Tr}(\mathbf{m}^T(\mathbf{D}\odot\boldsymbol{\xi} - \mathbf{D}\odot\mathbf{m} + \tfrac{1}{2}\langle -\mathbf{D}\odot\mathbf{m}, \mathbf{D}\odot\boldsymbol{\xi}\rangle \oslash \mathbf{D}) + O(\langle \mathbf{m}, \langle \mathbf{m}, \boldsymbol{\xi}\rangle\rangle))\big|_{\xi=\xi_0} \\
&= \nabla_\xi \mathrm{Tr}((\mathbf{m}\oslash\mathbf{D})^T(\mathbf{D}\odot\boldsymbol{\xi} - \mathbf{D}\odot\mathbf{m} + \tfrac{1}{2}\langle -\mathbf{D}\odot\mathbf{m}, \mathbf{D}\odot\boldsymbol{\xi}\rangle + O(\langle \mathbf{m}, \langle \mathbf{m}, \boldsymbol{\xi}\rangle\rangle)))\big|_{\xi=\xi_0} \\
&= \underbrace{\mathbf{m}}_{O(\beta)} + \tfrac{1}{2}\mathbf{D}\odot\underbrace{\big\langle(-\mathbf{D}\odot\mathbf{m})^T, \mathbf{m}\oslash\mathbf{D}\big\rangle}_{O(\beta^2)} + \underbrace{O\left(\big\langle\mathbf{m}, \big\langle\mathbf{m}, \mathbf{m}^T\big\rangle\big\rangle\right)}_{O(\beta^3)}.
\end{aligned} \tag{42}$$

E. Simplification of the Metric Calculation at $\boldsymbol{\eta}_0$

Consider $\boldsymbol{\tau} = \boldsymbol{\phi}_{\tau^{(\mathrm{cur})}}(\boldsymbol{\eta}) = \mathbf{A}\mathbf{A}^T \in \mathcal{S}_{++}^{k\times k}$, where $\mathbf{A} = \mathbf{A}^{(\mathrm{cur})}\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\eta})$.

For notation simplicity, we let $\mathbf{A}_0 = \mathbf{A}^{(\mathrm{cur})}$ and $\boldsymbol{\tau}_0 = \boldsymbol{\tau}^{(\mathrm{cur})} = \mathbf{A}_0\mathbf{A}_0^T$. Let $\tilde{\boldsymbol{\eta}}$ denote the vector representation of the learnable part of $\boldsymbol{\eta}$. By definition of the affine-invariant metric, we have

$$\begin{aligned}
\mathbf{F}(\boldsymbol{\eta}_0) &= -2\mathbb{E}_{\mathcal{N}(\mathbf{0},\boldsymbol{\tau})}\big[\nabla^2_{\tilde{\eta}}\log\mathcal{N}(\mathbf{0},\boldsymbol{\tau})\big]\big|_{\eta=\eta_0} \\
&= \mathbb{E}_{\mathcal{N}(x|\mathbf{0},\boldsymbol{\tau})}\big[\nabla^2_{\tilde{\eta}}\big\{\mathrm{Tr}[\mathbf{x}\mathbf{x}^T\mathbf{A}_0^{-T}\mathrm{Expm}^T(-\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})\mathbf{A}_0^{-1}] + 2\log\det\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\eta})\big\}\big]\big|_{\eta=\eta_0} \\
&= \nabla^2_{\tilde{\eta}}\big\{\mathrm{Tr}[\mathbb{E}_{\mathcal{N}(x|\mathbf{0},\boldsymbol{\tau}_0)}\big[\mathbf{x}\mathbf{x}^T\big]\mathbf{A}_0^{-T}\mathrm{Expm}^T(-\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})\mathbf{A}_0^{-1}] + 2\log\det\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\eta})\big\}\big|_{\eta=\eta_0} \\
&= \nabla^2_{\tilde{\eta}}\big\{\mathrm{Tr}[[\mathbf{A}_0\mathbf{A}_0^T]\mathbf{A}_0^{-T}\mathrm{Expm}^T(-\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})\mathbf{A}_0^{-1}] + 2\log\det\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\eta})\big\}\big|_{\eta=\eta_0} \\
&= \nabla^2_{\tilde{\eta}}\big\{\mathrm{Tr}[\mathrm{Expm}^T(-\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})] + 2\log\det\mathrm{Expm}(\mathbf{D}\odot\boldsymbol{\eta})\big\}\big|_{\eta=\eta_0} \\
&= \nabla^2_{\tilde{\eta}}\big\{\mathrm{Tr}[\mathrm{Expm}^T(-\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})] + 2\mathrm{Tr}[\mathbf{D}\odot\boldsymbol{\eta}]\big\}\big|_{\eta=\eta_0} \text{ (ignore linear terms for a 2nd order derivative)} \\
&= \nabla^2_{\tilde{\eta}}\big\{\mathrm{Tr}[\mathrm{Expm}^T(-\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})]\big\}\big|_{\eta=\eta_0}.
\end{aligned} \tag{43}$$

Note that we express $\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})$ as $\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta}) = \mathbf{I} - \mathbf{D}\odot\boldsymbol{\eta} + \frac{1}{2}(\mathbf{D}\odot\boldsymbol{\eta})^2 + O(\boldsymbol{\eta}^3)$. Since we evaluate the metric at $\boldsymbol{\eta}_0 = \mathbf{0}$, we can get rid of the higher-order term $O(\boldsymbol{\eta}^3)$, which leads to the following simplification,

$$\begin{aligned}
&\nabla_{\eta_{ij}}\mathrm{Tr}[\mathrm{Expm}^T(-\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})] \\
&\quad= 2\mathrm{Tr}[\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})[\nabla_{\eta_{ij}}\mathrm{Expm}^T(-\mathbf{D}\odot\boldsymbol{\eta})]] \\
&\quad= 2D_{ij}\nabla_{\eta_{ij}}\big\{\mathrm{Tr}[\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})[-\mathbf{E}_{ij} + \tfrac{1}{2}\mathbf{E}_{ij}(\mathbf{D}\odot\boldsymbol{\eta}) + \tfrac{1}{2}(\mathbf{D}\odot\boldsymbol{\eta})\mathbf{E}_{ij} + O(\boldsymbol{\eta}^2)]^T]\big\}.
\end{aligned} \tag{44}$$

Thus, we have

$$\begin{aligned}
&\nabla_\eta\mathrm{Tr}[\mathrm{Expm}^T(-\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})] \\
&\quad= \mathbf{D}\odot[-2\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta}) + \mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})(\mathbf{D}\odot\boldsymbol{\eta})^T + (\mathbf{D}\odot\boldsymbol{\eta})^T\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})] + O(\boldsymbol{\eta}^2).
\end{aligned} \tag{45}$$

To show $\mathbf{F}(\boldsymbol{\eta}_0) = \mathbf{I}$, we show that $\mathbf{F}(\boldsymbol{\eta}_0)\mathbf{v} = \mathbf{v}$ for any $\mathbf{v}$. Let $\mathbf{V}$ be the matrix representation of $\mathbf{v}$, which has the same structure as $\boldsymbol{\eta}$, such as being symmetric or being lower-triangular. Then,

$$\begin{aligned}
&\mathbf{F}(\boldsymbol{\eta}_0)\mathbf{v} \\
&= \nabla^2_{\tilde{\eta}}\big\{\mathrm{Tr}[\mathrm{Expm}^T(-\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})]\big\}\big|_{\eta=\eta_0}\mathbf{v} \\
&= \nabla_{\tilde{\eta}}\big\{\mathbf{v}^T\nabla_{\tilde{\boldsymbol{\eta}}}\mathrm{Tr}[\mathrm{Expm}^T(-\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})]\big\}\big|_{\eta=\eta_0}, \text{ (note: } \tilde{\eta}, \mathbf{v} \text{ are vectors)} \\
&= \nabla_{\tilde{\eta}}\mathrm{Tr}\big\{\mathbf{V}^T\nabla_\eta\mathrm{Tr}[\mathrm{Expm}^T(-\mathbf{D}\odot\boldsymbol{\eta})\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})]\big\}\big|_{\eta=\eta_0} \text{ (note: } \boldsymbol{\eta}, \mathbf{V} \text{ are matrices)} \\
&= \nabla_{\tilde{\eta}}\mathrm{Tr}\big\{\mathbf{V}^T(\mathbf{D}\odot[-2\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta}) + \mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})(\mathbf{D}\odot\boldsymbol{\eta})^T + (\mathbf{D}\odot\boldsymbol{\eta})^T\mathrm{Expm}(-\mathbf{D}\odot\boldsymbol{\eta})] + O(\boldsymbol{\eta}^2))\big\}\big|_{\eta=\eta_0}.
\end{aligned}$$

We can get rid of the higher-order term $O(\boldsymbol{\eta}^2)$ by evaluating at $\boldsymbol{\eta} = \boldsymbol{\eta}_0 = \mathbf{0}$ and noting that

$$\mathrm{Tr}(\mathbf{A}^T(\mathbf{D} \odot \mathbf{B})) = \sum(\mathbf{A} \odot (\mathbf{D} \odot \mathbf{B})) = \mathrm{Tr}((\mathbf{D} \odot \mathbf{A})^T\mathbf{B}). \tag{46}$$

Also note that

$$\begin{aligned}&\nabla_{\eta_{ij}}\mathrm{Tr}\big\{(\mathbf{D} \odot \mathbf{V})^T[-2\mathrm{Expm}(-\mathbf{D} \odot \boldsymbol{\eta}) + \mathrm{Expm}(-\mathbf{D} \odot \boldsymbol{\eta})(\mathbf{D} \odot \boldsymbol{\eta})^T + (\mathbf{D} \odot \boldsymbol{\eta})^T\mathrm{Expm}(-\mathbf{D} \odot \boldsymbol{\eta})]\big\}\big|_{\eta=\eta_0}\\ &\quad = \mathrm{Tr}\big\{(\mathbf{D} \odot \mathbf{V})^T 2D_{ij}[\mathbf{E}_{ij} + \mathbf{E}_{ij}^T]\big\}\big|_{\eta=\eta_0}.\end{aligned} \tag{47}$$

Thus, we have

$$\begin{aligned}&\nabla_{\eta}\mathrm{Tr}\big\{(\mathbf{D} \odot \mathbf{V})^T[-2\mathrm{Expm}(-\mathbf{D} \odot \boldsymbol{\eta}) + \mathrm{Expm}(-\mathbf{D} \odot \boldsymbol{\eta})(\mathbf{D} \odot \boldsymbol{\eta})^T + (\mathbf{D} \odot \boldsymbol{\eta})^T\mathrm{Expm}(-\mathbf{D} \odot \boldsymbol{\eta})]\big\}\big|_{\eta=\eta_0}\\ &\quad = 2\mathbf{D} \odot (\mathbf{D} \odot \mathbf{V} + (\mathbf{D} \odot \mathbf{V})^T).\end{aligned} \tag{48}$$

E.1. Symmetric Cases

When $\boldsymbol{\eta}$ is symmetric, $\mathbf{V}$ is also symmetric so

$$\begin{aligned}&\mathbf{F}(\boldsymbol{\eta}_0)\mathbf{v}\\ &= \nabla_{\tilde{\eta}}\mathrm{Tr}\big\{\mathbf{V}^T(\mathbf{D} \odot [-2\mathrm{Expm}(-\mathbf{D} \odot \boldsymbol{\eta}) + \mathrm{Expm}(-\mathbf{D} \odot \boldsymbol{\eta})(\mathbf{D} \odot \boldsymbol{\eta})^T + (\mathbf{D} \odot \boldsymbol{\eta})^T\mathrm{Expm}(-\mathbf{D} \odot \boldsymbol{\eta})] + O(\boldsymbol{\eta}^2))\big\}\big|_{\eta=\eta_0}\\ &= \mathrm{vech}(2\mathbf{D} \odot (\mathbf{D} \odot \mathbf{V} + \mathbf{D} \odot \mathbf{V})) = 4\mathrm{vech}(\mathbf{D}^2 \odot \mathbf{V}).\end{aligned} \tag{49}$$

When $\mathbf{D} = \frac{1}{2}\mathbf{1}$, we have $4\mathrm{vech}(\mathbf{D}^2 \odot \mathbf{V}) = \mathrm{vech}(\mathbf{V}) = \mathbf{v}$, where $\mathbf{1}$ is a matrix of ones. Thus, $\mathbf{F}(\boldsymbol{\eta}_0) = \mathbf{I}$.

E.2. Triangular Cases

Without loss of generality, we assume that $\boldsymbol{\eta}$ is lower-triangular, in which case $\mathbf{V}$ is lower-triangular, and thus

$$\begin{aligned}&\mathbf{F}(\boldsymbol{\eta}_0)\mathbf{v}\\ &= \nabla_{\tilde{\eta}}\mathrm{Tr}\big\{\mathbf{V}^T(\mathbf{D} \odot [-2\mathrm{Expm}(-\mathbf{D} \odot \boldsymbol{\eta}) + \mathrm{Expm}(-\mathbf{D} \odot \boldsymbol{\eta})(\mathbf{D} \odot \boldsymbol{\eta})^T + (\mathbf{D} \odot \boldsymbol{\eta})^T\mathrm{Expm}(-\mathbf{D} \odot \boldsymbol{\eta})] + O(\boldsymbol{\eta}^2))\big\}\big|_{\eta=\eta_0}\\ &= \mathrm{tril}(2\mathbf{D} \odot (\mathbf{D} \odot \mathbf{V} + \mathbf{D} \odot \mathbf{V}^T) \ \ (\mathbf{V}^T \text{ is upper-triangular. Thus } \mathrm{tril}(\mathbf{V}^T) = \mathrm{Diag}(\mathbf{V}^T) = \mathrm{Diag}(\mathbf{V}))\\ &= \mathrm{tril}(2\mathbf{D}^2 \odot (\mathbf{V} + \mathrm{Diag}(\mathbf{V}))),\end{aligned} \tag{50}$$

where $\mathrm{tril}(\cdot)$ represents a vector representation of the learnable part of a lower-triangular matrix and $\mathrm{Tril}(\cdot)$ denotes a lower-triangular matrix.

When $\mathbf{D} = \frac{1}{\sqrt{2}}\mathrm{Tril}(\mathbf{1}) + (\frac{1}{2} - \frac{1}{\sqrt{2}}\mathbf{I})$, we have $\mathrm{tril}(2\mathbf{D}^2 \odot (\mathbf{V} + \mathrm{Diag}(\mathbf{V}))) = \mathrm{tril}(\mathbf{V}) = \mathbf{v}$, where $\mathrm{Tril}(\mathbf{1})$ is a lower-triangular matrix of ones. Thus, $\mathbf{F}(\boldsymbol{\eta}_0) = \mathbf{I}$.

F. An Accurate Approximation of the Euclidean Transport Map

We consider the first-order approximation of the Euclidean transport

$$\mathbf{m}^{(\eta_1)} = T_{\eta_0 \to \eta_1}(\mathbf{m}^{(\eta_0)}) = \boldsymbol{\omega}(1) \approx \underbrace{\boldsymbol{\omega}(0)}_{\mathbf{m}^{(\eta_0)}} + \dot{\boldsymbol{\omega}}(0), \tag{51}$$

where we have to evaluate the Christoffel symbols as discussed below.

By the transport ODE in Eq. (31), we can compute $\dot{\boldsymbol{\omega}}(0)$ via

$$\dot{\omega}_c(0) - \sum_{a,b}\Gamma^a_{cb}(\mathbf{r}(0))\omega_a(0)\dot{r}^b(0) = 0, \tag{52}$$

where $\mathbf{r}(0) = \boldsymbol{\eta}_0$ is the current point and $\dot{\mathbf{r}}(0)$ is the Riemannian gradient so that $\boldsymbol{\eta}_1 = \mathrm{RExp}(\boldsymbol{\eta}_0, \dot{\mathbf{r}}(0))$. In our case, as shown in Eq. (11), we have that $\dot{\mathbf{r}}(0) = -\mathbf{F}^{-1}(\eta_0)\mathbf{m}^{(\eta_0)} = \mathbf{m}^{(\eta_0)}$ and $\boldsymbol{\omega}(0) = \mathbf{m}^{(\eta_0)}$.

Note that the metric and Christoffel symbols are evaluated at $\boldsymbol{\eta}_0 = \mathbf{0}$. The computation can be simplified due to the orthonormal metric as $\mathbf{F}^{-1}(\boldsymbol{\eta}_0) = \mathbf{I}$ and

$$\Gamma^a_{cb}(\boldsymbol{\eta}_0) = \sum_d F^{ad}(\boldsymbol{\eta}_0)\Gamma_{d,\,cb}(\boldsymbol{\eta}_0) = \sum_d \delta^{ad}\tfrac{1}{2}[\partial_c F_{bd}(\boldsymbol{\eta}_0) + \partial_b F_{cd}(\boldsymbol{\eta}_0) - \partial_d F_{cb}(\boldsymbol{\eta}_0)], \tag{53}$$

where $F^{ad}(\boldsymbol{\eta}_0) = \delta^{ad}$. Thus, we have the following simplification,

$$\begin{aligned}\dot{\omega}_c(0) &= -\tfrac{1}{2}\sum_{b,d}[\partial_c F_{bd}(\boldsymbol{\eta}_0) + \partial_b F_{cd}(\boldsymbol{\eta}_0) - \partial_d F_{cb}(\boldsymbol{\eta}_0)](\mathbf{m}^{(\eta_0)})^d(\mathbf{m}^{(\eta_0)})^b \\ &= -\tfrac{1}{2}\sum_{b,d}\partial_c F_{bd}(\boldsymbol{\eta}_0)(\mathbf{m}^{(\eta_0)})^d(\mathbf{m}^{(\eta_0)})^b. \end{aligned}\tag{54}$$

For notation simplicity, we let $\mathbf{m} = \mathbf{m}^{(\eta_0)}$ and $\mathbf{A}_0 = \mathbf{A}^{(\text{cur})}$.

For normal coordinate $\mathbf{A} = \mathbf{A}_0\text{Expm}(\mathbf{D}\odot\boldsymbol{\eta})$, we can obtain the following result. The calculation is similar to the metric calculation in Appx. E.

$$\dot{\boldsymbol{\omega}}(0) = \underbrace{\mathbf{D}\odot\left(\left\langle \mathbf{D}\odot\mathbf{m}, (\mathbf{D}\odot\mathbf{m})^T\right\rangle\right)}_{O(\beta^2)}, \tag{55}$$

where $\langle\mathbf{N},\mathbf{M}\rangle := \mathbf{NM} - \mathbf{MN}$ is the Lie bracket, the β is the stepsize used in Eq. (11).

When $\boldsymbol{\eta}$ is symmetric, we know that $\mathbf{m}$ is symmetric. Thus, we have $\dot{\boldsymbol{\omega}}(0) = \mathbf{0}$.

G. Structured NGD as a Special Case

G.1. Normal Coordinate for Structured NGD

We can obtain coordinate (16) from coordinate (15).

In Eq. (15), the normal coordinate is defined as $\mathbf{A} = \mathbf{A}_0\text{Expm}\left(\begin{bmatrix}\frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0\end{bmatrix}\right)$, where we use $\mathbf{A}_0$ to denote $\mathbf{A}^{(\text{cur})}$.

Note that

$$\text{Expm}\left(\begin{bmatrix}\frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0\end{bmatrix}\right) = \begin{bmatrix}\text{Expm}(\frac{1}{2}\boldsymbol{\eta}_L) & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu + O(\boldsymbol{\eta}_L\boldsymbol{\eta}_\mu) \\ \mathbf{0} & 1\end{bmatrix}. \tag{56}$$

The main point is that $O(\boldsymbol{\eta}_L\boldsymbol{\eta}_\mu)$ vanishes in the metric computation since we evaluate the metric at $\boldsymbol{\eta}_0 = \{\boldsymbol{\eta}_L, \boldsymbol{\eta}_\mu\} = \mathbf{0}$. Thus, we can ignore $O(\boldsymbol{\eta}_L\boldsymbol{\eta}_\mu)$, and recover Eq. (16):

$$\mathbf{A} = \mathbf{A}_0\begin{bmatrix}\text{Expm}(\frac{1}{2}\boldsymbol{\eta}_L) & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 1\end{bmatrix} = \begin{bmatrix}\mathbf{L}_0 & \boldsymbol{\mu}_0 \\ \mathbf{0} & 1\end{bmatrix}\begin{bmatrix}\text{Expm}(\frac{1}{2}\boldsymbol{\eta}_L) & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 1\end{bmatrix} = \begin{bmatrix}\mathbf{L}_0\text{Expm}(\frac{1}{2}\boldsymbol{\eta}_L) & \boldsymbol{\mu}_0 + \frac{1}{\sqrt{2}}\mathbf{L}_0\boldsymbol{\eta}_\mu \\ \mathbf{0} & 1\end{bmatrix}. \tag{57}$$

To show that $O(\boldsymbol{\eta}_L\boldsymbol{\eta}_\mu)$ vanishes in the metric computation, we have to show that all the cross terms between $\boldsymbol{\eta}_L$ and $\boldsymbol{\eta}_\mu$ of the metric vanish. Using Eq. (43),

$$\begin{aligned}&\mathbb{E}_{\mathcal{N}(\mathbf{0},\boldsymbol{\tau})}\left[\nabla_{\boldsymbol{\eta}_{L_{jk}}}\nabla_{\boldsymbol{\eta}_{\mu_i}}\log\mathcal{N}(\mathbf{0},\boldsymbol{\tau})\right]\big|_{\eta=\eta_0} \\ &\quad= \nabla_{\boldsymbol{\eta}_{L_{jk}}}\nabla_{\boldsymbol{\eta}_{\mu_i}}\left\{\text{Tr}\left[\text{Expm}^T\left(-\begin{bmatrix}\frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0\end{bmatrix}\right)\text{Expm}\left(-\begin{bmatrix}\frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0\end{bmatrix}\right)\right]\right\}\big|_{\eta=\eta_0}. \end{aligned}\tag{58}$$

We can drop higher order terms since we evaluate at $\boldsymbol{\eta}_0 = \{\boldsymbol{\eta}_L, \boldsymbol{\eta}_\mu\} = \mathbf{0}$. We get

$$
\begin{aligned}
&\nabla_{\boldsymbol{\eta}_{L_{jk}}} \nabla_{\boldsymbol{\eta}_{\mu_i}} \left\{ \operatorname{Tr}\left[\operatorname{Expm}^T\left(-\begin{bmatrix} \frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0 \end{bmatrix} \right) \operatorname{Expm}\left(-\begin{bmatrix} \frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0 \end{bmatrix} \right) \right] \right\} \\
=& 2\nabla_{\boldsymbol{\eta}_{L_{jk}}} \left\{ \operatorname{Tr}\left[\left\{ \nabla_{\boldsymbol{\eta}_{\mu_i}} \operatorname{Expm}^T\left(-\begin{bmatrix} \frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0 \end{bmatrix} \right) \right\} \operatorname{Expm}\left(-\begin{bmatrix} \frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0 \end{bmatrix} \right) \right] \right\} \\
=& 2\nabla_{\boldsymbol{\eta}_{L_{jk}}} \operatorname{Tr}\left[\left(-\begin{bmatrix} \mathbf{0} & \frac{1}{\sqrt{2}}\mathbf{e}_i \\ \mathbf{0} & 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \mathbf{0} & \frac{1}{\sqrt{2}}\mathbf{e}_i \\ \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \frac{1}{\sqrt{2}}\mathbf{e}_i \\ \mathbf{0} & 0 \end{bmatrix} \right)^T \operatorname{Expm}\left(-\begin{bmatrix} \frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0 \end{bmatrix} \right) \right] \\
=& 2\nabla_{\boldsymbol{\eta}_{L_{jk}}} \operatorname{Tr}\left[\left(-\begin{bmatrix} \mathbf{0} & \frac{1}{\sqrt{2}}\mathbf{e}_i \\ \mathbf{0} & 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \frac{1}{\sqrt{2}}\mathbf{e}_i \\ \mathbf{0} & 0 \end{bmatrix} \right)^T \operatorname{Expm}\left(-\begin{bmatrix} \frac{1}{2}\boldsymbol{\eta}_L & \frac{1}{\sqrt{2}}\boldsymbol{\eta}_\mu \\ \mathbf{0} & 0 \end{bmatrix} \right) \right] \\
=& 2\operatorname{Tr}\left[\begin{bmatrix} \mathbf{0} & \frac{1}{\sqrt{2}}\mathbf{e}_i \\ \mathbf{0} & 0 \end{bmatrix}^T \begin{bmatrix} \frac{1}{2}\mathbf{E}_{jk} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} + \frac{1}{2}\left(\begin{bmatrix} \frac{1}{2}\mathbf{E}_{jk} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \frac{1}{\sqrt{2}}\mathbf{e}_i \\ \mathbf{0} & 0 \end{bmatrix} \right)^T \right] = 0,
\end{aligned} \tag{59}
$$

and therefore

$$
\mathbb{E}_{\mathcal{N}(\mathbf{0},\boldsymbol{\tau})}\left[\nabla_{\boldsymbol{\eta}_{L_{jk}}} \nabla_{\boldsymbol{\eta}_{\mu_i}} \log \mathcal{N}(\mathbf{0}, \boldsymbol{\tau}) \right]\big|_{\eta=\eta_0} = 0. \tag{60}
$$

G.2. Gaussian Identities in Structured NGD

Recall that the manifold is defined as

$$
\mathcal{M} = \left\{ \boldsymbol{\tau} = \begin{bmatrix} \mathbf{V} & \boldsymbol{\mu} \\ \boldsymbol{\mu}^T & 1 \end{bmatrix} \in \mathbb{R}^{(d+1)\times(d+1)} \mid \boldsymbol{\tau} \succ 0 \right\}.
$$

To use Gaussian gradient identities, we first change the notation from $\boldsymbol{\mu}$ to $\mathbf{m}$ to avoid confusion:

$$
\mathcal{M} = \left\{ \boldsymbol{\tau} = \begin{bmatrix} \mathbf{V} & \mathbf{m} \\ \mathbf{m}^T & 1 \end{bmatrix} \in \mathbb{R}^{(d+1)\times(d+1)} \mid \boldsymbol{\tau} \succ 0 \right\}.
$$

In Sec. 3.3.1, we can compute the Euclidean gradient

$$
\mathbf{g}(\boldsymbol{\eta}_0) = \{\mathbf{L}^T\mathbf{g}_1\mathbf{L},\ \sqrt{2}\mathbf{L}^T(\mathbf{g}_1\mathbf{m} + \mathbf{g}_2)\}, \tag{61}
$$

where $\mathbf{g}(\boldsymbol{\tau}^{(\text{cur})}) = \begin{bmatrix} \mathbf{g}_1 & \mathbf{g}_2 \\ \mathbf{g}_2^T & 0 \end{bmatrix}$ is a (symmetric) Euclidean gradient w.r.t. $\boldsymbol{\tau} \in \mathcal{M}$.

By the chain rule, we have

$$
\frac{\partial \ell}{\partial m_i} = \operatorname{Tr}\Bigg(\underbrace{\left(\frac{\partial \ell}{\partial \boldsymbol{\tau}} \right)^T}_{\mathbf{g}^T(\boldsymbol{\tau})} \frac{\partial \boldsymbol{\tau}}{\partial m_i} \Bigg) = \operatorname{Tr}\left(\begin{bmatrix} \mathbf{g}_1 & \mathbf{g}_2 \\ \mathbf{g}_2^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{e}_i \\ \mathbf{e}_i^T & 1 \end{bmatrix} \right) = 2\mathbf{g}_2^T\mathbf{e}_i, \tag{62}
$$

so $\mathbf{g}_m = \frac{\partial \ell}{\partial m} = 2\mathbf{g}_2$. Similarly, we have $\mathbf{g}_V = \frac{\partial \ell}{\partial V} = \mathbf{g}_1$.

Note that in Gaussian cases, we have $\boldsymbol{\mu} = \mathbf{m}$ and $\boldsymbol{\Sigma} = \mathbf{V} - \mathbf{m}\mathbf{m}^T$, and thus we have

$$
\nabla_{\mu_i}\ell = \operatorname{Tr}\left(\left(\frac{\partial \ell}{\partial m} \right)^T \frac{\partial \mathbf{m}}{\partial \mu_i} \right) + \operatorname{Tr}\left(\left(\frac{\partial \ell}{\partial V} \right)^T \frac{\partial \mathbf{V}}{\partial \mu_i} \right) = \mathbf{g}_m^T\mathbf{e}_i + \operatorname{Tr}\left(\mathbf{g}_V^T(\mathbf{e}_i\mathbf{m}^T + \mathbf{m}\mathbf{e}_i^T) \right), \tag{63}
$$

$$
\nabla_{\Sigma_{jk}}\ell = \operatorname{Tr}\left(\left(\frac{\partial \ell}{\partial m} \right)^T \frac{\partial \mathbf{m}}{\partial \Sigma_{jk}} \right) + \operatorname{Tr}\left(\left(\frac{\partial \ell}{\partial V} \right)^T \frac{\partial \mathbf{V}}{\partial \Sigma_{jk}} \right) = 0 + \operatorname{Tr}\left(\mathbf{g}_V^T\mathbf{E}_{jk} \right), \tag{64}
$$

which implies that

$$
\begin{aligned}
\mathbf{g}_\mu &= \mathbf{g}_m + (\mathbf{g}_V + \mathbf{g}_V^T)\mathbf{m} = \mathbf{g}_m + 2\mathbf{g}_V\mathbf{m} = 2\mathbf{g}_2 + 2\mathbf{g}_1\boldsymbol{\mu}, \\
\mathbf{g}_\Sigma &= \mathbf{g}_V = \mathbf{g}_1.
\end{aligned} \tag{65}
$$

Thus, $\mathbf{g}_1$ and $\mathbf{g}_2$ can be reexpressed using Gaussian gradients $\mathbf{g}_\mu$ and $\mathbf{g}_\Sigma$ as $\mathbf{g}_1 = \mathbf{g}_\Sigma$ and $\mathbf{g}_2 = \frac{1}{2}(\mathbf{g}_\mu - 2\mathbf{g}_\Sigma\,\boldsymbol{\mu})$.

H. SPD Manifolds

H.1. Generalized Normal Coordinates

We first show that the local coordinate $\boldsymbol{\tau} = \boldsymbol{\phi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta}) = \mathbf{A}\mathbf{A}^T$ is a generalized normal coordinate defined at the reference point $\boldsymbol{\tau}^{(\text{cur})} = \mathbf{A}^{(\text{cur})}\big(\mathbf{A}^{(\text{cur})}\big)^T$, where $\boldsymbol{\eta} \in \mathbb{R}^{k\times k}$ is symmetric, and $\mathbf{A} = \mathbf{A}^{(\text{cur})}\text{Expm}(\frac{1}{2}\boldsymbol{\eta})$.

It is easy to verify that Assumption 1 holds since $\boldsymbol{\phi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta}_0) = \boldsymbol{\tau}^{(\text{cur})}$ at $\boldsymbol{\eta}_0 = \mathbf{0}$.

As shown in Appx. E.1, the metric is orthonormal at $\boldsymbol{\eta}_0 = \mathbf{0}$, so Assumption 2 holds.

Recall that the standard normal coordinate is $\boldsymbol{\tau} = \boldsymbol{\psi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta}) = \big(\boldsymbol{\tau}^{(\text{cur})}\big)^{1/2}\text{Expm}(\boldsymbol{\eta})\big(\boldsymbol{\tau}^{(\text{cur})}\big)^{1/2}$, where Assumption 3 holds. Our generalized normal coordinate is defined as $\boldsymbol{\tau} = \boldsymbol{\psi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta}) = \mathbf{A}^{(\text{cur})}\text{Expm}(\boldsymbol{\eta})\big(\mathbf{A}^{(\text{cur})}\big)^T$, where $\boldsymbol{\eta}$ is symmetric. The only difference between these two coordinates is a multiplicative constant. Differentiability and smoothness remain the same. The injectivity for symmetric $\boldsymbol{\eta}$ is due to the uniqueness of the symmetric square root of a matrix. Thus, Assumption 3 holds in our coordinate. This statement can be extended to the case where $\boldsymbol{\eta}$ is a triangular matrix due to the uniqueness of the Cholesky decomposition.

The space of symmetric matrices $\boldsymbol{\eta} \in \mathbb{R}^{k\times k}$ is an abstract vector space since scalar products and matrix additions of symmetric matrices are also symmetric. As a result, Assumption 4 holds.

H.2. Euclidean Gradients in Normal Coordinates

As mentioned in Sec. 3.2, there are many generalized normal coordinates such as

- $\boldsymbol{\phi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta}) = \mathbf{A}\mathbf{A}^T$, where $\boldsymbol{\eta}$ is symmetric and $\mathbf{A} := \mathbf{A}^{(\text{cur})}\text{Expm}(\frac{1}{2}\boldsymbol{\eta})$,
- $\boldsymbol{\phi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta}) = \mathbf{B}^{-T}\mathbf{B}^{-1}$, where $\boldsymbol{\eta}$ is symmetric and $\mathbf{B} := \mathbf{B}^{(\text{cur})}\text{Expm}(-\frac{1}{2}\boldsymbol{\eta})$,
- $\boldsymbol{\phi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta}) = \mathbf{C}^T\mathbf{C}$, where $\boldsymbol{\eta}$ is symmetric and $\mathbf{C} := \text{Expm}(\frac{1}{2}\boldsymbol{\eta})\mathbf{C}^{(\text{cur})}$.

We show how to compute the Euclidean gradient $\mathbf{g}(\boldsymbol{\eta}_0)$ needed in Eq. (11), where we assume that the Euclidean gradient $\nabla_{\boldsymbol{\tau}}\ell = \mathbf{g}(\boldsymbol{\tau})$ w.r.t. $\boldsymbol{\tau}$ is given. Let us consider $\boldsymbol{\tau} = \boldsymbol{\phi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta}) = \mathbf{A}\mathbf{A}^T$. By the chain rule, we have

$$\nabla_{\eta_{ij}}\ell = \text{Tr}\big(\mathbf{g}^T(\boldsymbol{\tau})\nabla_{\eta_{ij}}\boldsymbol{\tau}\big)\big|_{\boldsymbol{\eta}=\boldsymbol{\eta}_0} = \text{Tr}\big(\mathbf{g}^T(\boldsymbol{\tau})\mathbf{A}^{(\text{cur})}\mathbf{E}_{ij}(\mathbf{A}^{(\text{cur})})^T\big), \tag{66}$$

so

$$\mathbf{g}(\boldsymbol{\eta}_0) = (\mathbf{A}^{(\text{cur})})^T\mathbf{g}(\boldsymbol{\tau})\mathbf{A}^{(\text{cur})}. \tag{67}$$

Similarly, when $\boldsymbol{\tau} = \mathbf{B}^{-T}\mathbf{B}^{-1}$, we have

$$\nabla_{\eta_{ij}}\ell = \text{Tr}\big(\mathbf{g}^T(\boldsymbol{\tau})\nabla_{\eta_{ij}}\boldsymbol{\tau}\big)\big|_{\boldsymbol{\eta}=\boldsymbol{\eta}_0} = \text{Tr}\big(\mathbf{g}^T(\boldsymbol{\tau})(\mathbf{B}^{(\text{cur})})^{-T}\mathbf{E}_{ij}(\mathbf{B}^{(\text{cur})})^{-1}\big), \tag{68}$$

which gives

$$\mathbf{g}(\boldsymbol{\eta}_0) = (\mathbf{B}^{(\text{cur})})^{-1}\mathbf{g}(\boldsymbol{\tau})(\mathbf{B}^{(\text{cur})})^{-T}. \tag{69}$$

H.3. Simplification of Our Update

Consider the normal coordinate $\boldsymbol{\phi}_{\boldsymbol{\tau}^{(\text{cur})}}(\boldsymbol{\eta}) = \mathbf{A}\mathbf{A}^T$, where $\boldsymbol{\eta}$ is symmetric and $\mathbf{A} := \mathbf{A}^{(\text{cur})}\text{Expm}(\frac{1}{2}\boldsymbol{\eta})$.

We can compute the Euclidean gradient as $\mathbf{g}(\boldsymbol{\eta}_0) = (\mathbf{A}^{(\text{cur})})^T\mathbf{g}(\boldsymbol{\tau})\mathbf{A}^{(\text{cur})}$.

Using the approximation in Eq. (12), we have

$$\mathbf{w}^{(\eta_1)} \leftarrow \mathbf{m}^{(\eta_0)}. \tag{70}$$

Since $\boldsymbol{\eta}$ is symmetric, we can further show that the accurate approximation also gives the same update since the second dominant term vanishes as shown in Eq. (55).

By Eq. (38), the vector-Jacobian product needed in Eq. (13) can be expressed as

$$\mathbf{w}^{(\xi_0)} = \mathbf{J}(\boldsymbol{\xi}_0)\mathbf{w}^{(\eta_1)} = \mathbf{w}^{(\eta_1)}. \tag{71}$$

Thus, we have $\mathbf{w}^{(\xi_0)} = \mathbf{m}^{(\eta_0)}$. As a consequence, our update (defined in Eq. (11)) can be simplified as below, where we can drop all the superscripts and let $\mathbf{w} = \mathbf{m}$,

$$\begin{aligned}
\text{Momentum}: \mathbf{m} &\leftarrow \alpha\mathbf{m} + \beta \overbrace{(\mathbf{A}^{(\text{cur})})^T\mathbf{g}(\boldsymbol{\tau})\mathbf{A}^{(\text{cur})}}^{=\mathbf{g}(\boldsymbol{\eta}_0)},\\
\text{GD}: \boldsymbol{\eta}_1 &\leftarrow -\mathbf{m},\\
\boldsymbol{\tau}^{(\text{new})} &\leftarrow \boldsymbol{\phi}_{\tau^{(\text{cur})}}(\boldsymbol{\eta}_1) = \mathbf{A}^{(\text{cur})}\text{Expm}(\boldsymbol{\eta}_1)\big(\mathbf{A}^{(\text{cur})}\big)^T \Longleftrightarrow \mathbf{A}^{(\text{new})} \leftarrow \mathbf{A}^{(\text{cur})}\text{Expm}(\tfrac{1}{2}\boldsymbol{\eta}_1).
\end{aligned} \tag{72}$$

Using Eq. (69), we can also obtain the following update if the normal coordinate $\boldsymbol{\phi}_{\tau^{(\text{cur})}}(\boldsymbol{\eta}) = \mathbf{B}^{-T}\mathbf{B}^{-1}$ is used, where $\boldsymbol{\eta}$ is symmetric and $\mathbf{B} := \mathbf{B}^{(\text{cur})}\text{Expm}(-\frac{1}{2}\boldsymbol{\eta})$,

$$\begin{aligned}
\text{Momentum}: \mathbf{m} &\leftarrow \alpha\mathbf{m} + \beta \overbrace{(\mathbf{B}^{(\text{cur})})^{-1}\mathbf{g}(\boldsymbol{\tau})(\mathbf{B}^{(\text{cur})})^{-T}}^{=\mathbf{g}(\boldsymbol{\eta}_0)},\\
\text{GD}: \boldsymbol{\eta}_1 &\leftarrow -\mathbf{m},\\
\boldsymbol{\tau}^{(\text{new})} &\leftarrow \boldsymbol{\phi}_{\tau^{(\text{cur})}}(\boldsymbol{\eta}_1) = \big(\mathbf{B}^{(\text{cur})}\big)^{-T}\text{Expm}(-\boldsymbol{\eta}_1)\big(\mathbf{B}^{(\text{cur})}\big)^{-1} \Longleftrightarrow \mathbf{B}^{(\text{new})} \leftarrow \mathbf{B}^{(\text{cur})}\text{Expm}(-\tfrac{1}{2}\boldsymbol{\eta}_1).
\end{aligned} \tag{73}$$

I. SPD Kronecker-product Submanifolds

We consider the SPD submanifold

$$\mathcal{M} = \Big\{\boldsymbol{\tau} = \mathbf{A}\mathbf{A}^T \in \mathcal{S}_{++}^{(pd)\times(pd)} \mid \mathbf{A} := \mathbf{K}\otimes\mathbf{C}, \mathbf{K}\in\mathbb{R}^{p\times p}, \mathbf{C}\in\mathbb{R}^{d\times d}\Big\},$$

where $\mathbf{U} = \mathbf{K}\mathbf{K}^T \succ 0$, $\mathbf{W} = \mathbf{C}\mathbf{C}^T \succ 0$, and both $\mathbf{K}$ and $\mathbf{C}$ are dense and non-singular.

I.1. Blockwise Normal Coordinates

As mentioned in Sec. 4, we consider a block-diagonal approximation of the affine-invariant metric. For block $\mathbf{K}$, we consider the coordinate

$$\mathbf{A} = \big(\mathbf{K}^{(\text{cur})}\text{Expm}(\frac{1}{2\sqrt{d}}\boldsymbol{\eta}_K)\big)\otimes\mathbf{C}^{(\text{cur})}, \tag{74}$$

where $\boldsymbol{\eta}_K \in \mathbb{R}^{p\times p}$ is symmetric and $\mathbf{A}^{(\text{cur})} = \mathbf{K}^{(\text{cur})}\otimes\mathbf{C}^{(\text{cur})}$.

We will show that the block-diagonal approximated metric is orthonormal at $\boldsymbol{\eta}_K = \mathbf{0}$ under coordinate $\boldsymbol{\eta}_K$.

For notation simplicity, we let $\mathbf{K}_0 = \mathbf{K}^{(\text{cur})}$, $\mathbf{C}_0 = \mathbf{C}^{(\text{cur})}$, and $\boldsymbol{\tau}_0 = \boldsymbol{\tau}^{(\text{cur})}$. Let $\tilde{\boldsymbol{\eta}}_K$ denote the learnable part of $\boldsymbol{\eta}_K$.

By the Kronecker-product, we have $\text{vec}^T(\mathbf{X})(\mathbf{U}\otimes\mathbf{W})\text{vec}(\mathbf{X}) = \text{vec}^T(\mathbf{X})\text{vec}(\mathbf{W}\mathbf{X}\mathbf{U}^T) = \text{Tr}(\mathbf{X}^T\mathbf{W}\mathbf{X}\mathbf{U}^T)$, where $\mathbf{x} := \text{vec}(\mathbf{X})$ and $\mathbf{X}\in\mathbb{R}^{d\times p}$.

By definition, the metric $\mathbf{F}$ w.r.t. block $\mathbf{K}$ in coordinate $\boldsymbol{\eta}_K$ is

$$\begin{aligned}
&\mathbf{F}_K(\mathbf{0}) = -2\mathbb{E}_{\mathcal{N}(\mathbf{0},\boldsymbol{\tau})}\big[\nabla^2_{\tilde{\eta}_K}\log\mathcal{N}(\mathbf{0},\boldsymbol{\tau})\big]\big|_{\eta_K=0}\\
&= \mathbb{E}_{\mathcal{N}(x|\mathbf{0},\boldsymbol{\tau})}\Big[\nabla^2_{\tilde{\eta}_K}\Big\{\text{Tr}[\mathbf{x}^T\Big(\big(\mathbf{K}_0^{-T}\text{Expm}(-\frac{1}{\sqrt{d}}\boldsymbol{\eta}_K)\mathbf{K}_0^{-1}\big)\otimes\big(\mathbf{C}_0^{-T}\mathbf{C}_0^{-1}\big)\Big)\mathbf{x}]\Big\}\Big]\big|_{\eta_K=0}\ \text{(drop linear terms in the log-det term)}\\
&= \mathbb{E}_{\mathcal{N}(x|\mathbf{0},\boldsymbol{\tau})}\Big[\nabla^2_{\tilde{\eta}_K}\Big\{\text{Tr}[\mathbf{X}^T\big(\mathbf{C}_0^{-T}\mathbf{C}_0^{-1}\big)\mathbf{X}\big(\mathbf{K}_0^{-T}\text{Expm}(-\frac{1}{\sqrt{d}}\boldsymbol{\eta}_K)\mathbf{K}_0^{-1}\big)]\Big\}\Big]\big|_{\eta_K=0}\ \text{(note: } \mathbf{x}^T(\mathbf{U}\otimes\mathbf{W})\mathbf{x} = \text{Tr}(\mathbf{X}^T\mathbf{W}\mathbf{X}\mathbf{U}^T))\\
&= d\mathbb{E}_{\mathcal{N}(x|\mathbf{0},\boldsymbol{\tau})}\Big[\nabla^2_{\tilde{\eta}_E}\Big\{\text{Tr}[\text{Expm}(-\frac{1}{\sqrt{d}}\boldsymbol{\eta}_E)\}]\Big]\big|_{\eta_K=0}\ \text{(note: } \mathbb{E}_{\mathcal{N}(x|\mathbf{0},\boldsymbol{\tau}_0)}[\mathbf{X}^T\big(\mathbf{C}_0^{-T}\mathbf{C}_0^{-1}\big)\mathbf{X}] = d\mathbf{K}_0\mathbf{K}_0^T).
\end{aligned} \tag{75}$$

It is easy to show that $\mathbf{F}_K(\mathbf{0}) = \mathbf{I}$ w.r.t. block $\mathbf{K}$ in coordinate $\boldsymbol{\eta}_K$, which means Assumption 2 holds.

Since block $\mathbf{C}$ is frozen, we can prove as in Appx. H.1 that all assumptions are satisfied for the coordinate $\boldsymbol{\eta}_K$.

Similarly, for block $\mathbf{C}$, we can consider the coordinate

$$\mathbf{A} = \mathbf{K}^{(\text{cur})} \otimes \big(\mathbf{C}^{(\text{cur})}\text{Expm}(\frac{1}{2\sqrt{p}}\boldsymbol{\eta}_C)\big), \tag{76}$$

where $\boldsymbol{\eta}_C \in \mathbb{R}^{d\times d}$ is symmetric and $\mathbf{A}^{(\text{cur})} = \mathbf{K}^{(\text{cur})} \otimes \mathbf{C}^{(\text{cur})}$, and show that it defines a normal coordinate.

I.2. (Euclidean) Gradient Computation for Deep Learning

We consider $\boldsymbol{\tau} = \boldsymbol{\Sigma} = (\mathbf{K}\mathbf{K}^T) \otimes (\mathbf{C}\mathbf{C}^T)$.

As suggested by Lin et al. (2021b), the Euclidean gradient w.r.t. $\boldsymbol{\tau}$ is computed as $\mathbf{g}_\Sigma := \frac{1}{2}(\nabla^2_\mu \ell(\boldsymbol{\mu}) - \boldsymbol{\Sigma}^{-1})$.

In KFAC (Martens & Grosse, 2015), the Hessian is approximated as $\nabla^2_\mu \ell(\boldsymbol{\mu}) \approx \boldsymbol{\mu}_{AA} \otimes \boldsymbol{\mu}_{GG}$, where matrices $\boldsymbol{\mu}_{AA} \in \mathbb{R}^{p\times p}$ and $\boldsymbol{\mu}_{GG} \in \mathbb{R}^{d\times d}$ are two dense symmetric positive semi-definite matrices and are computed as suggested by the authors.

To handle the singularity of $\boldsymbol{\mu}_{AA}$ and $\boldsymbol{\mu}_{GG}$, Martens & Grosse (2015) introduce a damping term λ when it comes to inverting $\boldsymbol{\mu}_{AA}$ and $\boldsymbol{\mu}_{GG}$ such as $\boldsymbol{\mu}_{AA}^{-1} \approx (\boldsymbol{\mu}_{AA} + \lambda\mathbf{I}_p)^{-1}$ and $\boldsymbol{\mu}_{GG}^{-1} \approx (\boldsymbol{\mu}_{GG} + \lambda\mathbf{I}_d)^{-1}$.

In our update, we use the KFAC approach to approximate the Hessian. We add a damping term by including it in $\mathbf{g}_\Sigma$ as

$$\mathbf{g}_\Sigma \approx \tfrac{1}{2}(\underbrace{\boldsymbol{\mu}_{AA} \otimes \boldsymbol{\mu}_{GG}}_{\approx\nabla^2_\mu \ell(\boldsymbol{\mu})} + \lambda\mathbf{I}_{pd} - \boldsymbol{\Sigma}^{-1}), \tag{77}$$

where $\mathbf{I}_{pd} = \mathbf{I}_p \otimes \mathbf{I}_d$ and $\boldsymbol{\Sigma}^{-1} = (\mathbf{K}^{-T}\mathbf{K}^{-1}) \otimes (\mathbf{C}^{-T}\mathbf{C}^{-1})$.

The Euclidean gradient $\mathbf{g}(\boldsymbol{\eta}_{K_0})$ w.r.t. $\boldsymbol{\eta}_E$ can be computed as

$$\frac{\partial \ell}{\partial \eta_{K_{ij}}} = \text{Tr}\big((\frac{\partial \ell}{\partial \boldsymbol{\tau}})^T \frac{\partial \boldsymbol{\tau}}{\partial \eta_{K_{ij}}}\big) = \text{Tr}\big([\mathbf{g}_\Sigma]^T \frac{\partial \boldsymbol{\tau}}{\partial \eta_{K_{ij}}}\big). \tag{78}$$

There are three terms in the Euclidean gradient w.r.t. $\boldsymbol{\tau} = \boldsymbol{\Sigma}$:

$$\mathbf{g}_\Sigma \approx \tfrac{1}{2}(\boldsymbol{\mu}_{AA} \otimes \boldsymbol{\mu}_{GG} + \lambda\mathbf{I}_p \otimes \mathbf{I}_d - (\mathbf{K}^{-T}\mathbf{K}^{-1}) \otimes (\mathbf{C}^{-T}\mathbf{C}^{-1}). \tag{79}$$

Thus, the Euclidean gradient w.r.t. $\boldsymbol{\eta}$ can be decomposed into three parts. For notation simplicity, we let $\mathbf{K}_0 = \mathbf{K}^{(\text{cur})}$, $\mathbf{C}_0 = \mathbf{C}^{(\text{cur})}$, and $\boldsymbol{\tau}_0 = \boldsymbol{\tau}^{(\text{cur})}$.

The first part of $\frac{\partial \ell}{\partial \eta_{K_{ij}}}$ can be computed via

$$\begin{aligned}\tfrac{1}{2}\text{Tr}\big([\boldsymbol{\mu}_{AA} \otimes \boldsymbol{\mu}_{GG}]^T \frac{\partial \boldsymbol{\tau}}{\partial \eta_{K_{ij}}}\big) &= \frac{1}{2\sqrt{d}}\text{Tr}\big[(\boldsymbol{\mu}_{AA}^T\mathbf{K}_0\mathbf{E}_{ij}\mathbf{K}_0^T) \otimes (\boldsymbol{\mu}_{GG}^T\mathbf{C}_0\mathbf{C}_0^T)\big] \\ &= \frac{1}{2\sqrt{d}}\text{Tr}(\boldsymbol{\mu}_{GG}^T\mathbf{C}_0\mathbf{C}_0^T)\text{Tr}(\boldsymbol{\mu}_{AA}^T\mathbf{K}_0\mathbf{E}_{ij}\mathbf{K}_0^T).\end{aligned} \tag{80}$$

We obtain the expression for the first part of $\frac{\partial \ell}{\partial \eta_K}$ as $\frac{1}{2\sqrt{d}}\text{Tr}(\mathbf{C}_0^T\boldsymbol{\mu}_{GG}\mathbf{C}_0)\mathbf{K}_0^T\boldsymbol{\mu}_{AA}\mathbf{K}_0$.

Similarly, we can obtain the second and third parts, which gives altogether the Euclidean gradient $\mathbf{g}(\boldsymbol{\eta}_K)$ via

$$\mathbf{g}(\boldsymbol{\eta}_{K_0}) = \frac{1}{2\sqrt{d}}[\text{Tr}(\mathbf{C}_0^T\boldsymbol{\mu}_{GG}\mathbf{C}_0)\mathbf{K}_0^T\boldsymbol{\mu}_{AA}\mathbf{K}_0 + \lambda\text{Tr}(\mathbf{C}_0^T\mathbf{C}_0)\mathbf{K}_0^T\mathbf{K}_0 - d\mathbf{I}_p]. \tag{81}$$

Likewise, the Euclidean gradient $\mathbf{g}(\boldsymbol{\eta}_C)$ is

$$\mathbf{g}(\boldsymbol{\eta}_{C_0}) = \frac{1}{2\sqrt{p}}[\text{Tr}(\mathbf{K}_0^T\boldsymbol{\mu}_{AA}\mathbf{K}_0)\mathbf{C}_0^T\boldsymbol{\mu}_{GG}\mathbf{C}_0 + \lambda\text{Tr}(\mathbf{K}_0^T\mathbf{K}_0)\mathbf{C}_0^T\mathbf{C}_0 - p\mathbf{I}_d]. \tag{82}$$

I.3. Derivation of the Update

We consider the update for block $\boldsymbol{\eta}_K$. By the approximation in Eq. (12), we have

$$\mathbf{w}^{(\eta_{K_1})} \leftarrow \mathbf{m}^{(\eta_{K_0})}, \tag{83}$$

for block $\boldsymbol{\eta}_K$. Since $\boldsymbol{\eta}_K$ is symmetric, we can further show that the accurate approximation also gives the same update since the second dominant term vanishes as shown in Eq. (55).

Since $\boldsymbol{\eta}_K$ is symmetric (see Eq. (38)), the vector-Jacobian product needed in Eq. (13) can be expressed as

$$\mathbf{w}^{(\xi_{K_0})} = \mathbf{J}(\boldsymbol{\xi}_{K_0})\mathbf{w}^{(\eta_{K_1})} = \mathbf{w}^{(\eta_{K_1})}. \tag{84}$$

Thus, we have $\mathbf{w}^{(\xi_{K_0})} = \mathbf{m}^{(\eta_{K_0})}$ for $\boldsymbol{\eta}_K$.

As a consequence (similar to Appx. H.3), our update (defined in Eq. (11)) for block $\boldsymbol{\eta}_K$ can be expressed as below, where we drop all superscripts and let $\mathbf{w} = \mathbf{m}$,

$$\begin{aligned} \text{Momentum}: \mathbf{m}_K &\leftarrow \alpha\mathbf{m}_K + \beta\mathbf{g}(\boldsymbol{\eta}_{K_0}), \\ \text{GD}: \boldsymbol{\eta}_{K_1} &\leftarrow -\mathbf{m}_K, \\ \mathbf{K} &\leftarrow \mathbf{K}^{(\text{cur})}\text{Expm}(\frac{1}{2\sqrt{d}}\boldsymbol{\eta}_{K_1}). \end{aligned} \tag{85}$$

Since we initialize $\mathbf{m}_K$ to $\mathbf{0}$, we can merge factor $\frac{1}{2\sqrt{d}}$ into $\mathbf{m}_K$ as shown below.

$$\begin{aligned} \text{Momentum}: \mathbf{m}_K &\leftarrow \alpha\mathbf{m}_K + \frac{\beta}{2\sqrt{d}}\mathbf{g}(\boldsymbol{\eta}_{K_0}), \\ \text{GD}: \boldsymbol{\eta}_{K_1} &\leftarrow -\mathbf{m}_K, \\ \mathbf{K} &\leftarrow \mathbf{K}^{(\text{cur})}\text{Expm}(\boldsymbol{\eta}_{K_1}). \end{aligned} \tag{86}$$

Note that the affine-invariant metric is defined as twice of the Fisher-Rao metric.

To recover structured NGD, we have to set our stepsize β to twice the stepsize of structured NGD. Letting $\beta = 2\beta_2$, we can reexpress the above update for block $\boldsymbol{\eta}_K$ as

$$\begin{aligned} \mathbf{m}_K &\leftarrow \alpha\mathbf{m}_K + \frac{\beta_2}{\sqrt{d}}\mathbf{g}(\boldsymbol{\eta}_{K_0}), \\ \mathbf{K} &\leftarrow \mathbf{K}^{(\text{cur})}\text{Expm}(-\mathbf{m}_K). \end{aligned} \tag{87}$$

A similar update for the block $\boldsymbol{\eta}_C$ can also be obtained.

J. Implementation for the Baseline Methods

We consider the following manifold optimization problem on a SPD full manifold:

$$\min_{\boldsymbol{\tau} \in \mathcal{S}_{++}^{k\times k}} \ell(\boldsymbol{\tau}) \tag{88}$$

Recall that a Riemannian gradient w.r.t. $\boldsymbol{\tau}$ is $\hat{\mathbf{g}}(\boldsymbol{\tau}) := \boldsymbol{\tau}\,(\nabla_{\tau}\ell)\,\boldsymbol{\tau} = -\nabla_{\tau^{-1}}\ell$.

The Riemannian gradient descent (RGD) is

$$\boldsymbol{\tau}^{(\text{new})} \leftarrow \text{RExp}(\boldsymbol{\tau}^{(\text{cur})}, -\beta\hat{\mathbf{g}}(\boldsymbol{\tau}^{(\text{cur})})). \tag{89}$$

The update of Alimisis et al. (2020) is shown below, where we initialize $\mathbf{z}$ by 0:

$$\begin{aligned}\boldsymbol{\nu}^{(\text{cur})} &\leftarrow \alpha \mathbf{z}^{(\text{cur})} + \beta \hat{\mathbf{g}}(\boldsymbol{\tau}^{(\text{cur})}) \\ \boldsymbol{\tau}^{(\text{new})} &\leftarrow \text{RExp}(\boldsymbol{\tau}^{(\text{cur})}, -\boldsymbol{\nu}^{(\text{cur})}) \\ \mathbf{z}^{(\text{new})} &\leftarrow \hat{T}_{\tau^{(\text{cur})} \rightarrow \tau^{(\text{new})}}(\boldsymbol{\nu}^{(\text{cur})})\end{aligned} \tag{90}$$

The update of Alimisis et al. (2021) is shown below, where we initialize $\mathbf{y}$ and $\mathbf{z}$ by $\boldsymbol{\tau}$:

$$\begin{aligned}\mathbf{z}^{(\text{new})} &\leftarrow \text{RExp}(\boldsymbol{\tau}^{(\text{cur})}, -\beta \hat{\mathbf{g}}(\boldsymbol{\tau}^{(\text{cur})})) \\ \mathbf{y}^{(\text{new})} &\leftarrow \text{RExp}(\mathbf{y}^{(\text{cur})}, -\frac{\beta}{1-\alpha} \hat{T}_{\tau^{(\text{cur})} \rightarrow y^{(\text{cur})}}(\hat{\mathbf{g}}(\boldsymbol{\tau}^{(\text{cur})}))) \\ \boldsymbol{\tau}^{(\text{new})} &\leftarrow \text{RExp}(\mathbf{y}^{(\text{new})}, \alpha \text{RExp}^{-1}(\mathbf{y}^{(\text{new})}, \mathbf{z}^{(\text{new})}))\end{aligned} \tag{91}$$

The update of Ahn & Sra (2020) is shown below, where we initialize $\mathbf{y}$ and $\mathbf{z}$ by $\boldsymbol{\tau}$:

$$\begin{aligned}\mathbf{y}^{(\text{new})} &\leftarrow \text{RExp}(\boldsymbol{\tau}^{(\text{cur})}, -\beta \hat{\mathbf{g}}(\boldsymbol{\tau}^{(\text{cur})})) \\ \mathbf{z}^{(\text{new})} &\leftarrow \text{RExp}(\boldsymbol{\tau}^{(\text{cur})}, \frac{\alpha}{1-\alpha} \text{RExp}^{-1}(\boldsymbol{\tau}^{(\text{cur})}, \mathbf{z}^{(\text{cur})}) - 2\beta \hat{\mathbf{g}}(\boldsymbol{\tau}^{(\text{cur})})) \\ \boldsymbol{\tau}^{(\text{new})} &\leftarrow \text{RExp}(\mathbf{y}^{(\text{new})}, \alpha \text{RExp}^{-1}(\mathbf{y}^{(\text{new})}, \mathbf{z}^{(\text{new})}))\end{aligned} \tag{92}$$

We properly select momentum weights and stepsizes in Ahn & Sra (2020) and Alimisis et al. (2020; 2021) so that these updates are equivalent in Euclidean cases.

Recall that our update with momentum in the GNC $\boldsymbol{\tau} = \mathbf{C}^T \mathbf{C}$ is

$$\begin{aligned}\mathbf{m}^{(\text{new})} &\leftarrow \alpha \mathbf{m}^{(\text{cur})} + \beta \overbrace{(\mathbf{C}^{(\text{cur})})^{-T} \hat{\mathbf{g}}(\boldsymbol{\tau}^{(\text{cur})}) (\mathbf{C}^{(\text{cur})})^{-1}}^{= \mathbf{C}^{(\text{cur})} \left(\nabla_\tau \ell(\boldsymbol{\tau}^{(\text{cur})})\right) \left(\mathbf{C}^{(\text{cur})}\right)^T} \\ \boldsymbol{\eta}_1 &\leftarrow \mathbf{0} - \mathbf{m}^{(\text{new})} \\ \mathbf{C}^{(\text{new})} &\leftarrow \text{Expm}(\tfrac{1}{2}\boldsymbol{\eta}_1) \mathbf{C}^{(\text{cur})} \\ \boldsymbol{\tau}^{(\text{new})} &\leftarrow \left(\mathbf{C}^{(\text{new})}\right)^T \mathbf{C}^{(\text{new})}\end{aligned} \tag{93}$$

where we initialize $\mathbf{m}$ by 0, and we use the quadratic truncation of the matrix exponential as $\text{Expm}(\mathbf{N}) \approx \mathbf{I} + \mathbf{N} + \frac{1}{2}\mathbf{N}^2$. Thus,

$$\text{Expm}(\mathbf{N})\mathbf{C} \approx \tfrac{1}{2}\left(\mathbf{I} + (\mathbf{I} + \mathbf{N})(\mathbf{I} + \mathbf{N})\right)\mathbf{C}. \tag{94}$$

Note that when $\mathbf{N}$ is a symmetric matrix, we have $\mathbf{I} + \mathbf{N} + \frac{1}{2}\mathbf{N}^2 = \frac{1}{2}\left(\mathbf{I} + (\mathbf{I} + \mathbf{N})(\mathbf{I} + \mathbf{N})^T\right) \succ 0$. Since $\boldsymbol{\eta}_1$ is a symmetric matrix, we know that the updated $\boldsymbol{\tau}$ is SPD even when we use the truncation.

We can recover the update of Lin et al. (2021a) on a SPD manifold by setting $\alpha = 0$ in Eq. (93).

For a Gaussian submanifold $\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T & \boldsymbol{\mu} \\ \boldsymbol{\mu}^T & 1 \end{bmatrix}$ considered in Sec. 3.3.1, the update of Lin et al. (2021a) on this submanifold is shown below, where we can use the Gaussian gradient identities in Eq. (65) and $\boldsymbol{\Sigma} = \mathbf{U}^T\mathbf{U}$:

$$\begin{aligned}\boldsymbol{\mu} &\leftarrow \boldsymbol{\mu} - \frac{\beta}{2}\boldsymbol{\Sigma}\left(\nabla_\mu \ell\right) \\ \mathbf{U} &\leftarrow \text{Expm}\left(-\frac{\beta}{2}\mathbf{U}\left(\nabla_\Sigma \ell\right)\mathbf{U}^T\right)\mathbf{U}\end{aligned} \tag{95}$$

where we also use the quadratic truncation for the matrix exponential.