Wrapped Flow Matching for Symmetric Positive Definite Matrices

Thibault de Surrel *thibault.de-surrel@lamsade.dauphine.fr*
LAMSADE, CNRS,
Université Paris Dauphine-PSL,
Paris, France

Aimi Okabayashi *aimi.okabayashi@univ-ubs.fr*
IRISA, UMR CNRS 6074,
Université Bretagne Sud, Vannes, France

Nicolas Courty *ncourty@irisa.fr*
IRISA, UMR CNRS 6074,
Université Bretagne Sud, Vannes, France

Reviewed on OpenReview: *https://openreview.net/forum?id=1og5KW8QBv*

Abstract

Conditional Flow Matching (CFM) has emerged as a state-of-the-art paradigm for generative modeling, yet its extension to non-Euclidean geometries remains an open challenge. Current RiemannianCFM methods rely strictly on geodesic interpolation or on pullback to Euclidean spaces, limiting their expressivity. In this work, we propose a flexible formulation for RiemannianCFM that leverages the wrapped Gaussian distribution to construct conditional probability paths. Our approach provides a unified framework that encompasses existing RiemannianCFM variants, enabling more adaptable flows that respect the intrinsic geometry of the manifold. Moreover, we are able to derive from this framework a new method that we coined *WrappedCFM*. We focus our development on the manifold of Symmetric Positive Definite (SPD) matrices denoted $\mathcal{P}_d$, a core component of information geometry. Our geometric formulation respects the intrinsic structure of $\mathcal{P}_d$ and relies neither on extrinsic Euclidean approximations nor on geodesic interpolation. Through both synthetic and real-world experiments, we demonstrate that WrappedCFM accurately models complex distributions on $\mathcal{P}_d$ and achieves competitive performance.

1 Introduction

Denoising Diffusion Probabilistic Models (DDPM) (Ho et al., 2020; Song et al., 2021) and Conditional Flow Matching (CFM) (Lipman et al., 2023; Liu et al., 2022; Jung et al., 2024) have established themselves as the state-of-the-art (SOTA) for generative modeling on Euclidean data. However, real-world data often possess a richer geometric structure, and modeling them as lying on a manifold is often a powerful solution (Papillon et al., 2025). This assumption requires to adapt classical methods to capture the intrinsic geometry of the underlying space (Pennec, 2006; Guigui et al., 2023).

In this paper, we bridge this gap by extending CFM algorithms to the Riemannian manifold $\mathcal{P}_d$ of Symmetric Positive Definite (SPD) matrices. SPD matrices are widely used throughout statistics and machine learning, appearing in domains such as *Brain Computer Interfaces* (BCI) (Lotte et al., 2018), *Diffusion Tensor Imaging* (Pennec, 2020), *radar signals* (Brooks et al., 2019) or *hyperspectral imaging* (Rußwurm et al., 2020). This manifold is also of central interest in information geometry where it is associated to the statistical model of multivariate centered Gaussians (Skovgaard, 1984). The space $\mathcal{P}_d$ is a

curved Riemannian manifold, and taking into account this non-flat geometry is crucial when manipulating SPD matrices (see for example the *swelling effect* in Arsigny et al. 2007).

In this paper, we propose two main contributions.

1. **First, the key novelty** is a general framework for CFM on $\mathcal{P}_d$, encompassing SOTA RiemannianCFM methods: *RiemannianCFM* from Chen & Lipman (2024) and *DiffeoCFM* from Collas et al. (2025). For this, we leverage an infinitely supported distribution defined on the manifold $\mathcal{P}_d$, the *wrapped Gaussian* (de Surrel et al., 2025), to design the conditional probability paths.

2. **Secondly**, using the flexibility of our framework, we propose a novel interpolation scheme that extends beyond restricted geodesic paths. We call this method *WrappedCFM*.

This paper is organized as follows: after reviewing SOTA CFM methods on Riemannian manifolds in Section 2, we introduce the Riemannian geometry of $\mathcal{P}_d$ as well as the wrapped Gaussian distribution in Section 3. Next, we recall the classical theory of EuclideanCFM and review two SOTA RiemannianCFM methods in Section 4. We then introduce our unified framework in Section 5 and propose our new interpolation *WrappedCFM* in Section 6. We finally validate our method on synthetic data in Section 7 and on real-world data in Section 8.

2 Related works

The works of Ho et al. (2020); Song et al. (2021) on Denoising Diffusion Probabilistic Models (DDPM) and of Lipman et al. (2023); Liu et al. (2022); Jung et al. (2024) on Conditional Flow Matching (CFM) have led to SOTA generative models. Several attempts have been made to generalize these models to Riemannian manifolds. In Li et al. (2023), the authors propose a DDPM specific to the manifold $\mathcal{P}_d$ of SPD matrices: using the isotropic Gaussian of Said et al. (2016), they adapt the classical forward and backward process of DDPM to $\mathcal{P}_d$. Other Riemannian adaptations of the diffusion generative framework have been proposed for general Riemannian manifolds (Jo & Hwang, 2023; De Bortoli et al., 2022; Huang et al., 2022) that learn to reverse the noising process. However, most of these works struggle to perform on complex manifolds (outside of the sphere or the torus) as they rely on approximations of the intractable heat kernel. Moreover, they lack a clear geometrical interpretation, which will not be the case for the proposed framework.

Regarding CFM, Chen & Lipman (2024) propose to adapt the EuclideanCFM framework to the setup of a Riemannian manifold. They generalize the linear interpolation used to build conditional probability paths to geodesic interpolation, building Riemannian conditional probability paths based on geodesics. In our work, we focus on infinitely supported Gaussian conditional probability paths. The RiemannianCFM with geodesic interpolation has been used in Bose et al. (2023) to generate novel protein structures working on the special Euclidean group SE(3) or in Braun et al. (2024) to learn and generate robot sensorimotor policies working on the hypersphere or on SO(3). In Zaghen et al. (2025), the authors developed a Riemannian version of Variational Flow Matching, based on the isotropic Gaussian of Said et al. (2016). Another approach to adapting CFM to Riemannian manifolds is to use pullback geometries, as in Collas et al. (2025). Their so-called *DiffeoCFM* sends the data to a Euclidean space E (for example a specific tangent space) *via* a global diffeomorphism and relies on EuclideanCFM in E before sending back the generated samples to the manifold. Our approach is different as we work directly on the manifold without relying on any artificial Euclidean space. In their paper, the authors apply their DiffeoCFM to fMRI and EEG datasets. A similar approach is Zhou et al. (2025), in which the authors use the Cholesky decomposition to perform efficient FM on the simpler manifold of lower-triangular matrices with positive diagonal.

3 The manifold of SPD matrices

Bold uppercase letters ($\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V}$) denote matrices. For $d \in \mathbb{N}^\star$, let $D := d(d+1)/2$. $\boldsymbol{I}_d$ is the $d \times d$ identity matrix, while exp and log respectively refer to the matrix exponential and logarithm.

3.1 The Riemannian geometry of SPD matrices

The set of $d \times d$ *Symmetric Positive Definite* (SPD) matrices, denoted by $\mathcal{P}_d$, is defined as

$$\mathcal{P}_d = \{\boldsymbol{X} \in \mathbb{R}^{d \times d} \mid \boldsymbol{X}^\top = \boldsymbol{X} \text{ and } \forall z \neq 0,\ z^\top \boldsymbol{X} z > 0\}.$$

This set is an open convex cone in the space $\mathcal{S}_d$ of symmetric matrices. Therefore, it is a manifold of dimension $D = d(d+1)/2$ and the tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$ at any point $\boldsymbol{X} \in \mathcal{P}_d$ is identified with $\mathcal{S}_d$. A Riemannian metric can be defined on $\mathcal{P}_d$ by endowing each tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$ with the following inner product, called the *Affine Invariant Riemannian Metric* (AIRM) (Pennec, 2020):

$$\forall \boldsymbol{U}, \boldsymbol{V} \in T_{\boldsymbol{X}}\mathcal{P}_d,\ \langle \boldsymbol{U}, \boldsymbol{V} \rangle_{\boldsymbol{X}} = \operatorname{tr}(\boldsymbol{X}^{-1}\boldsymbol{U}\boldsymbol{X}^{-1}\boldsymbol{V}). \tag{1}$$

Once endowed with this metric, $\mathcal{P}_d$ is a geodesically complete manifold with non-positive curvature, thus $\mathcal{P}_d$ forms a *Hadamard manifold* (Shiga, 1984). The AIRM distance δ between two points $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{P}_d$ is:

$$\delta(\boldsymbol{X}, \boldsymbol{Y}) = \| \log(\boldsymbol{X}^{-1/2}\boldsymbol{Y}\boldsymbol{X}^{-1/2}) \|_F, \tag{2}$$

where $\| \cdot \|_F$ is the Frobenius norm. We will also need the exponential map $\operatorname{Exp}_{\boldsymbol{X}} : T_{\boldsymbol{X}}\mathcal{P}_d \to \mathcal{P}_d$ and its inverse, the logarithmic map $\operatorname{Log}_{\boldsymbol{X}} : \mathcal{P}_d \to T_{\boldsymbol{X}}\mathcal{P}_d$. As $\mathcal{P}_d$ is a Hadamard manifold, these maps are diffeomorphisms. Moreover, Bhatia (2007) gives them in closed form:

$$\operatorname{Exp}_{\boldsymbol{X}}(\boldsymbol{U}) = \boldsymbol{X}^{1/2} \exp(\boldsymbol{X}^{-1/2}\boldsymbol{U}\boldsymbol{X}^{-1/2})\boldsymbol{X}^{1/2}, \text{ and } \operatorname{Log}_{\boldsymbol{X}}(\boldsymbol{Y}) = \boldsymbol{X}^{1/2} \log(\boldsymbol{X}^{-1/2}\boldsymbol{Y}\boldsymbol{X}^{-1/2})\boldsymbol{X}^{1/2}. \tag{3}$$

Remark 3.1. Other Riemannian metrics have been defined on the manifold $\mathcal{P}_d$ such as the *log-Euclidean metric* (Arsigny et al., 2007), the *Cholesky metric* (Dryden et al., 2009) or the *Bures-Wasserstein metric* (Bhatia et al., 2019). In this work, we decided to focus on the AIRM metric as it leads to a curved space (unlike the log-Euclidean nor the Cholesky metric) with a non-positive curvature (unlike the Bures-Wasserstein metric). This makes the space interesting (non-flat) with nice mathematical properties ($\operatorname{Exp}_{\boldsymbol{X}}$ is a diffeomorphism). Moreover, it is among the most used metrics in practice (Chevallier et al., 2021).

3.2 Wrapped Gaussian on $\mathcal{P}_d$

There are several ways to define a Gaussian distribution on $\mathcal{P}_d$ (Pennec, 2006; Said et al., 2016). In this work, we consider the *wrapped Gaussian distribution* (de Surrel et al., 2025; Chevallier et al., 2022), which is defined as the pushforward of a Euclidean multivariate Gaussian distribution from a tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$ at a given point $\boldsymbol{X} \in \mathcal{P}_d$ by the exponential map $\operatorname{Exp}_{\boldsymbol{X}}$. In order to simplify the notations, we consider in all this paper the affine-invariant (AIRM-orthonormal) identification of $T_{\boldsymbol{X}}\mathcal{P}_d$ with $\mathbb{R}^D$, i.e. a tangent vector $\boldsymbol{U} \in T_{\boldsymbol{X}}\mathcal{P}_d$ is represented by the coordinates of $\boldsymbol{X}^{-1/2}\boldsymbol{U}\boldsymbol{X}^{-1/2}$ in a fixed Frobenius-orthonormal basis of $\mathcal{S}_d$. More on this in de Surrel et al. (2025) or in Section 3.3.3.3 of Pennec (2020).

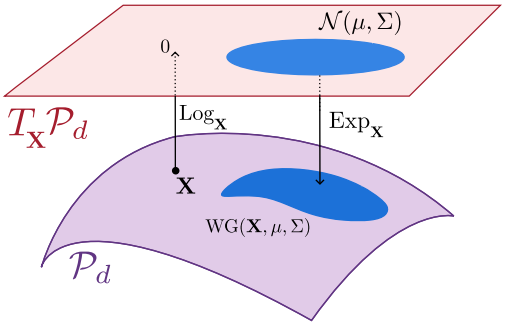


Figure 1: An illustration of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{X}, \mu, \Sigma)$ defined on the manifold $\mathcal{P}_d$ of SPD matrices.

Definition 3.2 (Wrapped Gaussian)**.** Let $\boldsymbol{X} \in \mathcal{P}_d$, $\mu \in \mathbb{R}^D$ and $\Sigma \in \mathcal{P}_D$. A random variable $\boldsymbol{Z}$ on $\mathcal{P}_d$ follows a wrapped Gaussian (WG) denoted $\mathrm{WG}(\boldsymbol{X}, \mu, \Sigma)$ if

$$\boldsymbol{Z} = \operatorname{Exp}_{\boldsymbol{X}}(\mathbf{t}),\ \mathbf{t} \sim \mathcal{N}(\mu, \Sigma).$$

In other words, $\mathrm{WG}(\boldsymbol{X}, \mu, \Sigma) = \operatorname{Exp}_{\boldsymbol{X}} \#\mathcal{N}(\mu, \Sigma)$, where $\#$ denotes the push-forward. This distribution relies on three parameters: $\boldsymbol{X} \in \mathcal{P}_d$ is the anchor point of the distribution, which gives a tangent space from which the Euclidean Gaussian will be pushed forward. Then, $\mu \in \mathbb{R}^D$ is the mean of the distribution in the tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$ and Σ is the covariance matrix in the tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$.

This distribution has been studied in more depth in de Surrel et al. (2025). It is simple to sample from a wrapped Gaussian $\mathrm{WG}(\boldsymbol{X}, \mu, \Sigma)$, as it solely requires sampling from a Euclidean Gaussian $\mathcal{N}(\mu, \Sigma)$ and applying the exponential map given at Equation (3). The wrapped Gaussian is illustrated at Figure 1. Let us give an important rescaling property of the wrapped Gaussian that will be used in the following:

Proposition 3.3 (Rescaling property)**.** *Let $\boldsymbol{X} \in \mathcal{P}_d, \mu \in \mathbb{R}^D, \sigma > 0$ and $\boldsymbol{Z} \sim \mathrm{WG}(\boldsymbol{I}_d, 0_D, \boldsymbol{I}_D)$. Then*

- $\boldsymbol{X}^{1/2}\boldsymbol{Z}\boldsymbol{X}^{1/2} \sim \mathrm{WG}(\boldsymbol{X}, 0_D, \boldsymbol{I}_D)$,
- $\exp(\log(\boldsymbol{Z}) + \mu) \sim \mathrm{WG}(\boldsymbol{I}_d, \mu, \boldsymbol{I}_D)$
- $\boldsymbol{Z}^\sigma = \exp(\sigma \log \boldsymbol{Z}) \sim \mathrm{WG}(\boldsymbol{I}_d, 0_D, \sigma^2 \boldsymbol{I}_D)$.

This proposition is a special case of Proposition 4.5 of de Surrel et al. (2025). Finally, we give a last proposition on the spreading of the wrapped Gaussian with respect to d. The proof is given at Appendix A.

Proposition 3.4. *Let $\boldsymbol{Z} \sim \mathrm{WG}(\boldsymbol{I}_d, 0_D, \boldsymbol{I}_D)$, then*

$$\mathbb{E}[\delta(\boldsymbol{I}_d, \boldsymbol{Z})] = \sqrt{2}\, \frac{\Gamma\left(\frac{d(d+1)+2}{4}\right)}{\Gamma\left(\frac{d(d+1)}{4}\right)} \sim \sqrt{\frac{d(d+1)}{2}} \quad (d \to +\infty),$$

where Γ is the Gamma function (see Eq. 16 of Davis 1959).

Remark 3.5. Several other Gaussians have been defined on the manifold $\mathcal{P}_d$ of SPD matrices, such as the entropy-maximizing Gaussian proposed by Pennec (2006) or the isotropic Gaussian studied in Said et al. (2017). Here, we focus on the wrapped Gaussian as it comes with a very simple sampling algorithm, compared to the previously cited alternatives.

4 Background on Conditional Flow Matching

In this section, we recall the classical theory of Conditional Flow Matching (CFM). The general goal of CFM is to sample points from an unknown data distribution q_1. In order to do this, CFM learns a parametric map Φ that pushes forward a "simple" distribution p (e.g. a Gaussian) to the data distribution q_1. We start by describing the classical EuclideanCFM and then we describe two SOTA approaches to adapt CFM to a Riemannian manifold.

4.1 The classical Euclidean Conditional Flow Matching

Let us first start by describing the classical Euclidean Conditional Flow Matching (CFM) framework on $\mathbb{R}^d$ from Lipman et al. (2023). Let $p = \mathcal{N}(0, \boldsymbol{I}_d)$ be the source distribution and q_1 be the target distribution. In order to build a model capable of generating new samples from q_1, the Flow Matching framework builds a *probability path* $(p_t)_{0 \leqslant t \leqslant 1}$ such that $p_0 = p$ and $p_1 = q_1$. To model it, a time dependent flow $\Phi : [0,1] \times \mathbb{R}^d \longrightarrow \mathbb{R}^d$ is defined as the solution of the ODE driven by a vector field $u_t(\cdot)$ at time t and initial condition $\Phi_0(x) = x$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(x) = u_t(\Phi_t(x))\,, \quad \Phi_0(x) = x. \tag{4}$$

The flow Φ pushes p to $p_t = \Phi_t \# p$. Thus, transforming samples of p with Φ_1 will give a sample of q_1. Moreover, p_t and u_t are linked by the continuity equation:

$$\partial_t p_t(x) + \mathrm{div}(p_t(x)\, u_t(x)) = 0. \tag{5}$$

Assuming that we know how to define p_t and u_t, one could regress the vector field u_t with a neural network $v_t^\theta(x)$ directly:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{\substack{t \sim \mathcal{U}(0,1) \\ x \sim p_t}} \|v_t^\theta(x) - u_t(x)\|^2.$$

The main challenge lies in constructing such a flow without access to the explicit form of p_1, relying solely on available samples. Therefore, Lipman et al. (2023) introduces CFM by designing p_t as a mixture of tractable conditional probability paths $p_t(\cdot \mid z)$ that vary with some conditioning variable z:

$$p_t(x) = \int p_t\left(x \mid z\right) q\left(z\right) \mathrm{d}z,$$

where $q(z)$ is a distribution over the conditioning variable. The choice of the conditional probability path $p_t(\cdot \mid z)$ defines a conditional vector field $u_t(\cdot \mid z)$, such that the marginalization against z of $u_t(\cdot \mid z)$ generates p_t:

$$u_t(x) = \int u_t\left(x \mid z\right) \frac{p_t\left(x \mid z\right) q\left(z\right)}{p_t(x)} \mathrm{d}z.$$

Therefore, rather than regressing $u_t(x)$, we regress the conditional vector field $u_t(x \mid z)$:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{\substack{t\sim\mathcal{U}(0,1) \\ z\sim q \\ x\sim p_t(\cdot|z)}} \left\| v_t^\theta(x) - u_t(x \mid z) \right\|^2.$$

The power of the CFM framework lies in the fact that $\mathcal{L}_{\mathrm{CFM}}$ and $\mathcal{L}_{\mathrm{FM}}$ are equal up to a constant (see Theorem 2 of Lipman et al. 2023). Therefore, their gradients are equal and minimizing $\mathcal{L}_{\mathrm{CFM}}$ will also minimize $\mathcal{L}_{\mathrm{FM}}$. By choosing a conditional variable $x_1 \sim q_1$ from the target distribution (in practice, x_1 is a random training point), and Gaussian probability paths $p_t(x \mid x_1) = \mathcal{N}(x \mid \mu_t(x_1), \sigma_t^2(x_1)\boldsymbol{I}_d)$, the Euclidean conditional vector field $u_t\left(x \mid x_1\right)$ is given by (see Lipman et al. 2023):

$$u_t\left(x \mid x_1\right) = \frac{\sigma_t'\left(x_1\right)}{\sigma_t\left(x_1\right)} \left(x - \mu_t\left(x_1\right)\right) + \mu_t'\left(x_1\right).$$

4.2 Riemannian Conditional Flow Matching

Now that we have explained how EuclideanCFM works, let us describe two possible ways of adapting it when the data distribution q_1 lives on the Riemannian manifold $\mathcal{P}_d$ of SPD matrices. We start by describing *RiemannianCFM* introduced in Chen & Lipman (2024) that relies on geodesic interpolation and then, we review *DiffeoCFM*, from Collas et al. (2025) that leverages the tangent spaces.

RiemannianCFM In Chen & Lipman (2024), the authors propose to adapt the EuclideanCFM described in Section 4.1 to the case of a Riemannian manifold. For this, the conditioning variable z is a pair $(\boldsymbol{X}_0, \boldsymbol{X}_1) \sim q = p(\boldsymbol{X}_0)q_1(\boldsymbol{X}_1)$ sampled from the source distribution p and the target distribution q_1 (in practice, $\boldsymbol{X}_1$ is uniformly chosen from the training dataset). Then, they leverage geodesics to build the conditional probability paths $p_t^{\mathrm{RCFM}}(\cdot \mid \boldsymbol{X}_0, \boldsymbol{X}_1)$, as they are the shortest paths between two points on a manifold. The resulting conditional probability path $p_t^{\mathrm{RCFM}}(\cdot \mid \boldsymbol{X}_0, \boldsymbol{X}_1)$ and its associated flow are:

$$\begin{aligned} p_t^{\mathrm{RCFM}}(\cdot \mid \boldsymbol{X}_0, \boldsymbol{X}_1) &= \delta_{\gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t)} \\ \Phi_t^{\mathrm{RCFM}}(\boldsymbol{X} \mid \boldsymbol{X}_0, \boldsymbol{X}_1) &= \gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t), \end{aligned} \tag{6}$$

where $\delta_{\boldsymbol{X}}$ is the Dirac distribution at $\boldsymbol{X}$ and $\gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}$ is the unique geodesic from $\boldsymbol{X}_0$ to $\boldsymbol{X}_1$. Then, the conditional vector field $u_t^{\mathrm{RCFM}}(\cdot \mid \boldsymbol{X}_0, \boldsymbol{X}_1)$ that generates $p_t^{\mathrm{RCFM}}(\cdot \mid \boldsymbol{X}_0, \boldsymbol{X}_1)$ is simply $\frac{\mathrm{d}}{\mathrm{d}t}\gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t)$.

DiffeoCFM The authors of Collas et al. (2025) use a Euclidean space E to pullback the distributions p and q_1 onto E and then rely on classical EuclideanCFM in E as described in Section 4.1. When the space of interest is $\mathcal{P}_d$, the chosen Euclidean space E is the tangent space at identity $T_{\boldsymbol{I}_d}\mathcal{P}_d$ and the pullback is the matrix logarithm $\log = \mathrm{Log}_{\boldsymbol{I}_d}$. Thus, by denoting p_t^{E} the Euclidean conditional probability path and by Φ_t^{E} the Euclidean flow (as given in Lipman et al. 2023), we can express them for *DiffeoCFM*:

$$\begin{aligned} p_t^{\mathrm{DiffeoCFM}}(\cdot \mid \boldsymbol{X}_1) &= \exp \# p_t^{\mathrm{E}}(\cdot \mid \log \boldsymbol{X}_1) \\ \Phi_t^{\mathrm{DiffeoCFM}}(\cdot \mid \boldsymbol{X}_1) &= \exp \circ \Phi_t^{\mathrm{E}}(\cdot \mid \log \boldsymbol{X}_1) \end{aligned} \tag{7}$$

In this case, the conditional variable z is the log of a point $\boldsymbol{X}_1 \sim q_1$. One can also express the conditional vector field generating $p_t^{\text{DiffeoCFM}}(\cdot \mid \boldsymbol{X}_1)$ (see Equation (7) of Collas et al. 2025):

$$u_t^{\text{DiffeoCFM}}(\boldsymbol{X} \mid \boldsymbol{X}_1) = \mathrm{d}\exp(\log \boldsymbol{X}) \cdot \left[u_t^{\text{E}}(\log \boldsymbol{X})\right]. \tag{8}$$

where $\mathrm{d}\exp(\boldsymbol{U}) \cdot \boldsymbol{V}$ is the differential of exp at $\boldsymbol{U}$ applied at $\boldsymbol{V}$.

5 A global framework unifying Riemannian FM using the wrapped Gaussian

Now that we have described the general approach of CFM and two specific Riemannian methods, *RiemannianCFM* and *DiffeoCFM*, we propose a general CFM framework on the Riemannian manifold $\mathcal{P}_d$ of SPD matrices. This framework encompasses *RiemannianCFM* and *DiffeoCFM* but can also propose new RiemannianCFM methods. This framework is partially inspired by the one provided in Tong et al. (2024a).

5.1 The general framework

This framework relies on the wrapped Gaussian distribution described at Section 3.2. In this framework, the source distribution p will be a simple centered wrapped Gaussian: $p = \mathrm{WG}(\boldsymbol{I}_d, 0_D, \sigma_{\text{init}}^2 \boldsymbol{I}_D)$. This distribution is convenient due to its ease of sampling (see Section 3.2). Let us now consider a conditioning variable $\boldsymbol{Z} \sim q$, where $q(\boldsymbol{Z})$ is a distribution over the conditioning variable. We consider the following conditional probability path $(p_t(\cdot \mid \boldsymbol{Z}))_{0 \leqslant t \leqslant 1}$:

$$p_t(\cdot \mid \boldsymbol{Z}) = \mathrm{WG}(\bar{\boldsymbol{X}}_t(\boldsymbol{Z}), \mu_t(\boldsymbol{Z}), \sigma_t(\boldsymbol{Z})^2 \boldsymbol{I}_D), \tag{9}$$

where $\bar{\boldsymbol{X}} \colon [0,1] \times \mathcal{P}_d \to \mathcal{P}_d$, $\mu \colon [0,1] \times \mathcal{P}_d \to \mathbb{R}^D$ and $\sigma \colon [0,1] \times \mathcal{P}_d \to \mathbb{R}_{\geqslant 0}$ are C^1 curves parameterizing the conditional probability path. This conditional probability path $(p_t(\cdot \mid \boldsymbol{X}_1))_t$ is defined directly on the manifold $\mathcal{P}_d$ and relies on an infinitely supported distribution. In order to match the source distribution p at $t = 0$, we impose that the marginal $p_0 = p$. This marginal p_0 can be computed as $p_0 = \int p_0(\cdot \mid \boldsymbol{Z}) q(\boldsymbol{Z}) \mathrm{d}\boldsymbol{Z}$, which constrains jointly the conditional path $p_0(\cdot \mid \boldsymbol{Z})$ and the distribution q over the conditioning variable $\boldsymbol{Z}$.

Remark 5.1. Using Proposition 3.4, one can see that the spread of $\mathrm{WG}(\boldsymbol{I}_d, 0_D, \boldsymbol{I}_D)$ scales roughly like d. This can lead to some numerical issues when d is large. This is why, to solve this issue, we start from $p_0 = \mathrm{WG}(\boldsymbol{I}_d, 0_D, \sigma_{\text{init}}^2 \boldsymbol{I}_D)$ with $\sigma_{\text{init}} < 1$, thus a more concentrated source distribution. In the experiments, we will set $\sigma_{\text{init}} = \sqrt{\frac{2}{d(d+1)}}$. An ablation of this choice is provided in Appendix D.4.

We now need to build a conditional flow Φ_t (conditioned by $\boldsymbol{Z}$) such that $\Phi_t \# p_0 = p_t(\cdot \mid \boldsymbol{Z})$. Using the properties of the wrapped Gaussian given in Proposition 3.3, we have the following proposition (see Appendix B for the proof):

Proposition 5.2. *The following conditional flow Φ_t satisfies $\Phi_t \# p_0 = p_t(\cdot \mid \boldsymbol{Z})$ where $p_t(\cdot \mid \boldsymbol{Z})$ is given at Equation* (9)

$$\Phi_t(\boldsymbol{X}) = \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \exp\left[\frac{\sigma_t(\boldsymbol{Z})}{\sigma_{init}} \log(\boldsymbol{X}) + \mu_t(\boldsymbol{Z})\right] \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}. \tag{10}$$

This conditional flow Φ_t can be decomposed into two parts: the first one is inside the exp, where one retrieves the classical Euclidean flow between two Gaussians (up to the change of variables $\log(\boldsymbol{X})$) as given at Equation 11 of Lipman et al. (2023), this part affects the parameters of the Gaussian in the tangent space. Then, the second part is the outside of the exp, that controls and adapts the base point of the wrapped Gaussian defining the conditional probability path (see the first point of Proposition 3.3).

This flow then provides a vector field that generates the conditional probability path $(p_t(\cdot \mid \boldsymbol{Z}))_t$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(\boldsymbol{X}) = u_t(\Phi_t(\boldsymbol{X}) \mid \boldsymbol{Z}) \in T_{\Phi_t(\boldsymbol{X})}\mathcal{P}_d.$$

One can solve this ODE for u_t to get a closed form of the conditional vector field:

Theorem 5.3. *Let $p_t(\cdot \mid \boldsymbol{Z})$ be the conditional probability path given at Equation* (9) *and Φ_t its flow. The unique vector field defining Φ_t depends on the value of $\sigma_t(\boldsymbol{Z})$ in the following manner:*

Method	$q(\boldsymbol{Z})$	$\bar{\boldsymbol{X}}_t(\boldsymbol{Z})$	$\mu_t(\boldsymbol{Z})$	$\sigma_t(\boldsymbol{Z})$
RiemannianCFM	$p(\boldsymbol{X}_0)q_1(\boldsymbol{X}_1)$	$\gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t)$	0_D	0
DiffeoCFM	$q_1(\boldsymbol{X}_1)$	$\boldsymbol{I}_d$	$t\log(\boldsymbol{X}_1)$	σ_{lin}
WrappedCFM	$q_1(\boldsymbol{X}_1)$	$\boldsymbol{X}_1^t$	0_D	σ_{lin}

Table 1: Unified Wrapped-Gaussian framework for RiemannianCFM on $\mathcal{P}_d$. Each method corresponds to a specific parameterization of the conditional path $p_t(\cdot \mid \boldsymbol{Z}) = \text{WG}(\bar{\boldsymbol{X}}_t, \mu_t, \sigma_t^2 \boldsymbol{I}_D)$. We denote the linear scheduler $\sigma_{\text{lin}} = \sigma_{\text{init}} - (\sigma_{\text{init}} - \sigma_{\min})\, t$

- *If $\sigma_t(\boldsymbol{Z}) > 0$, then the unique vector field defining Φ_t is*

$$u_t(\boldsymbol{Y} \mid \boldsymbol{Z}) = \boldsymbol{S}(t)\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{Y} + \boldsymbol{Y}\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{S}(t) + \bar{\boldsymbol{X}}_t^{1/2}\, \mathrm{d}\exp(\log \boldsymbol{M}_t) \cdot \left[\frac{\sigma_t'}{\sigma_t}(\log \boldsymbol{M}_t - \mu_t) + \mu_t'\right] \bar{\boldsymbol{X}}_t^{1/2},$$

  where $\boldsymbol{S}$ is the derivative of $t \mapsto \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}$, $\boldsymbol{M}_t = \bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{Y}\bar{\boldsymbol{X}}_t^{-1/2}$, and all time-dependent quantities are evaluated at $\boldsymbol{Z}$.

- *If $\sigma_t(\boldsymbol{Z}) = 0$, then*

$$u_t(\boldsymbol{Y} \mid \boldsymbol{Z}) = \frac{\mathrm{d}}{\mathrm{d}t}\left[\bar{\boldsymbol{X}}_t^{1/2}\exp(\mu_t)\bar{\boldsymbol{X}}_t^{1/2}\right],$$

  where all time-dependent quantities are evaluated at $\boldsymbol{Z}$.

The proof can be found at Appendix B. Let us comment on this result. First, the distinction between the cases $\sigma_t(\boldsymbol{Z}) = 0$ and $\sigma_t(\boldsymbol{Z}) > 0$ is due to the fact that when $\sigma_t(\boldsymbol{Z}) = 0$, the conditional flow Φ_t defined in Equation (10) is not invertible. Second, the time derivative $\boldsymbol{S}$ of $t \mapsto \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}$ is the solution to the following Sylvester equation (see Theorem 1.26 or Chapter 6 of Higham 2008):

$$\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\, \boldsymbol{S}(t) + \boldsymbol{S}(t)\, \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} = \frac{\mathrm{d}}{\mathrm{d}t}\bar{\boldsymbol{X}}_t(\boldsymbol{Z}), \tag{11}$$

which can be numerically solved. However, for the specific choices of $\bar{\boldsymbol{X}}_t(\boldsymbol{Z})$ considered in the remainder of this work, $\boldsymbol{S}$ can be computed explicitly. Third, the direction of the exp differential is exactly the conditional vector field for the EuclideanCFM as given at Theorem 3 of Lipman et al. (2023). Finally, recall that $u_t(\cdot \mid \boldsymbol{Z})$ is a vector field on $\mathcal{P}_d$ and that $T_{\boldsymbol{X}}\mathcal{P}_d \cong \mathcal{S}_d$. One can verify that $u_t(\boldsymbol{X} \mid \boldsymbol{Z})$ is indeed symmetric.

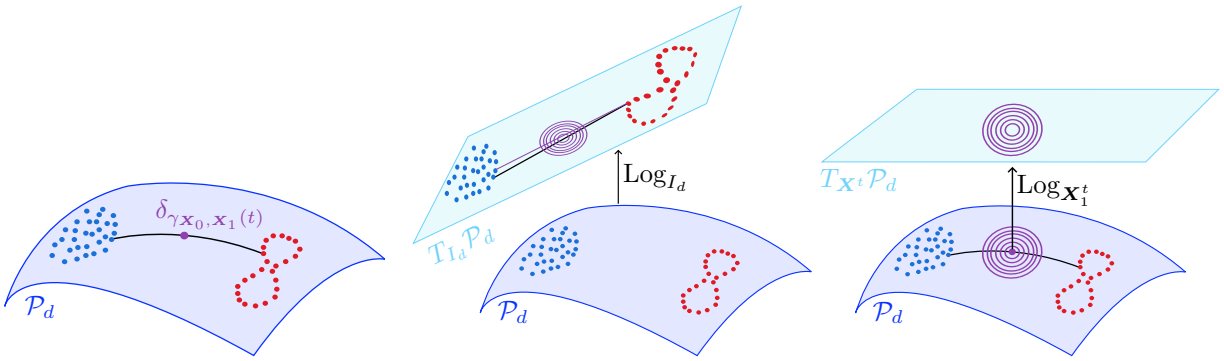
Remark 5.4. de Surrel et al. (2025) showed that the wrapped Gaussian distribution on $\mathcal{P}_d$ is not identifiable (see their Proposition 4.9); several distinct sets of parameters therefore define the same wrapped Gaussian. We show in Appendix C that equivalent choices of parameters for the conditional probability path in Equation (9) change neither the conditional flow given in Equation (10) nor the conditional vector field given in Theorem 5.3.

5.2 Specific parameterizations

The newly built general RiemannianCFM framework for the manifold $\mathcal{P}_d$ relies on the parameterization of the conditional probability paths $(p_t(\cdot \mid \boldsymbol{Z}))_t$ by the three following mappings: $\bar{\boldsymbol{X}}$, μ and σ and by the choice of the distribution over the conditioning variable $\boldsymbol{Z}$. We will now see that, depending on the choices of these parameterizations, we are able to retrieve the SOTA methods described in Section 4: the *RiemannianCFM* and the *DiffeoCFM*.

RiemannianCFM Let us start with the RiemannianCFM. In order to retrieve this method from our general framework, we chose the conditioning variable $\boldsymbol{Z}$ to be a pair of a random source point $\boldsymbol{X}_0 \sim p$ and a random target point $\boldsymbol{X}_1 \sim q_1$ thus, $\boldsymbol{Z} = (\boldsymbol{X}_0, \boldsymbol{X}_1)$ and $q(\boldsymbol{Z}) = p(\boldsymbol{X}_0)q_1(\boldsymbol{X}_1)$ is the independent coupling. Then, the choice for the parameterization of the conditional probability paths is:

$$\bar{\boldsymbol{X}}_t(\boldsymbol{Z}) = \gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t),\ \mu_t(\boldsymbol{Z}) = 0_D,\ \text{and}\ \sigma_t(\boldsymbol{Z}) = 0,$$



(a) RiemannianCFM. (b) DiffeoCFM. (c) WrappedCFM.

Figure 2: Illustration of the different CFM models on the manifold $\mathcal{P}_d$: RiemannianCFM that uses the geodesic interpolation between $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$, DiffeoCFM that relies on a mapping to the tangent space at identity I_d and then uses the Euclidean CFM, and finally, the proposed WrappedCFM that uses a wrapped Gaussian whose anchor moves along the geodesic between I_d and $\boldsymbol{X}_1$.

where we recall that $\gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}$ is the unique geodesic joining $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$. This is the degenerated case where the covariance of the wrapped Gaussian is zero, leading to a Dirac distribution on $\mathcal{P}_d$. It is similar to the Euclidean setting as described in Tong et al. (2024a) (in which it is referred to as "Basic form of CFM").

DiffeoCFM In order to get DiffeoCFM from our general framework, the conditioning variable $\boldsymbol{Z}$ is a single random point $\boldsymbol{X}_1$ from the target distribution q_1, and one chooses the following parameterization:

$$\bar{\boldsymbol{X}}_t(\boldsymbol{X}_1) = \boldsymbol{I}_d,\ \mu_t(\boldsymbol{X}_1) = t\log(\boldsymbol{X}_1),\ \text{and}\ \sigma_t(\boldsymbol{X}_1) = \sigma_{\text{init}} - (\sigma_{\text{init}} - \sigma_{\min})\,t.$$

Here, everything is sent to the tangent space $T_{\boldsymbol{I}_d}\mathcal{P}_d$ at identity, so the base point $\bar{\boldsymbol{X}}_t(\boldsymbol{X}_1)$ of the WG does not change and then classical CFM is applied in $T_{\boldsymbol{I}_d}\mathcal{P}_d$.

In Appendix B.2, we show how the general formulas given above recover, for each specific method, the correct expressions of the conditional flow Φ_t and the conditional vector field $u_t(\cdot \mid \boldsymbol{X}_1)$. We summarize the different parameterizations in Table 1.

6 Wrapped Conditional Flow Matching

After introducing a general framework for CFM on the Riemannian manifold $\mathcal{P}_d$ of SPD matrices and showing how SOTA methods can be recovered through specific parameterizations, we introduce in this section a new parameterization that leads to a novel RiemannianCFM model. We call this method *WrappedCFM*. This WrappedCFM relies on a conditional probability path that is denoted $(p_t^{\text{WCFM}})_{0\leqslant t\leqslant 1}$. Like DiffeoCFM, the conditioning variable $\boldsymbol{Z}$ is simply a random point $\boldsymbol{X}_1$ from the target distribution q_1. Then, $(p_t^{\text{WCFM}})_t$ is built using the following parameterization of the general conditional probability paths given at Equation (9):

$$\bar{\boldsymbol{X}}_t(\boldsymbol{X}_1) = \boldsymbol{X}_1^t,\ \mu_t(\boldsymbol{X}_1) = 0_D,\ \text{and}\ \sigma_t(\boldsymbol{X}_1) = (1-t)\,\sigma_{\text{init}} + t\,\sigma_{\min}, \tag{12}$$

where $\sigma_{\min} \ll \sigma_{\text{init}}$. Therefore, the conditional probability path at time $t = 0$ is $p_0^{\text{WCFM}}(\cdot \mid \boldsymbol{X}_1) = \text{WG}(\boldsymbol{I}_d, 0_D, \sigma_{\text{init}}^2 \boldsymbol{I}_D) = p$, and at time $t = 1$, is a concentrated WG around $\boldsymbol{X}_1$. This conditional probability path $(p_t^{\text{WCFM}})_{0\leqslant t\leqslant 1}$ is the equivalent of the classical Gaussian conditional probability paths from Lipman et al. (2023) but defined on $\mathcal{P}_d$ using wrapped Gaussians. Remark that $t \mapsto \bar{\boldsymbol{X}}_t(\boldsymbol{X}_1)$ is the unique geodesic between $\boldsymbol{I}_d$ and $\boldsymbol{X}_1$ under the AIRM metric given at Equation (1).

The differences between the proposed WrappedCFM and RiemannianCFM is that the WrappedCFM relies on an infinitely supported distribution, whereas the RiemannianCFM relies on Dirac distributions (recall that RiemannianCFM is a degenerate case of our framework, when $\sigma_t(\boldsymbol{X}_1) = 0$). The difference between

the DiffeoCFM and the WrappedCFM is that the DiffeoCFM uses only *one* tangent space to project the data onto. This tangent space $T_{\boldsymbol{I}_d}\mathcal{P}_d$ is a good local approximation of the manifold $\mathcal{P}_d$, but if the target distribution q_1 is very spread out, relying only on $T_{\boldsymbol{I}_d}\mathcal{P}_d$ can induce some deformations. In our proposed WrappedCFM, as the center $\bar{\boldsymbol{X}}_t(\boldsymbol{X}_1)$ of the wrapped Gaussian modeling the conditional probability path $p_t(\cdot \mid \boldsymbol{X}_1)$ changes at each time step t, it provides a better approximation of the curved space $\mathcal{P}_d$. The three methods are illustrated in Figure 2

We now need to build a conditional flow Φ_t (conditioned by $\boldsymbol{X}_1$) such that $\Phi_t \# p_0^{\mathrm{WCFM}} = p_t^{\mathrm{WCFM}}(\cdot \mid \boldsymbol{X}_1)$. Using the general flow given at Equation (10) and the specific parameterization given at Equation (12), we have that the flow Φ_t^{WCFM} is:

$$\Phi_t^{\mathrm{WCFM}}(\boldsymbol{X}) = \boldsymbol{X}_1^{t/2} \boldsymbol{X}^{\frac{\sigma_t(\boldsymbol{X}_1)}{\sigma_{\mathrm{init}}}} \boldsymbol{X}_1^{t/2}. \tag{13}$$

In this case, σ_t is invertible, so one can apply the first case of Theorem 5.3 to get the conditional vector field that generates $p_t^{\mathrm{WCFM}}(\cdot \mid \boldsymbol{X}_1)$:

Theorem 6.1. *Let $p_t^{WCFM}(\cdot \mid \boldsymbol{X}_1)$ be the conditional probability path given by the WrappedCFM and Φ_t^{WCFM} its corresponding flow map as in Equation* (13)*. Then, the unique vector field that defines Φ_t^{WCFM} has the form:*

$$u_t^{WCFM}(\boldsymbol{Y} \mid \boldsymbol{X}_1) = \tfrac{1}{2}\log(\boldsymbol{X}_1)\,\boldsymbol{Y} + \tfrac{1}{2}\boldsymbol{Y}\log(\boldsymbol{X}_1) + \tfrac{\sigma_t'(\boldsymbol{X}_1)}{\sigma_t(\boldsymbol{X}_1)}\,\boldsymbol{X}_1^{t/2}\boldsymbol{M}_t^{1/2}\log(\boldsymbol{M}_t)\boldsymbol{M}_t^{1/2}\boldsymbol{X}_1^{t/2},$$

where $\boldsymbol{M}_t = \boldsymbol{X}_1^{-t/2}\boldsymbol{Y}\boldsymbol{X}_1^{-t/2}$. Consequently, $u_t(\cdot \mid \boldsymbol{X}_1)$ generates the wrapped Gaussian path $p_t(\cdot \mid \boldsymbol{X}_1)$.

The proof is given at Appendix B.2. As stated after Theorem 5.3, we know explicitly how to compute the derivative $\boldsymbol{S}$ of $t \mapsto \bar{\boldsymbol{X}}_t(\boldsymbol{X}_1) = \boldsymbol{X}_1^t$ using Lemma B.1. Thus, we do not need to solve numerically the Sylvester equation given at Equation (11) as in the general case.

Finally, the CFM objective is adapted from Chen & Lipman (2024) to the Riemannian setting of $\mathcal{P}_d$:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t,\boldsymbol{X}_1,\boldsymbol{X}} \|v_t^\theta(\boldsymbol{X}) - u_t^{\mathrm{WCFM}}(\boldsymbol{X} \mid \boldsymbol{X}_1)\|_{\boldsymbol{X}}^2. \tag{14}$$

where $t \sim \mathcal{U}(0,1)$, $\boldsymbol{X}_1 \sim q_1$, $\boldsymbol{X} \sim p_t^{\mathrm{WCFM}}(\cdot \mid \boldsymbol{X}_1)$ and $\|\cdot\|_{\boldsymbol{X}}$ is the norm induced by the AIRM given at Equation (1) in the tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$.

Once the vector field u_t is learned using v_t^θ, to generate a sample $\tilde{\boldsymbol{X}}_1 \sim q_1$, one draws an initial sample $\boldsymbol{X}_0 \sim p_0^{\mathrm{WCFM}} = \mathrm{WG}(\boldsymbol{I}_d, 0_D, \sigma_{\mathrm{init}}^2 \boldsymbol{I}_D)$ and integrating the learned vector field v_t^θ from $t=0$ to $t=1$ using a Riemannian Euler scheme:

$$\boldsymbol{X}_{k+1} = \mathrm{Exp}_{\boldsymbol{X}_k}(h v_{t_k}^\theta(\boldsymbol{X}_k)) \tag{15}$$

where h is a time step usually set to $\frac{1}{100}$ in the experiments and $t_k = kh$.

7 Synthetic experiments

Now that we have introduced a global framework for the different CFM methods using wrapped Gaussians, and proposed a new method for doing CFM on $\mathcal{P}_d$, namely *WrappedCFM*, we want to compare the different methods numerically[1]. We start with synthetic experiments to compare them. Let us start by detailing the setup of this experiment, including the contestants, the dataset description as well as the metric used.

The contestants In the following experiments, we will compare our proposed method, WrappedCFM, with the other RiemannianCFM methods that were introduced in Section 4.2: *RiemannianCFM* and *DiffeoCFM*. We will also compare these three Riemannian approaches to a classical EuclideanCFM. In this case, the SPD matrices are treated as Euclidean symmetric matrices by vectorizing their upper-triangular part, and the Riemannian geometry is not taken into account.

All of the methods rely on a deep network to learn v_t^θ, the architectures of which are detailed in Appendix D.

[1]Our code is available at `https://github.com/thibaultdesurrel/Wrapped_Flow_Matching_SPD`.

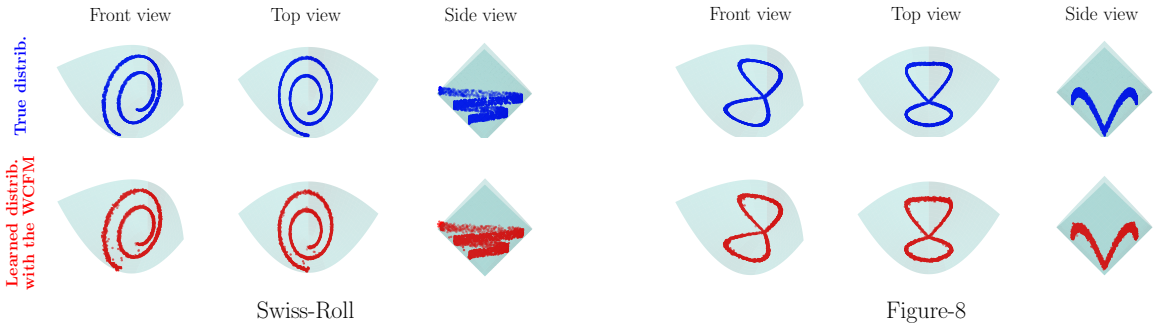


Figure 3: Two examples of the result of the proposed WrappedCFM once the flow is learned. The true target distribution is on the top row and the learned one on the bottom row. In these examples (described in Section 7), each point represents a 2×2 SPD matrix, that can thus be represented in 3D. We can see that, qualitatively, the proposed WCFM is able to learn the correct distributions.

Datasets In this synthetic experiment, we consider two different datasets that each serve different purposes:

1. We first consider two 2×2 datasets. As the matrices are 2×2 SPD matrices, they are symmetric and thus of the form $\begin{pmatrix} a & b \\ b & c \end{pmatrix}$ and one can plot them in 3D by plotting the tuple (a, b, c). These datasets are inspired by the ones used in Zhou et al. (2025), and we call them *Swiss-Roll* and *Figure-8*. The goal of these two datasets is to see qualitatively if the WrappedCFM produces coherent samples as one can plot them in 3D.
2. The second dataset is composed of 10×10 SPD matrices sampled from a mixture of 3 wrapped Gaussians $\mathrm{WG}(\boldsymbol{X}_k, 0_{55}, \sigma_{\text{init}}^2 I_{55})$ for $k = 1, 2, 3$. The goal of this dataset is to have a simple target, but controlling the condition number κ of the class centers $\boldsymbol{X}_k$. For each experiment, a unique $\kappa \in \{10, 10^2, 10^3, 10^4\}$ is chosen, and these class centers $\boldsymbol{X}_k$ are built so as to have condition number exactly κ. We want to see how the performance of the different FM methods evolves when the conditioning of the dataset worsens. The condition number measures how "stretched" or anisotropic the matrix is (Goldstine & Von Neumann, 1951). When this condition number is very large compared to one, the matrix $\boldsymbol{X}$ is said to be *ill-conditioned*, and the computation of its inverse, or solution of a linear system of equations is prone to large numerical errors.

More details on how the different datasets are generated are given at Appendix D.

The metrics Let us now detail the different metrics we will use to evaluate the different models. The first metric used in this synthetic experiment will be **the number of sampled matrices that are SPD**. Indeed, our main goal is to build CFM methods that respect the geometry of $\mathcal{P}_d$, so we want the sampled points to be SPD. The three Riemannian methods are built to ensure the SPDness of the sampled matrices, but it is not the case for the EuclideanCFM. Then, the main metric used to evaluate the different CFM methods is the **Wasserstein-2 distance** between the points sampled and the true test distribution. We recall that the Wasserstein-2 distance between the true sample distribution q_1 and the learned one $\tilde{q}_1$ is defined as follows (Peyré & Cuturi, 2020):

$$W_2(q_1, \tilde{q}_1)^2 = \inf_{\pi \in \Pi(q_1, \tilde{q}_1)} \mathbb{E}_{(x,y)\sim\pi} \delta(x, y)^2,$$

where $\Pi(q_1, \tilde{q}_1)$ is the set of couplings of q_1 and $\tilde{q}_1$. It was computed using POT (Flamary et al., 2024).

7.1 The results

We start by giving some qualitative results for the 2×2 datasets *Swiss-Roll* and *Figure-8* by plotting the true and learned distributions at Figure 3. We can see that the proposed WrappedCFM method is able to

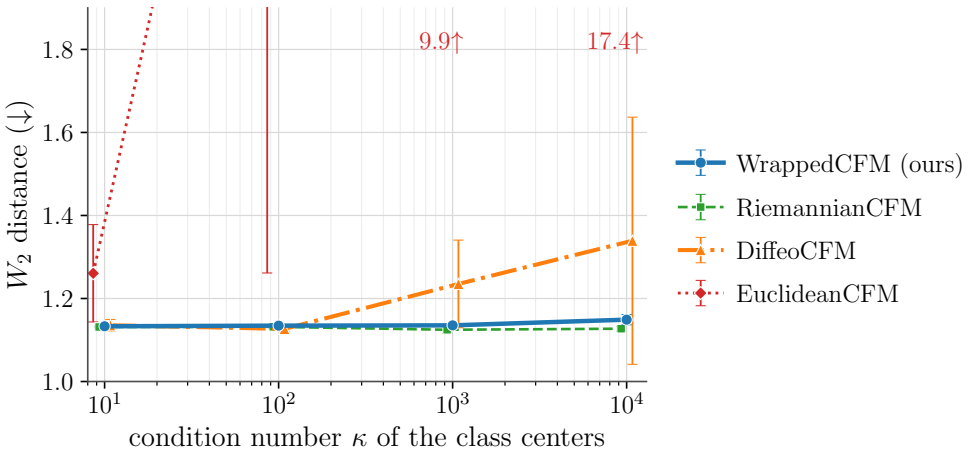


Figure 4: W_2-distance (↓, log-log) as a function of the condition number κ of the class centers on the *WG Mixture* dataset. WrappedCFM and RiemannianCFM stay flat and tight across the whole range of κ. DiffeoCFM, by contrast, degrades monotonically as κ grows. EuclideanCFM fares far worse still, collapsing by more than an order of magnitude at $\kappa = 10^4$.

Dataset	EuclideanCFM	RiemannianCFM	DiffeoCFM	WrappedCFM
Swiss-Roll	0.375 ± 0.273	0.291 ± 0.021	0.266 ± 0.015	$\mathbf{0.265 \pm 0.015}$
Figure-8	0.310 ± 0.157	0.219 ± 0.021	0.206 ± 0.016	$\mathbf{0.201 \pm 0.020}$

Table 2: W_2 distance between the sampled distribution using the different CFM methods and the true distribution for the 2×2 synthetic datasets (Swiss-Roll and Figure-8). Results are reported as mean $\pm$ standard deviation over 5 random seeds. Lower is better. Best results are bold.

correctly reproduce the true datasets. The learned distributions using our WrappedCFM are noisier than the true distributions, and one can see some outliers. However, the global shape of the two true datasets is well reproduced, which validates our approach. For the 2×2 datasets, we also give the quantitative results of the W_2 distance for the different models at Table 2. We see that the proposed WrappedCFM achieves the lowest W_2. For the EuclideanCFM, for these datasets, this method was able to produce 99.9% of SPD matrices. The datasets being low-dimensional and well conditioned, this is not surprising.

For the experiment on the dataset composed of mixture of wrapped Gaussians, let us first focus on the percentage of generated samples that are SPD. For the methods taking into account the geometry (RiemannianCFM, DiffeoCFM and WrappedCFM), 100% of the generated samples are SPD. Similarly, for the EuclideanCFM, when $\kappa = 10$ (the conditioning number of the centers of the mixture components), 99.9% of generated samples are SPD. However, when κ grows, the percentage of SPD samples significantly drops: for $\kappa = 10^2$, we have 96.0% of SPD matrices, for $\kappa = 10^3$, only 61.2% were SPD matrices and when $\kappa = 10^4$, only 1.8% of generated samples are SPD. This shows that taking into account the geometry is crucial. We analyze the spectrum of the generated non-SPD matrices in Appendix D.3. When the generated samples were not SPD, we projected them onto $\mathcal{P}_d$ using a convex shrinkage toward the identity, calibrated so that the smallest eigenvalue after shrinkage is exactly 10^{-8} (Ledoit & Wolf, 2004). Then, we computed the W_2 distance to the true distribution. The results are given at Figure 4. We can see that the W_2 distance for the EuclideanCFM explodes when κ grows, showing that projecting non-SPD samples on the cone $\mathcal{P}_d$ is not a good solution. For small conditioning number κ, the geometry-aware methods behave all the same, while when κ grows, one can see that DiffeoCFM starts to perform worse than RiemannianCFM and WrappedCFM with a +18% W_2 from $\kappa = 10$ to $\kappa = 10^4$, together with a larger seed-to-seed variance. This can be explained because its fixed-tangent-chart interpolation grows increasingly anisotropic.

Dataset	DiffeoCFM	EuclideanCFM	RiemannianCFM	WrappedCFM
Training time (minutes), mean ± std				
Figure-8	23.2 ± 0.4	$\mathbf{18.3} \pm 1.5$	24.2 ± 0.2	26.8 ± 1.1
Swiss-roll	21.6 ± 1.4	$\mathbf{17.6} \pm 0.5$	24.2 ± 0.4	24.7 ± 0.2
WG mixture $\kappa = 10$	91.0 ± 8.9	$\mathbf{78.3} \pm 5.0$	104.4 ± 3.6	115.8 ± 7.3
WG mixture $\kappa = 10^2$	94.2 ± 9.8	$\mathbf{75.8} \pm 4.5$	108.1 ± 5.0	120.6 ± 5.6
WG mixture $\kappa = 10^3$	$\mathbf{46.8} \pm 13.6$	53.3 ± 4.6	59.7 ± 4.5	63.8 ± 4.7
WG mixture $\kappa = 10^4$	62.1 ± 13.6	$\mathbf{50.2} \pm 0.2$	63.3 ± 4.5	66.3 ± 4.8
Sampling time, 1000 samples (seconds)				
2×2 datasets	0.12	**0.11**	0.28	0.28
WG mixture (all κ)	1.27	**1.26**	2.4–2.5	2.4–2.5

Table 3: Training and sampling times (mean over 5 seeds) on the different synthetic datasets.

7.2 Training and sampling times

We report the training and sampling times of all methods across the synthetic datasets in Table 3. Unsurprisingly, EuclideanCFM is the quickest method to train and to sample from: it ignores the Riemannian geometry of $\mathcal{P}_d$ and relies only on Euclidean transformations. DiffeoCFM is generally the next quickest to train, as it mostly uses Euclidean transformations in the tangent space at $\boldsymbol{I}_d$, with a few additional Riemannian exponential and logarithm evaluations. Relative to RiemannianCFM, the proposed WrappedCFM adds only 2–12% training time, an increase mainly due to its more complex conditional vector field; this remains reasonable. More importantly, there is no measurable difference in sampling time between RiemannianCFM and WrappedCFM: both use the same procedure, the Riemannian Euler scheme described in Equation (15). This scheme is expectedly slower than the Euclidean Euler scheme used by EuclideanCFM and DiffeoCFM, since it performs one $\mathrm{Exp}_{\boldsymbol{X}_k}$ per integration step. In absolute terms, no method takes more than about 2.5 s to sample 1000 matrices.

We also investigate the time taken by one training step as well as by the sampling when the dimension grows in Appendix E.

8 Real-world data experiments

8.1 The setup

The datasets We evaluate our proposed WrappedCFM framework on 4 real-world datasets from different downstream applications: BNCI2014002, BNCI2015001 (Leeb et al., 2007) from Brain Computer Interfaces (BCI), Textile (Bergmann et al., 2021) for anomaly detection for textile manufacturing, and BreizhCrops Rußwurm et al. (2020), a satellite image time series classification dataset. All these datasets include time series from which we computed covariance matrices. Details on preprocessing are presented in Appendix F.1 and the datasets are summarized at Table 7.

Evaluation procedure After training, each model generates a set of synthetic SPD matrices conditioned on the class labels of both the training and test splits. We measure the quality of the synthetic distribution through two complementary lenses.

1. First, we measure **distribution quality metrics**: *Density*, *Coverage* from Naeem et al. (2020) and *Wasserstein-2 distance* between real and synthetic data. The *Density* evaluates the fidelity of the data, that is, how realistic the synthetic data is while the *Coverage* evaluates the diversity of the data, to see how well the synthetic data captures the variations from the real data. According to Naeem et al. (2020), these metrics improve over the classical *Precision/Recall* from Kynkäänniemi et al. (2019). We use the code provided by Naeem et al. (2020) to compute these metrics. These are reported for "Train vs. Generated Train", "Test vs. Generated Test", and we also provide "Train vs. Test" metrics on real data for comparison.

	Train / Test			Train / Generated			Test / Generated		
Model	Dens. ↑	Cov. ↑	W2 ↓	Dens. ↑	Cov. ↑	W2 ↓	Dens. ↑	Cov. ↑	W2 ↓
BNCI2014002									
Real data	0.943±0.045	0.892±0.054	2.186±0.061						
EuclideanCFM				0.018±0.022	0.105±0.107	13.894±2.299	0.021±0.020	0.082±0.070	13.804±2.584
DiffeoCFM				0.661±0.047	0.969±0.011	2.010±0.027	**0.411**±0.041	0.721±0.052	2.421±0.061
RiemannianCFM				0.910±0.114	0.992±0.004	1.916±0.022	0.390±0.040	0.795±0.045	2.346±0.057
WrappedCFM				**0.915**±0.088	**0.994**±0.004	**1.910**±0.026	0.403±0.039	**0.801**±0.045	**2.327**±0.062
BNCI2015001									
Real data	0.049±0.002	0.130±0.003	2.073±0.005						
EuclideanCFM				0.058±0.037	0.257±0.125	6.023±0.851	0.006±0.000	0.019±0.010	6.084±0.985
DiffeoCFM				0.270±0.021	0.817±0.019	1.674±0.012	0.028±0.004	0.080±0.002	2.095±0.008
RiemannianCFM				0.732±0.113	0.972±0.018	1.600±0.037	**0.069**±0.001	**0.161**±0.008	**2.019**±0.007
WrappedCFM				**0.749**±0.092	**0.975**±0.014	**1.584**±0.036	0.063±0.004	0.155±0.026	2.038±0.017
Textile									
Real data	0.974±0.025	0.999±0.000	0.723±0.001						
EuclideanCFM				0.381±0.032	0.928±0.018	1.448±0.060	0.464±0.041	0.957±0.016	1.395±0.155
DiffeoCFM				0.757±0.030	0.991±0.002	0.697±0.002	0.834±0.035	0.996±0.001	0.757±0.002
RiemannianCFM				**0.962**±0.017	**0.997**±0.001	**0.678**±0.001	**1.005**±0.024	**0.999**±0.001	**0.742**±0.003
WrappedCFM				0.934±0.017	**0.997**±0.001	0.679±0.002	0.978±0.031	0.998±0.002	0.744±0.002
BreizhCrops									
Real data	0.965±0.012	0.969±0.006	0.684±0.002						
EuclideanCFM				0.372±0.016	0.925±0.005	1.270±0.078	0.216±0.008	0.626±0.011	1.235±0.101
DiffeoCFM				0.875±0.015	0.995±0.001	0.621±0.001	0.524±0.010	0.853±0.011	0.685±0.003
RiemannianCFM				**0.963**±0.015	0.997±0.001	**0.614**±0.001	**0.585**±0.009	**0.873**±0.009	**0.681**±0.003
WrappedCFM				0.939±0.006	**0.998**±0.001	**0.614**±0.001	0.573±0.011	0.869±0.007	0.682±0.002

Table 4: Distribution quality of generated SPD matrices, measured between train, test and generated sets. Density (Dens.) and coverage (Cov.): higher is better; Wasserstein-2 distance (W2): lower is better. Best generated result per dataset and column in **bold** (based on the mean only).

2. Second, we assess **downstream utility** via a classifier. The classifier is a logistic regression classifier in the tangent space at the Fréchet mean, under three regimes:
   (a) **T**rained on **R**eal data and **T**ested on **R**eal data (**TRTR**), serving as the upper bound and a control, captures the intrinsic task difficulty.
   (b) **T**rained on **S**ynthetic data and **T**ested on **R**eal data (**TSTR**). It measures to what point can synthetic data replace real data for learning.
   (c) **T**rained on **R**eal data and **T**ested on **S**ynthetic data (**TRTS**). The question asked here is: are synthetic samples recognizable as their conditioning class?

To ensure a fair comparison with the competing methods, we use the same evaluation pipeline (based on a logistic classifier in the tangent space) as in Collas et al. (2025) and detail it in Appendix F.2.

8.2 The results

We report the distribution quality results in Table 4 and the downstream classification results in Table 5. Riemannian methods substantially outperform EuclideanCFM across all datasets on both evaluation protocols demonstrating the importance of manifold-aware methods. This trend is reflected in both the distributional and downstream evaluation metrics. EuclideanCFM produces larger W_2 distances and lower density scores. For example, on BNCI2014002, EuclideanCFM attains a W_2 distance of 13.894 compared to 2.010 for the Riemannian methods, while its density score decreases from 0.661 to 0.018. These deficiencies are also reflected in downstream classification performance, where EuclideanCFM achieves an F1 score of 0.457 on BNCI2014002 compared to 0.718 for the Riemannian approaches. Among the Riemannian approaches, WrappedCFM and RiemannianCFM exhibit similar performance and both outperform DiffeoCFM in terms of sample quality. For example, on BNCI2015001, DiffeoCFM attains a Train/Generated density of 0.270 compared to 0.732 and 0.749 for RiemannianCFM and WrappedCFM, respectively, while on Textile the corresponding scores are 0.757 versus 0.962 and 0.934. Overall, these results demonstrate that modeling the SPD geometry is essential for generative modeling, and that the proposed WrappedCFM achieves perfor-

	TRTR		TSTR		TRTS	
Model	ROC-AUC ↑	F1 ↑	ROC-AUC ↑	F1 ↑	ROC-AUC ↑	F1 ↑
BNCI2014002						
Real data	0.827±0.032	0.734±0.038				
EuclideanCFM			0.712±0.036	0.457±0.297	0.657±0.053	0.682±0.055
DiffeoCFM			0.808±0.025	0.720±0.021	0.871±0.018	0.788±0.027
RiemannianCFM			0.810±0.024	0.718±0.029	0.888±0.032	0.807±0.040
WrappedCFM			**0.818**±0.030	**0.729**±0.033	**0.893**±0.023	**0.818**±0.026
BNCI2015001						
Real data	0.734±0.014	0.675±0.037				
EuclideanCFM			0.702±0.025	0.655±0.013	0.731±0.018	0.691±0.001
DiffeoCFM			0.719±0.017	**0.677**±0.032	0.829±0.008	0.757±0.006
RiemannianCFM			**0.722**±0.018	0.673±0.044	**0.860**±0.013	**0.781**±0.006
WrappedCFM			0.714±0.016	0.651±0.044	0.853±0.013	0.771±0.016
Textile						
Real data	0.975±0.002	0.922±0.004				
EuclideanCFM			0.968±0.003	0.908±0.009	0.945±0.007	0.875±0.016
DiffeoCFM			0.972±0.003	0.917±0.005	0.966±0.004	0.900±0.013
RiemannianCFM			**0.973**±0.002	**0.919**±0.004	**0.976**±0.003	**0.919**±0.010
WrappedCFM			**0.973**±0.002	**0.919**±0.002	0.975±0.003	0.913±0.012
BreizhCrops						
Real data	0.977±0.000	0.853±0.001				
EuclideanCFM			0.962±0.001	0.778±0.002	0.863±0.039	0.605±0.054
DiffeoCFM			0.974±0.000	0.825±0.003	0.970±0.005	0.803±0.018
RiemannianCFM			**0.975**±0.000	**0.829**±0.002	**0.984**±0.002	**0.856**±0.010
WrappedCFM			**0.975**±0.000	0.826±0.001	0.981±0.003	0.845±0.015

Table 5: Downstream classification performance of classifiers trained and tested on real or generated SPD matrices (TRTR: train real/test real; TSTR: train synthetic/test real; TRTS: train real/test synthetic). Best result per dataset and column in **bold** (based on the mean only).

mance comparable to RiemannianCFM while consistently improving over DiffeoCFM on distribution quality metrics.

9 Conclusion

In this paper, we introduced *WrappedCFM*, a novel framework for Conditional Flow Matching to the Riemannian manifold $\mathcal{P}_d$ of SPD matrices. By leveraging the Wrapped Gaussian distribution, our approach provides a unified perspective that encompasses existing SOTA RiemannianCFM methods as special cases while bypassing the need for extrinsic approximations or restrictive pullbacks. This unifying framework for learning tools on the manifold $\mathcal{P}_d$ of SPD matrices has started to be discussed in de Surrel et al. (2025). In the latter, the authors focus on parametric tools such as classifiers, while here, we extend this framework to Flow Matching tools on $\mathcal{P}_d$. Leveraging this framework, we introduce a novel interpolation scheme validated through extensive experiments on both synthetic and real-world datasets. Our results demonstrate that geometry-aware, infinitely supported conditional paths are able to sample diverse samples while maintaining high fidelity. A key limitation remains the computational cost of repeated matrix exponential and logarithm evaluations, particularly in high dimensions. Future work will focus on addressing this scalability and extending our Wrapped Gaussian framework to broader classes of Hadamard manifolds Shiga (1984) as wrapped Gaussians can be defined in a similar manner. We will also explore new parameterizations of our general probability paths, inspired by recent advances in Euclidean flow-based models (Albergo et al., 2023; Liu et al., 2022; Pooladian et al., 2023; Tong et al., 2024b), to generalize other CFM methods to the Riemannian manifold $\mathcal{P}_d$.

Acknowledgments

Thibault de Surrel was funded by the French National Research Agency for the project PROTEUS (grant ANR-22-CE33-0015-01). Aimi Okabayashi was also funded by the French Agence Nationale de la Recherche for the project MONI-TREE (ANR-23-CE04-0017). This work was granted access to the HPC resources of IDRIS under the allocation AD011015868R1 made by GENCI. Nicolas Courty acknowledges the support of PEPR IA Macleod project ANR-25-PEIA-0005 and contributes to AI Excellence Cluster SequoIA ANR-23-IACL-0009. The authors would like to thank Florian Yger for the insightful discussions regarding this project, in particular for the experimental part.

References

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Bruno Aristimunha, Igor Carrara, Pierre Guetschel, Sara Sedlar, Pedro Rodrigues, Jan Sosulski, Divyesh Narayanan, Erik Bjareholt, Quentin Barthelemy, Robin Tibor Schirrmeister, Reinmar Kobler, Emmanuel Kalunga, Ludovic Darmet, Cattan Gregoire, Ali Abdul Hussain, Ramiro Gatti, Vladislav Goncharenko, Anton Andreev, Jordy Thielen, Thomas Moreau, Yannick Roy, Vinay Jayaram, Alexandre Barachant, and Sylvain Chevallier. Mother of all bci benchmarks, 2025. URL `https://github.com/NeuroTechX/moabb`.

Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM journal on matrix analysis and applications*, 29(1):328–347, 2007.

Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. The mvtec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection. *Int. J. Comput. Vision*, 129(4):1038–1059, April 2021. ISSN 0920-5691. doi: 10.1007/s11263-020-01400-4. URL `https://doi.org/10.1007/s11263-020-01400-4`.

Rajendra Bhatia. *Positive Definite Matrices*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, N.J, 2007. ISBN 978-0-691-12918-1.

Rajendra Bhatia, Tanvi Jain, and Yongdo Lim. On the bures–wasserstein distance between positive definite matrices. *Expositiones mathematicae*, 37(2):165–191, 2019.

Avishek Joey Bose, Tara Akhound-Sadegh, Kilian Fatras, Guillaume Huguet, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael M. Bronstein, and Alexander Tong. Se(3)-stochastic flow matching for protein backbone generation. *CoRR*, abs/2310.02391, 2023. URL `https://doi.org/10.48550/arXiv.2310.02391`.

Max Braun, Noémie Jaquier, Leonel Rozo, and Tamim Asfour. Riemannian flow matching policy for robot motion learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5144–5151. IEEE, 2024.

Daniel A. Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Exploring complex time-series representations for riemannian machine learning of radar data. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3672–3676, 2019. doi: 10.1109/ICASSP.2019.8683056.

Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=g7ohDlTITL`.

Yilun Chen, Ami Wiesel, Yonina C Eldar, and Alfred O Hero. Shrinkage algorithms for mmse covariance estimation. *IEEE transactions on signal processing*, 58(10):5016–5029, 2010.

Emmanuel Chevallier, Didong Li, Yulong Lu, and David Dunson. Exponential-Wrapped Distributions on Symmetric Spaces. *SIAM Journal on Mathematics of Data Science*, 4(4):1347–1368, December 2022. ISSN 2577-0187. doi: 10.1137/21M1461551.

Sylvain Chevallier, Emmanuel K Kalunga, Quentin Barthélemy, and Eric Monacelli. Review of riemannian distances and divergences, applied to ssvep-based bci. *Neuroinformatics*, 19(1):93–106, 2021.

Antoine Collas, Ce Ju, Nicolas Salvy, and Bertrand Thirion. Riemannian Flow Matching for Brain Connectivity Matrices via Pullback Geometry, May 2025.

Philip J. Davis. Leonhard euler's integral: A historical profile of the gamma function: In memoriam: Milton abramowitz. *The American Mathematical Monthly*, 66(10):849–869, 1959. ISSN 00029890, 19300972. URL http://www.jstor.org/stable/2309786.

Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. *Advances in neural information processing systems*, 35:2406–2422, 2022.

Thibault de Surrel, Fabien Lotte, Sylvain Chevallier, and Florian Yger. Wrapped gaussian on the manifold of symmetric positive definite matrices. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=EhStXG4dCS.

Thibault de Surrel, Florian Yger, Fabien Lotte, and Sylvain Chevallier. A probabilistic view on riemannian machine learning models for spd matrices. In *International Conference on Geometric Science of Information*, pp. 142–151. Springer, 2025.

Ian L Dryden, Alexey Koloydenko, and Diwei Zhou. Non-euclidean statistics for covariance matrices, with applications to diffusion tensor imaging. *The Annals of Applied Statistics*, pp. 1102–1123, 2009.

Merran Gael Anderson Evans, Nicholas James Hastings, and Brian Peacock. Statistical distributions, third edition. *Measurement Science and Technology*, 12:117–117, 2001. URL https://api.semanticscholar.org/CorpusID:120262893.

Josef Faller, Carmen Vidaurre, Teodoro Solis-Escalante, Christa Neuper, and Reinhold Scherer. Autocalibration and recurrent adaptation: Towards a plug and play online erd-bci. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 20:313–9, 04 2012. doi: 10.1109/TNSRE.2012.2189584.

Rémi Flamary, Cédric Vincent-Cuaz, Nicolas Courty, Alexandre Gramfort, Oleksii Kachaiev, Huy Quang Tran, Laurène David, Clément Bonet, Nathan Cassereau, Théo Gnassounou, Eloi Tanguy, Julie Delon, Antoine Collas, Sonia Mazelet, Laetitia Chapel, Tanguy Kerdoncuff, Xizheng Yu, Matthew Feickert, Paul Krzakala, Tianlin Liu, and Eduardo Fernandes Montesuma. Pot python optimal transport (version 0.9.5), 2024. URL https://github.com/PythonOT/POT.

Herman H Goldstine and John Von Neumann. Numerical inverting of matrices of high order. ii. *Proceedings of the American Mathematical Society*, 2(2):188–202, 1951.

Gene H. Golub and Charles F. van Loan. *Matrix Computations*. JHU Press, fourth edition, 2013. ISBN 1421407949 9781421407944. URL http://www.cs.cornell.edu/cv/GVL4/golubandvanloan.htm.

Nicolas Guigui, Nina Miolane, Xavier Pennec, et al. Introduction to riemannian geometry and geometric statistics: from basic theory to implementation with geomstats. *Foundations and Trends® in Machine Learning*, 16(3):329–493, 2023.

Nicholas J. Higham. *Functions of Matrices*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, January 2008. ISBN 978-0-89871-646-7. doi: 10.1137/1.9780898717778.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Chin-Wei Huang, Milad Aghajohari, Joey Bose, Prakash Panangaden, and Aaron C Courville. Riemannian diffusion models. *Advances in Neural Information Processing Systems*, 35:2750–2761, 2022.

Jaehyeong Jo and Sung Ju Hwang. Generative modeling on manifolds through mixture of riemannian diffusion processes. *arXiv preprint arXiv:2310.07216*, 2023.

Chaeyoung Jung, Suyeon Lee, Ji Hoon Kim, and Joon Son Chung. Flowavse: Efficient audio-visual speech enhancement with conditional flow matching. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 2210–2214, 2024.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, February 2004. ISSN 0047259X. doi: 10.1016/S0047-259X(03)00096-4.

Robert Leeb, Felix Lee, Claudia Keinrath, Reinhold Scherer, Horst Bischof, and Gert Pfurtscheller. Brain–computer communication: Motivation, aim, and impact of exploring a virtual apartment. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 15(4):473–482, 2007.

Yunchen Li, Zhou Yu, Gaoqi He, Yunhang Shen, Ke Li, Xing Sun, and Shaohui Lin. SPD-DDPM: Denoising Diffusion Probabilistic Models in the Symmetric Positive Definite Space, December 2023.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling, February 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, and F. Yger. A review of classification algorithms for EEG-based brain–computer interfaces: A 10 year update. *Journal of Neural Engineering*, 15(3):031005, April 2018. ISSN 1741-2552. doi: 10.1088/1741-2552/aab2f2.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International conference on machine learning*, pp. 7176–7185. PMLR, 2020.

Mathilde Papillon, Sophia Sanborn, Johan Mathe, Louisa Cornelis, Abby Bertics, Domas Buracas, Hansen J Lillemark, Christian Shewmake, Fatih Dinc, Xavier Pennec, and Nina Miolane. Beyond euclid: an illustrated guide to modern machine learning with geometric, topological, and algebraic structures. *Machine Learning: Science and Technology*, 6(3):031002, aug 2025. doi: 10.1088/2632-2153/adf375. URL `https://dx.doi.org/10.1088/2632-2153/adf375`.

Xavier Pennec. Intrinsic Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127–154, July 2006. ISSN 0924-9907, 1573-7683. doi: 10.1007/s10851-006-6228-4.

Xavier Pennec. Manifold-valued image processing with SPD matrices. In Xavier Pennec, Stefan Sommer, and Tom Fletcher (eds.), *Riemannian Geometric Statistics in Medical Image Analysis*, pp. 75–134. Academic Press, January 2020. ISBN 978-0-12-814725-2. doi: 10.1016/B978-0-12-814725-2.00010-8.

Gabriel Peyré and Marco Cuturi. Computational optimal transport, 2020. URL `https://arxiv.org/abs/1803.00567`.

G. Pfurtscheller and C. Neuper. Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, 89(7):1123–1134, 2001. doi: 10.1109/5.939829.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992. doi: 10.1137/0330046. URL https://doi.org/10.1137/0330046.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28100–28127. PMLR, 2023. URL https://proceedings.mlr.press/v202/pooladian23a.html.

Marc Rußwurm, Charlotte Pelletier, Maximilian Zollner, Sébastien Lefèvre, and Marco Körner. Breizhcrops: A time series dataset for crop type mapping. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences ISPRS (2020)*, 2020.

Salem Said, Lionel Bombrun, Yannick Berthoumieu, and Jonathan Manton. Riemannian Gaussian Distributions on the Space of Symmetric Positive Definite Matrices, December 2016.

Salem Said, Lionel Bombrun, Yannick Berthoumieu, and Jonathan H Manton. Riemannian Gaussian distributions on the space of Symmetric Positive Definite matrices. *IEEE Transactions on Information Theory*, 63(4):2153–2170, 2017.

D. Serre. *Matrices: Theory and Applications*. Graduate Texts in Mathematics. Springer, 2002. ISBN 978-0-387-95460-8. URL https://books.google.fr/books?id=RDnUIFYgkrUC.

Kiyoshi Shiga. Hadamard Manifolds. In *Geometry of Geodesics and Related Topics*, volume 3, pp. 239–282. Mathematical Society of Japan, January 1984. doi: 10.2969/aspm/00310239.

Lene Theil Skovgaard. A riemannian geometry of the multivariate normal model. *Scandinavian Journal of Statistics*, 11(4):211–223, 1984. ISSN 03036898, 14679469. URL http://www.jstor.org/stable/4615960.

Alexander Smith, Jinxi Hua, Benjamin de Foy, James J Schauer, and Victor M Zavala. Multi-site, multi-pollutant atmospheric data analysis using riemannian geometry. *Science of The Total Environment*, 892: 164064, 2023.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

David Steyrl, Reinhold Scherer, Josef Faller, and Gernot R. Müller-Putz. Random forests in non-invasive sensorimotor rhythm brain-computer interfaces: a practical and convenient non-linear classifier. *Biomedical Engineering / Biomedizinische Technik*, 61(1):77–86, 2016. doi: doi:10.1515/bmt-2014-0117. URL https://doi.org/10.1515/bmt-2014-0117.

Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW. Expert Certification.

Alexander Y. Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free Schrödinger bridges via score and flow matching. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 1279–1287. PMLR, 02–04 May 2024b. URL https://proceedings.mlr.press/v238/tong24a.html.

Olga Zaghen, Floor Eijkelboom, Alison Pouplin, Cong Liu, Max Welling, Jan-Willem van de Meent, and Erik J. Bekkers. Riemannian variational flow matching for material and protein design, 2025. URL https://arxiv.org/abs/2502.12981.

Zhixuan Zhou, Tingting Dan, and Guorong Wu. Brainflow: A holistic pathway of dynamic neural system on manifold. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=fkYZKArFLF.

A More details on Wrapped Gaussian

In this appendix, we give more results on the wrapped Gaussian $\mathrm{WG}(\boldsymbol{X}, \mu, \Sigma)$. These results come from (de Surrel et al., 2025).

Theorem A.1. *The density $f_{\boldsymbol{X},\mu,\Sigma}$ of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{X}, \mu, \Sigma)$ exists and is:*

$$\forall \boldsymbol{Z} \in \mathcal{P}_d,\ f_{\boldsymbol{X},\mu,\Sigma}(\boldsymbol{Z}) = \frac{g_{\mu,\Sigma}(\mathrm{Log}_{\boldsymbol{X}}(\boldsymbol{Z}))}{|J_{\boldsymbol{X}}(\mathrm{Log}_{\boldsymbol{X}}(\boldsymbol{Z}))|} \tag{16}$$

where $g_{\mu,\Sigma}$ is the density of the multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$ and $J_{\boldsymbol{X}}(\cdot) = \det(\mathrm{d}\,\mathrm{Exp}_{\boldsymbol{X}}(\cdot))$ is the Jacobian determinant of the exponential map $\mathrm{Exp}_{\boldsymbol{X}}$.

The Jacobian determinant of the exponential map describes the volume change induced by the identification of $T_{\boldsymbol{X}}\mathcal{P}_d$ and $\mathcal{P}_d$ by the exponential map $\mathrm{Exp}_{\boldsymbol{X}}$. To compute the density explicitly, one needs to compute the Jacobian $J_{\boldsymbol{X}}(u)$, which has a closed form expression for the manifold $\mathcal{P}_d$ (see Appendix C of de Surrel et al. 2025).

Proposition A.2. *The Jacobian determinant of the exponential map at the identity $\mathrm{Exp}_{\boldsymbol{I}_d}$ is:*

$$\forall \boldsymbol{U} \in T_{\boldsymbol{I}_d}\mathcal{P}_d,\ J_{\boldsymbol{I}_d}(\boldsymbol{U}) = 2^{d(d-1)/2} \prod_{i<j} \frac{\sinh\left(\frac{\lambda_i(\boldsymbol{U})-\lambda_j(\boldsymbol{U})}{2}\right)}{\lambda_i(\boldsymbol{U}) - \lambda_j(\boldsymbol{U})}$$

where the $\lambda_i(\boldsymbol{U})$ are the eigenvalues of $\boldsymbol{U}$. Then, one can use the previous formula to compute the Jacobian determinant of the exponential map at any point $\boldsymbol{X} \in \mathcal{P}_d$:

$$\forall \boldsymbol{U} \in T_{\boldsymbol{X}}\mathcal{P}_d,\ J_{\boldsymbol{X}}(\boldsymbol{U}) = J_{\boldsymbol{I}_d}(\boldsymbol{X}^{-1/2}\boldsymbol{U}\boldsymbol{X}^{-1/2}).$$

Let us give a proof of Proposition 3.4 that we recall here:

Proposition A.3. *Let $\boldsymbol{Z} \sim \mathrm{WG}(\boldsymbol{I}_d, 0_D, \boldsymbol{I}_D)$, then*

$$\begin{aligned}\mathbb{E}[\delta(\boldsymbol{I}_d, \boldsymbol{Z})] &= \sqrt{2}\ \frac{\Gamma\left(\frac{d(d+1)+2}{4}\right)}{\Gamma\left(\frac{d(d+1)}{4}\right)} \\ &\sim \sqrt{\frac{d(d+1)}{2}} \quad (d \to +\infty),\end{aligned}$$

where Γ is the Gamma function (see Eq. 16 of Davis 1959).

In order to prove this, let us start by the Euclidean case.

Proposition A.4. *Let $Z \sim \mathcal{N}(0_d, \boldsymbol{I}_d)$, then*

$$\begin{aligned}\mathbb{E}[\|Z\|_2] &= \sqrt{2}\ \frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)} \\ &\sim \sqrt{d} \quad (d \to +\infty),\end{aligned}$$

Proof. As $Z \sim \mathcal{N}(0_d, \boldsymbol{I}_d)$, its norm $\|Z\|_2$ follows a Chi distribution. According to Section 11.3 of Evans et al. (2001), its expectation is

$$\mathbb{E}[\|Z\|_2] = \sqrt{2}\ \frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}.$$

Moreover, using Stirling's formula, one gets, according to formula 21 of Davis (1959) that

$$\Gamma(x) \sim x^{x-1/2}e^{-x}\sqrt{2\pi}.$$

Thus, denoting $n = \frac{d}{2}$, one gets

$$\mathbb{E}[\|Z\|_2] \sim \sqrt{2}\frac{(n+\frac{1}{2})^n e^{-(n+\frac{1}{2})}\sqrt{2\pi}}{n^{n-\frac{1}{2}}e^{-n}\sqrt{2\pi}} = \sqrt{2}e^{-1/2}\sqrt{n}(1+\frac{1}{2n})^n \sim \sqrt{2}\sqrt{n} = \sqrt{d}.$$

□

Going back to the wrapped Gaussian case:

Proof of Proposition 3.4. Let $\boldsymbol{Z} \sim \mathrm{WG}(\boldsymbol{I}_d, 0_D, \boldsymbol{I}_D)$. Then,

$$\begin{aligned}
\mathbb{E}[\delta(\boldsymbol{I}_d, \boldsymbol{Z})] &= \int_{\mathcal{P}_d} \delta(\boldsymbol{I}_d, \boldsymbol{Z}) f_{(\boldsymbol{I}_d, 0_D, \boldsymbol{I}_D)}(\boldsymbol{Z}) \mathrm{dvol}(\boldsymbol{Z}) \\
&= \int_{\mathcal{P}_d} \|\operatorname{Log}_{\boldsymbol{I}_d}(\boldsymbol{Z})\|_{\boldsymbol{I}_d}\ f_{(\boldsymbol{I}_d, 0_D, \boldsymbol{I}_D)}(\boldsymbol{Z}) \mathrm{dvol}(\boldsymbol{Z}) \\
&= \int_{\mathbb{R}^{\frac{d(d+1)}{2}}} \|t\|_2\ g_{0_D, \boldsymbol{I}_D}(t) \mathrm{d}t \quad (\text{where } t = \operatorname{Log}_{\boldsymbol{I}_d}(\boldsymbol{Z})) \\
&= \mathbb{E}[\|t\|_2]
\end{aligned}$$

where $f_{(\boldsymbol{I}_d, 0_D, \boldsymbol{I}_D)}$ is the density of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{I}_d, 0_D, \boldsymbol{I}_D)$ given at Theorem A.1, and $t \sim \mathcal{N}(0_D, \boldsymbol{I}_D)$ and $g_{0_D, \boldsymbol{I}_D}$ is the density of $\mathcal{N}(0_D, \boldsymbol{I}_D)$. Thus, we can use the Proposition A.4 to conclude:

$$\mathbb{E}[\delta(\boldsymbol{I}_d, \boldsymbol{Z})] \sim \sqrt{D} = \sqrt{\frac{d(d+1)}{2}}.$$

□

B Mathematical details on the general framework

B.1 The proof of the general conditional vector field

In this section, we give a proof of Theorem 5.3, which we start by recalling.

Theorem 5.3. *Let $p_t(\cdot \mid \boldsymbol{Z})$ be the conditional probability path given at Equation (9) and Φ_t its flow. The unique vector field defining Φ_t depends on the value of $\sigma_t(\boldsymbol{Z})$ in the following manner:*

- *If $\sigma_t(\boldsymbol{Z}) > 0$, then the unique vector field defining Φ_t is*

$$u_t(\boldsymbol{Y} \mid \boldsymbol{Z}) = \boldsymbol{S}(t)\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{Y} + \boldsymbol{Y}\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{S}(t) + \bar{\boldsymbol{X}}_t^{1/2}\,\mathrm{d}\exp(\log \boldsymbol{M}_t) \cdot \left[\frac{\sigma_t'}{\sigma_t}(\log \boldsymbol{M}_t - \mu_t) + \mu_t'\right]\bar{\boldsymbol{X}}_t^{1/2},$$

  where $\boldsymbol{S}$ is the derivative of $t \mapsto \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}$, $\boldsymbol{M}_t = \bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{Y}\bar{\boldsymbol{X}}_t^{-1/2}$, and all time-dependent quantities are evaluated at $\boldsymbol{Z}$.

- *If $\sigma_t(\boldsymbol{Z}) = 0$, then*

$$u_t(\boldsymbol{Y} \mid \boldsymbol{Z}) = \frac{\mathrm{d}}{\mathrm{d}t}\left[\bar{\boldsymbol{X}}_t^{1/2}\exp(\mu_t)\bar{\boldsymbol{X}}_t^{1/2}\right],$$

  where all time-dependent quantities are evaluated at $\boldsymbol{Z}$.

Proof. We begin by recalling the conditional flow Φ_t from Equation (10):

$$\Phi_t(\boldsymbol{X}) = \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\exp\left[\frac{\sigma_t(\boldsymbol{Z})}{\sigma_{\text{init}}}\log(\boldsymbol{X}) + \mu_t(\boldsymbol{Z})\right]\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}.$$

Let us differentiate this flow with respect to t. For this, we denote

$$\begin{cases} \boldsymbol{S} \text{ the derivative of } t \mapsto \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \\ g(t) = \frac{\sigma_t(\boldsymbol{Z})}{\sigma_{\text{init}}} \log(\boldsymbol{X}) + \mu_t(\boldsymbol{Z}). \end{cases}$$

Then, one has

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(\boldsymbol{X}) = \boldsymbol{S}(t) \exp\left[g(t)\right] \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} + \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \mathrm{d}\exp(g(t)) \cdot [g'(t)] \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} + \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \exp\left[g(t)\right] \boldsymbol{S}(t).$$

Then, one can compute that

$$g'(t) = \frac{\sigma_t'(\boldsymbol{Z})}{\sigma_{\text{init}}} \log(\boldsymbol{X}) + \mu_t'(\boldsymbol{Z}).$$

Thus, going back to $\frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(\boldsymbol{X})$:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(\boldsymbol{X}) &= \boldsymbol{S}(t)\bar{\boldsymbol{X}}_t^{-1/2}\Phi_t(\boldsymbol{X}) + \Phi_t(\boldsymbol{X})\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{S}(t) + \\ &\quad \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \mathrm{d}\exp\left(\frac{\sigma_t(\boldsymbol{Z})}{\sigma_{\text{init}}} \log(\boldsymbol{X}) + \mu_t(\boldsymbol{Z})\right) \cdot \left[\frac{\sigma_t'(\boldsymbol{Z})}{\sigma_{\text{init}}} \log(\boldsymbol{X}) + \mu_t'(\boldsymbol{Z})\right] \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \\ &\quad = u_t(\Phi_t(\boldsymbol{X}) \mid \boldsymbol{Z}). \end{aligned}$$

Let us denote $\boldsymbol{Y} = \Phi_t(\boldsymbol{X})$. We now need to distinguish between the values of σ_t.

- **If** $\sigma_t > 0$, then Φ_t is invertible and $\boldsymbol{X} = \exp\left(\frac{\sigma_{\text{init}}}{\sigma_t(\boldsymbol{Z})}\left(\log \boldsymbol{M}_t - \mu_t(\boldsymbol{Z})\right)\right)$ where $\boldsymbol{M}_t = \bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{Y}\bar{\boldsymbol{X}}_t^{-1/2}$. Plugging this into the previous equation, we get:

$$u_t(\boldsymbol{Y} \mid \boldsymbol{Z}) = \boldsymbol{S}(t)\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{Y} + \boldsymbol{Y}\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{S}(t) + \bar{\boldsymbol{X}}_t^{1/2}\,\mathrm{d}\exp(\log \boldsymbol{M}_t) \cdot \left[\frac{\sigma_t'}{\sigma_t}(\log \boldsymbol{M}_t - \mu_t) + \mu_t'\right] \bar{\boldsymbol{X}}_t^{1/2}.$$

- **If** $\sigma_t = 0$, then Φ_t is not invertible. Going back to the general form of Φ_t given at Equation (10), we have that

$$\Phi_t(\boldsymbol{X}) = \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \exp(\mu_t(\boldsymbol{Z}))\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}.$$

  Thus, the ODE is now simply

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(\boldsymbol{X}) = \frac{\mathrm{d}}{\mathrm{d}t}\left[\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \exp(\mu_t(\boldsymbol{Z}))\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\right] = u_t(\Phi_t(\boldsymbol{X}) \mid \boldsymbol{Z}),$$

  and thus $u_t(\Phi_t(\boldsymbol{X}) \mid \boldsymbol{Z}) = \frac{\mathrm{d}}{\mathrm{d}t}\left[\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \exp(\mu_t(\boldsymbol{Z}))\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\right]$.

Thus, $u_t(\cdot \mid \boldsymbol{Z})$ generates the wrapped Gaussian path $p_t(\cdot \mid \boldsymbol{Z})$. □

B.2 Getting back the specific methods

The goal of this section is to show that the general methods, once instantiated with the specific parameterizations of q the distribution over the conditioning variables, as well as for the parameters $\bar{\boldsymbol{X}}_t$, μ_t and σ_t, correctly lead to the formulas for the different methods. For this, we take the parameterizations from Table 1, and show how they lead to the each FM framework.

RiemannianCFM For RiemannianCFM, the conditional path depends both on $\boldsymbol{X}_0 \sim p$ the source distribution and from $\boldsymbol{X}_1 \sim q_1$ the target distribution. Thus, the conditioning variable $\boldsymbol{Z}$ is the couple $(\boldsymbol{X}_0, \boldsymbol{X}_1)$, and its distribution is the product of $p(\boldsymbol{X}_0)$ and $q_1(\boldsymbol{X}_1)$: $q = p(\boldsymbol{X}_0)q_1(\boldsymbol{X}_1)$. Then, the case of RiemannianCFM is the degenerated case of our framework: $\sigma_t(\boldsymbol{Z}) = 0$ so the covariance of the wrapped Gaussian is

0_D. Moreover, the mean $\mu_t(\boldsymbol{Z}) = 0$, thus, the conditional flow Φ_t given at Equation (10) does not depend on the point $\boldsymbol{X}$ anymore:

$$\Phi_t(\boldsymbol{X}) = \gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t)^{1/2} \exp(0_D) \gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t)^{1/2} = \gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t),$$

where we recall that $\gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t)$ is the unique geodesic joining $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$. Thus, the formula for the conditional vector field from Equation (10) cannot be used directly. However, using the previous formula, one can easily derive

$$u_t(\boldsymbol{Y} \mid \boldsymbol{X}_0, \boldsymbol{X}_1) = \frac{\mathrm{d}}{\mathrm{d}t} \gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t),$$

which is the conditional vector field of the RiemannianCFM as given in Chen & Lipman (2024).

DiffeoCFM For DiffeoCFM, the conditional probability path is only conditioned on $\boldsymbol{Z} = \boldsymbol{X}_1 \sim q_1$, thus $q = q_1(\boldsymbol{X}_1)$. Then, using the parameterization from Table 1, we use:

$$\boldsymbol{X}_t(\boldsymbol{Z}) = \boldsymbol{I}_d, \ \mu_t(\boldsymbol{Z}) = t \log(\boldsymbol{Z}), \text{ and } \sigma_t(\boldsymbol{Z}) = \sigma_{\text{init}} - (\sigma_{\text{init}} - \sigma_{\min})\, t.$$

Going back to the conditional flow Φ_t as given at Equation (10), we have:

$$\Phi_t(\boldsymbol{X}) = \exp\left[\frac{\sigma_t}{\sigma_{\text{init}}} \log(\boldsymbol{X}) + t \log(\boldsymbol{X}_1)\right],$$

which is exactly the flow as given at Equation (7) with the Euclidean flow being the one from Equation 11 of Lipman et al. (2023). Let us now focus on the conditional vector field given at Theorem 5.3. As $\boldsymbol{X}_t(\boldsymbol{X}_1) = \boldsymbol{I}_d$, the time derivative of $\boldsymbol{X}_t(\boldsymbol{X}_1)^{1/2}$ is 0, thus, $\boldsymbol{S}(t) = 0$. Then, the conditional vector field is:

$$\begin{aligned} u_t(\boldsymbol{Y} \mid \boldsymbol{X}_1) &= \mathrm{d}\exp(\log(\boldsymbol{Y})) \cdot \left[\frac{\sigma_t'}{\sigma_t}(\log \boldsymbol{Y} - t \log \boldsymbol{X}_1) + \log \boldsymbol{X}_1\right] \\ &= \mathrm{d}\exp(\log(\boldsymbol{Y})) \cdot \left[\frac{\sigma_{\text{init}} \log \boldsymbol{X}_1 - (\sigma_{\text{init}} - \sigma_{\min}) \log \boldsymbol{Y}}{\sigma_{\text{init}} - (\sigma_{\text{init}} - \sigma_{\min})\, t}\right], \end{aligned}$$

which is exactly the conditional vector field $u_t^{\text{DiffeoCFM}}$ from Equation (8) where the Euclidean conditional vector field is the one from Equation 21 of Lipman et al. (2023).

WrappedCFM Let us do the same work but for WrappedCFM. As for DiffeoCFM, the conditional probability path is only conditioned on $\boldsymbol{Z} = \boldsymbol{X}_1 \sim q_1$, thus $q = q_1(\boldsymbol{X}_1)$. Then, one uses:

$$\boldsymbol{X}_t(\boldsymbol{Z}) = \boldsymbol{X}_1^t, \ \mu_t(\boldsymbol{Z}) = 0, \text{ and } \sigma_t(\boldsymbol{Z}) = \sigma_{\text{init}} - (\sigma_{\text{init}} - \sigma_{\min})\, t.$$

Then, the conditional flow is:

$$\Phi_t(\boldsymbol{X}) = \boldsymbol{X}_1^{t/2} \exp\left(\frac{\sigma_t(\boldsymbol{X}_1)}{\sigma_{\text{init}}} \log(\boldsymbol{X})\right) \boldsymbol{X}_1^{t/2} = \boldsymbol{X}_1^{t/2} \boldsymbol{X}^{\frac{\sigma_t(\boldsymbol{X}_1)}{\sigma_{\text{init}}}} \boldsymbol{X}_1^{t/2},$$

which matches with Equation (13). Now for the conditional vector field, we first need the following lemma:

Lemma B.1 (Equation 10.3 of Serre (2002))**.** *Let $A \in \mathbb{R}^{d \times d}$, then*

$$\frac{\mathrm{d}}{\mathrm{d}t} \exp(tA) = A \exp(tA) = \exp(tA) A.$$

Using this lemma, one gets that

$$\boldsymbol{S}(t) = \frac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{X}_1^{t/2} = \frac{\mathrm{d}}{\mathrm{d}t} \exp\left(\frac{t}{2} \log(\boldsymbol{X}_1)\right) = \frac{1}{2} \log(\boldsymbol{X}_1) \boldsymbol{X}_1^{t/2} = \frac{1}{2} \boldsymbol{X}_1^{t/2} \log(\boldsymbol{X}_1).$$

Moreover, let us consider $\boldsymbol{W} \in \mathcal{P}_d$ and $\boldsymbol{U} \in \mathcal{S}_d$ such that they commute. Then, one has that $\mathrm{d}\exp(\boldsymbol{W}) \cdot [\boldsymbol{U}] = \exp(\boldsymbol{W})\boldsymbol{U}$ (see Equation 10.15 of Higham 2008). Using this, we get:

$$\begin{aligned} u_t(\boldsymbol{Y} \mid \boldsymbol{X}_1) &= \frac{1}{2}\log(\boldsymbol{X}_1)\boldsymbol{X}_1^{t/2}\boldsymbol{X}_1^{-t/2}\boldsymbol{Y} + \frac{1}{2}\boldsymbol{Y}\boldsymbol{X}_1^{-t/2}\boldsymbol{X}_1^{t/2}\log(\boldsymbol{X}_1) + \boldsymbol{X}_1^{t/2}\mathrm{d}\exp(\log(\boldsymbol{M}_t)) \cdot \left[\frac{\sigma_t'}{\sigma_t}\log \boldsymbol{M}_t\right]\boldsymbol{X}_1^{t/2} \\ &= \frac{1}{2}\log(\boldsymbol{X}_1)\boldsymbol{Y} + \frac{1}{2}\boldsymbol{Y}\log(\boldsymbol{X}_1) + \boldsymbol{X}_1^{t/2}\boldsymbol{M}_t\frac{\sigma_t'}{\sigma_t}\log(\boldsymbol{M}_t)\boldsymbol{X}_1^{t/2} \\ &= \frac{1}{2}\log(\boldsymbol{X}_1)\boldsymbol{Y} + \frac{1}{2}\boldsymbol{Y}\log(\boldsymbol{X}_1) + \boldsymbol{X}_1^{t/2}\boldsymbol{M}_t^{1/2}\frac{\sigma_t'}{\sigma_t}\log(\boldsymbol{M}_t)\boldsymbol{M}_t^{1/2}\boldsymbol{X}_1^{t/2}, \end{aligned}$$

where $\boldsymbol{M}_t = \boldsymbol{X}_1^{-t/2}\boldsymbol{Y}\boldsymbol{X}_1^{-t/2}$. This is exactly the conditional vector field from Theorem 6.1.

C No identifiability issue

The goal of this section is to show that the identifiability issues of the Wrapped Gaussian statistical model discussed in de Surrel et al. (2025) do not change the conditional flow given in Equation (10) nor the conditional vector field in Theorem 5.3. Let us start by recalling Proposition 4.9 of de Surrel et al. (2025):

Proposition C.1 (Proposition 4.9 of de Surrel et al. (2025))**.** *Let $\boldsymbol{X} \in \mathcal{P}_d, \mu \in \mathbb{R}^D$ and $\Sigma \in \mathcal{P}_D$. Then, for all $s \in \mathbb{R}$, one has that*

$$\mathrm{WG}(\boldsymbol{X}, \mu, \Sigma) = \mathrm{WG}(e^s\boldsymbol{X}, \mu - sI_d, \Sigma).$$

Let us show that changing the parameterization in the general conditional probability path by an equivalent one using the above formula does not have any impact on the rest of the derivations.

For the conditional flow Let us consider $s \in \mathbb{R}^D$ and let us denote by $\tilde{p}_t(\cdot \mid \boldsymbol{Z})$ the following equivalent conditional probability path:

$$\tilde{p}_t(\cdot \mid \boldsymbol{Z}) = \mathrm{WG}(e^s\bar{\boldsymbol{X}}_t(\boldsymbol{Z}), \mu_t(\boldsymbol{Z}) - sI_d, \sigma_t(\boldsymbol{Z})^2\boldsymbol{I}_D).$$

According to Proposition C.1, one has $\tilde{p}_t(\cdot \mid \boldsymbol{Z}) = p_t(\cdot \mid \boldsymbol{Z})$. Moreover, the flow $\tilde{\Phi}_t$ satisfying $\tilde{\Phi}_t \# p_0 = \tilde{p}_t(\cdot \mid \boldsymbol{Z})$ is, according to Equation (10):

$$\begin{aligned} \tilde{\Phi}_t(\boldsymbol{X}) &= e^{s/2}\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\exp\left[\frac{\sigma_t(\boldsymbol{Z})}{\sigma_{\mathrm{init}}}\log(\boldsymbol{X}) + \mu_t(\boldsymbol{Z}) - sI_d\right]e^{s/2}\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \\ &= e^s\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\exp\left[\frac{\sigma_t(\boldsymbol{Z})}{\sigma_{\mathrm{init}}}\log(\boldsymbol{X}) + \mu_t(\boldsymbol{Z})\right]\underbrace{\exp\left[-sI_d\right]}_{=e^s}I_d\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \\ &= \Phi_t(\boldsymbol{X}). \end{aligned}$$

For the conditional vector field We showed above that the conditional flow is the same for all equivalent parametrization of the conditional probability paths, thus, using Theorem 5.3, the conditional vector field being unique, it is also the same.

D More details on the synthetic experiments

In this section, we give more details on the synthetic experiments we conducted in Section 7. We start by detailing the different datasets used (*Swiss-Roll*, *Figure-8* and *Mixture of WG*) and then give the setup and the architecture used.

D.1 Details on the datasets

The 2×2 SPD Swiss-Roll We give here the equations that were used to generate the Swiss-roll dataset. We start by generating a 3D spiral using the following equations:

$$u = t\cos(t), \quad v = t\sin(t), \quad w = t + s + 1, \tag{17}$$

where $t \sim \mathcal{U}[0, 2\pi]$ is the angle of the spiral and $s \sim \mathcal{U}[0, 3]$ is the ribbon's width. Then, we rotate the 3D spiral to put it into the cone of equation $a > 0$ and $ac > b^2$ i.e. the cone of 2×2 SPD matrices of the form $\begin{pmatrix} a & b \\ b & c \end{pmatrix}$:

$$a = \frac{1}{2}(w + u), \quad c = \frac{1}{2}(w - u), \text{ and } b = \frac{1}{2}v. \tag{18}$$

Then, the matrix

$$\boldsymbol{X} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

will be SPD and will form a spiral in the cone of 2×2 SPD matrices.

The Figure-8 Now, let us give the equations used to create the SPD Figure-8 dataset. We use a technique similar to the one used for the above Swiss-Roll dataset. We start by creating a lemniscate-like shape using the following equations:

$$u = \cos(t), \quad v = \sin(2t), \quad w = t + s + \frac{1}{2}, \tag{19}$$

where $t \sim \mathcal{U}[0, 2\pi]$ is the angle of the spiral and $s \sim \mathcal{U}[0, 1]$ is the ribbon's width. Then, we rotate the 3D spiral to put it into the cone of equation $a > 0$ and $ac > b^2$ i.e. the cone of 2×2 SPD matrices of the form $\begin{pmatrix} a & b \\ b & c \end{pmatrix}$:

$$a = \frac{1}{2}(w + u), \quad c = \frac{1}{2}(w - u), \text{ and } b = \frac{1}{2}v. \tag{20}$$

Then, the matrix

$$\boldsymbol{X} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

will be SPD and will form an 8 shape in the cone of 2×2 SPD matrices.

The Mixture of WG The last dataset is composed of 10×10 SPD matrices sampled from a mixture of three wrapped Gaussians. The mixture is:

$$q = \frac{1}{3}\,\mathrm{WG}(\boldsymbol{X}_1, 0_{55}, \sigma^2_{\text{init}}\boldsymbol{I}_{55}) + \frac{1}{3}\,\mathrm{WG}(\boldsymbol{X}_2, 0_{55}, \sigma^2_{\text{init}}\boldsymbol{I}_{55}) + \frac{1}{3}\,\mathrm{WG}(\boldsymbol{X}_3, 0_{55}, \sigma^2_{\text{init}}\boldsymbol{I}_{55}).$$

Here, the dataset has a simple shape. However, the goal is to produce matrices with different condition numbers. We recall that, for an SPD matrix $\boldsymbol{X}$, the condition number $\kappa(\boldsymbol{X})$ can be computed as

$$\kappa(\boldsymbol{X}) = \frac{\lambda_{\max}(\boldsymbol{X})}{\lambda_{\min}(\boldsymbol{X})},$$

where $\lambda_{\max}(\boldsymbol{X})$ is the largest eigenvalue of $\boldsymbol{X}$ and $\lambda_{\min}(\boldsymbol{X})$ is the smallest (see Section 2.6.2 of Golub & van Loan 2013). When this condition number is very large compared to one, the matrix $\boldsymbol{X}$ is said to be *ill-conditioned*, and the computation of its inverse, or solution of a linear system of equations is prone to large numerical errors.

In this experiment, we wanted to generate the centers of the components of the wrapped Gaussian mixture to have specific condition numbers $\kappa \in \{10, 10^2, 10^3, 10^4\}$. For each experiment, we set a specific κ. Then, the centers $\boldsymbol{X}_k$ are chosen so that they all have as condition number κ. In order to do so, we choose $\boldsymbol{X}_k$ as:

$$\boldsymbol{X}_k = \boldsymbol{Q}_k \,\mathrm{diag}(\lambda_1, ..., \lambda_d)\boldsymbol{Q}_k^T$$

with the eigenvalues $(\lambda_1, ..., \lambda_d)$ log-spaced in $[\kappa^{-1/2}, \kappa^{1/2}]$ and an independent random orientation $\boldsymbol{Q}_k$. The matrices $\boldsymbol{X}_k$ have, by construction, exactly a condition number equal to κ. The dataset is centered and we sample the same amount of points from each of the three wrapped Gaussians, and label them according to the component of the mixture we sampled from.

For all of those three datasets, the training dataset is composed of 8000 matrices, the validation set is composed of 1000 matrices and the testing set is composed of 1000 matrices. We created 5 different versions of each dataset, each sampled with a seed ranging from 0 to 4.

D.2 Details on the architecture used

For the two first 2×2 datasets (the 2×2 *SPD Swiss-roll* and the *Figure-8*), the conditional vector field $u_t(\cdot \mid \boldsymbol{Z})$ was learned using a MLP composed of 6 layers, each composed of 128 neurons and a SELU activation function (*Scaled Exponential Linear Unit*) (Klambauer et al., 2017). Timestep and class information are injected by concatenating them to the input. The input $\boldsymbol{X}$ of conditional vector field $u_t(\boldsymbol{X} \mid \boldsymbol{Z})$ is an SPD matrix and the output is an element of the tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$; they are both symmetric matrices. Therefore, we vectorized the input matrix by taking only the elements on the upper-triangular part and unvectorizing the output matrix.

For the higher dimensional dataset, we used a more complex architecture composed of 10 layers each with 512 hidden units and a SELU activation function.

For all the synthetic experiments, we optimized the model using the AdamW optimizer (Loshchilov & Hutter, 2017) with default $\beta = (0.9, 0.999)$ and $\epsilon = 10^{-8}$ with weight decay explicitly disabled. The base learning rate is 10^{-3}, modulated by a two-phase schedule stepped once per epoch. It is built using a linear warmup going from 10^{-4} to 10^{-3} during the first 10 epochs and cosine annealing. Gradients are clipped to global norm 1.0, a guard against the loss spikes observed at high condition number. An exponential moving average on the weights (Polyak & Juditsky, 1992) was used with a decay of 0.999. We trained it for 12000 epochs. The synthetic experiments were conducted on NVIDIA A40 GPU with 40GB of memory. We train our Riemannian models with the AIRM loss used in Chen & Lipman (2024) and the EuclideanCFM using the classical Frobenius loss on symmetric matrices.

Importantly, all the trainings were done in `float64`, the computations on SPD matrices requiring more precision.

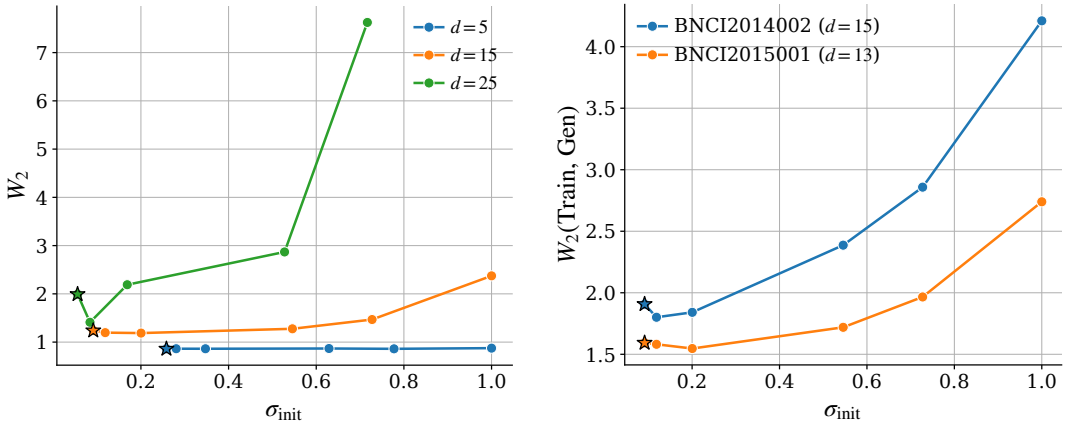
D.3 How badly does EuclideanCFM violate positive definiteness?

In Section 7 we noted that EuclideanCFM fails to consistently produce SPD matrices when the mixture centers are ill-conditioned. Here we quantify *how* the constraint is violated. For each condition number $\kappa \in \{100, 1000, 10000\}$ we draw $N = 1001$ samples X_1 from the trained EuclideanCFM model (the same setting as Section 7, $d = 10$) and compute their eigenvalues $\lambda_1 \leq \cdots \leq \lambda_d$. A sample is counted as *non-SPD* whenever $\lambda_1 \leq 0$.

	κ		
	100	1000	10000
Non-SPD samples	11.2%	47.7%	100%
Non-positive eigenvalues per non-SPD sample (mean / median / max)	1.0/1/2	1.1/1/2	2.7/3/5
Non-SPD samples with $\geqslant 2$ non-positive eigenvalues	0.9%	5.9%	98.7%
λ_1 of non-SPD samples (median / min)	$-0.039/-0.22$	$-0.031/-0.22$	$-2.28/-4.34$
Mean sorted spectrum of non-SPD samples			
Smallest four eigenvalues			
λ_1	-0.049	-0.040	-2.30
λ_2	$+0.143$	$+0.072$	-1.16
λ_3	$+0.305$	$+0.192$	-0.21
λ_4	$+0.494$	$+0.344$	$+0.67$

Table 6: Spectra of matrices generated by EuclideanCFM as a function of the conditioning κ of the mixture centers. For mild conditioning, violations of positive definiteness are characterized by a single shallow negative eigenvalue. At $\kappa = 10000$, a substantial part of the spectrum is negative.

Two regimes. For $\kappa = 100$ and $\kappa = 1000$ the violation is *marginal*: 94–99% of the non-SPD samples have a single non-positive eigenvalue, with median $\lambda_1 \approx -0.03$ and worst case ≈ -0.2. The next eigenvalue λ_2



(a) WG mixtures with $\kappa = 10$.

(b) Real-world datasets.

Figure 5: Effect of σ_{init} on WrappedCFM for different matrix sizes d. Stars indicate the default value used in our experiments, $\sigma_{\text{init}} = \sqrt{\frac{2}{d(d+1)}}$.

is already comfortably positive. What grows with κ in this regime is the *frequency* of these shallow dips (11% $\to$ 48%), not their depth; the samples are effectively near-singular rather than structurally indefinite.

For $\kappa = 10000$ the violation is *substantial*. Every generated sample is non-SPD, with on average 2.7 of the 10 eigenvalues non-positive (median 3, up to 5), and 98.7% of samples having at least two negative eigenvalues. The most negative eigenvalue is ≈ -2.3 on average and reaches -4.3, with the second around -1.2. Roughly the bottom quarter to third of the spectrum is genuinely negative. This is not a numerical artefact that a small diagonal shift would repair, and it motivates a geometry-aware model rather than post-hoc projection onto the SPD cone.

D.4 Sensitivity to σ_{init}

In our experiments, we set $\sigma_{\text{init}} = \sqrt{\frac{2}{d(d+1)}}$, which controls the spread of the source distribution $p_0 = \text{WG}(I_d, 0, \sigma_{\text{init}}^2 I_D)$. This choice was motivated by Proposition 3.4 and required no hyperparameter tuning. In Figure 5, we investigate the effect of varying σ_{init} between the proposed value and 1, as well as the influence of the dimension d. We evaluate WrappedCFM on synthetic WG mixtures with $d \in \{5, 15, 25\}$ and on two real-world datasets with $d = 13$ and $d = 15$. For $d = 5$, performance is nearly invariant (from $W_2 = 0.860$ to $W_2 = 0.875$), whereas sensitivity increases with dimension. For $d = 25$, W_2 increases from 1.990 to 7.625. At $d = 25$ and $\sigma_{\text{init}} = 1$, sampling could not be completed because of a numerical eigendecomposition error. The real-world experiments show the same general trend: W_2 increases from 1.907 to 4.210 on BNCI2014002 and from 1.593 to 2.739 on BNCI2015001. Although the proposed value does not always achieve the lowest observed W_2, it provides a reliable default value without the need for hyperparameter tuning.

E Training and sampling time with respect to the dimension

We also measure the time for one training step and for the sampling of 1000 new points when the dimension rises in Figure 6. For this, we used the mixture of Gaussian synthetic experiment detailed in Section 7. We fixed the condition number of the centers of the Gaussians to $\kappa = 10$ and varied the dimension. The architecture is kept identical across methods and its size is almost unaffected by n (from 2.4M to 3.7M parameters between $n = 5$ and $n = 50$), so the reported gaps reflect the cost of the geometry rather than that of the network. At low dimension all four methods are essentially equivalent: at $n = 5$, one training step takes between 40 and 49 ms, and sampling 1000 points between 1.26 and 1.66 s. As n grows, the methods separate according to how often they require matrix functions. EuclideanCFM, which ignores the geometry, is almost insensitive to the dimension, whereas DiffeoCFM, RiemannianCFM and WrappedCFM

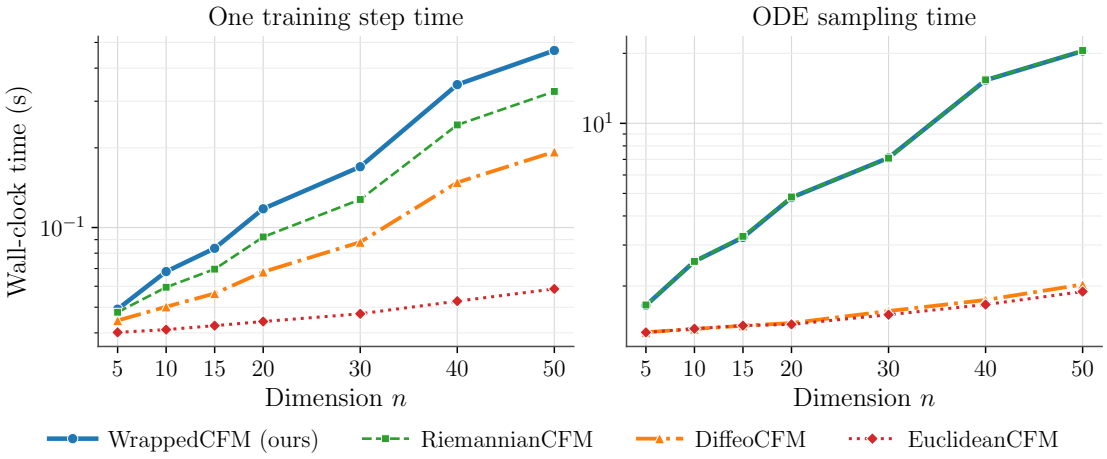


Figure 6: Computational time for one training step and for the sampling with respect to the dimension.

grow roughly as $n^{1.0}$, $n^{1.3}$ and $n^{1.4}$ respectively, reaching 0.19, 0.33 and 0.47 s per step at $n = 50$. The overhead of WrappedCFM over RiemannianCFM stays moderate throughout (a factor 1.03 at $n = 5$ and 1.43 at $n = 50$) and comes from the wrapping of the target samples, which is done once per step. The picture is different at sampling time: the cost of DiffeoCFM stays within 7% of the Euclidean baseline at every dimension, since its ODE is integrated in a fixed chart, while WrappedCFM and RiemannianCFM integrate on the manifold and therefore evaluate an exponential and a logarithm at every solver step. Their sampling times are indistinguishable from each other (they differ by less than 1% at every dimension), going from 1.65 s at $n = 5$ to 20.5 s at $n = 50$: the wrapped construction adds no cost at inference. Overall, in the range of dimensions relevant to our real-world experiments ($n \leq 40$), a training step of WrappedCFM stays below 0.35 s and the generation of 1000 matrices below 16 s, which we consider a reasonable price.

F More details on real-world experiments

F.1 Datasets preprocessing

Dataset	Application	Dim.	# Matrices (train/test)	# Classes	Reference
BNCI2014002	BCI	15×15	1,587/177	2	(Steyrl et al., 2016)
BNCI2015001	BCI	13×13	3888/432	2	Faller et al. (2012)
Textile	Image Analysis	10×10	14,400/1,600	2	Bergmann et al. (2021)
BreizhCrops	Satellite Images	13×13	327,760/80,421	4/9	Rußwurm et al. (2020)

Table 7: Summary of datasets used in the experiments.

BCI

The BCI datasets that we consider are *BNCI2014002* (Steyrl et al., 2016) and *BNCI2015001* (Faller et al., 2012). They consist of several subjects and several sessions per subjects performing a Motor Imagery task Pfurtscheller & Neuper (2001). We used the library MOABB Aristimunha et al. (2025) to load and preprocess the data. The preprocessing of BNCI2014002 consists of a band-pass filter in the range $[7; 35]$ Hz. Then, we used the OAS estimator (Chen et al., 2010) to compute the covariance matrices. We follow a leave-one-group-out protocol. BNCI2014002 comprises 5 groups, and BNCI2015001 comprises 2 groups. We followed the same preprocessing as in Collas et al. (2025).

Textile

This dataset is made of a set of real images from textile manufacturing that contain non-defective and defective woven textiles. These images come from the public MVTec Anomaly Detection dataset Bergmann et al. (2021). The same preprocessing as in Smith et al. (2023) was applied to get the covariance matrices. The two classes correspond to defective and non-defective textiles.

BreizhCrops

This dataset is intended for supervised classification of field crops from satellite time series. We use the default geographical split, where regions frh01–frh03 form the training set and frh04 the test set. Since the dataset is strongly imbalanced, we select the 4 most represented classes. More details on this dataset can be found in the original paper Rußwurm et al. (2020).

F.2 Training details

Hyperparameters The datasets are split into train, validation, test with 80/10/10 ratio. All the models are trained using the same MLP architecture as the synthetic experiments, with 6 layers and 512 hidden units. We trained for 200k iterations on BreizhCrops and 100k iterations for all other datasets. We used Adam optimizer with a learning rate of 1e-4 and exponential moving average on weights with a decay of 0.999 for all our experiments. All the experiments are repeated 5 times with different random seeds.

Classifier The classifier is a ℓ_2-regularized logistic regression classifier trained on SPD matrices that have been projected onto the tangent space at their Fréchet mean. The regularization parameter of the classifier is selected by 5-fold cross-validation.