Supplementary Material

Bayesian Quadrature on Riemannian Data Manifolds

A.1. Riemannian Geometry

A *manifold* $\mathcal{M}$ of dimension D is a topological space which does not carry a global vector space structure. As opposed to the familiar $\mathbb{R}^D$, a manifold lacks the possibility of adding or scaling vectors globally. Instead, an *atlas* it used to cover the manifold in *charts*, which only locally give a Euclidean view of the manifold. If transition maps between overlapping charts are smooth, we call $\mathcal{M}$ a *smooth manifold*, which provides the means for doing calculus. Our analysis is heavily simplified by viewing $\mathbb{R}^D$ as a manifold and employing the identity map as a global chart map, which covers the whole $\mathbb{R}^D$, thereby endowing the manifold automatically with the smoothness property. We use global Euclidean coordinates, which implies that we can solve the geodesic equations directly in this global chart.

A.1.1. Geodesic Equations

The energy or action functional of a curve $\boldsymbol{\gamma}$ with time derivative $\dot{\boldsymbol{\gamma}}(t)$ is defined as

$$E(\boldsymbol{\gamma}) = \frac{1}{2}\int_0^1 \underbrace{\langle \dot{\boldsymbol{\gamma}}(t), \mathbf{M}(\boldsymbol{\gamma}(t))\dot{\boldsymbol{\gamma}}(t)\rangle}_{=:\mathscr{L}} \,\mathrm{d}t. \tag{12}$$

In physics, the argument of the integral is known as *Lagrangian* and we therefore abbreviate the inner product as $\mathscr{L} := \langle \dot{\boldsymbol{\gamma}}(t), \mathbf{M}(\boldsymbol{\gamma}(t))\dot{\boldsymbol{\gamma}}(t)\rangle$. Geodesics are the stationary curves of this functional. We are interested in the minimizers, i.e., shortest paths. Minimizing curve energy instead of length avoids the issue of arbitrary reparameterization. Let γ^i denote the i-th coordinate of the curve $\boldsymbol{\gamma}$ at time t and M_{ik} the metric component at row i and column k, if it is represented as a matrix. We leave sums over repeated indices implicit (Einstein summation convention). Applying the Euler-Lagrange equations to the functional E results in a system of equations involving $\mathscr{L}$

$$\frac{\partial \mathscr{L}}{\partial \gamma^k} = \frac{\partial}{\partial t}\frac{\partial \mathscr{L}}{\partial \dot{\gamma}^k}, \quad \text{for } k \in 1, \ldots, D. \tag{13}$$

which is a system of 2^{nd} order differential equations. We first consider the left-hand side

$$\mathrm{I} \coloneqq \frac{\partial \mathscr{L}}{\partial \gamma^k} = \frac{1}{2}\frac{\partial M_{ij}}{\partial \gamma^k}\dot{\gamma}^i\dot{\gamma}^j, \tag{14}$$

which holds due to independence of the coordinates. The right-hand side is

$$\mathrm{II} \coloneqq \frac{\partial}{\partial t}\left[M_{ik}\dot{\gamma}^i\right] = \frac{\partial M_{ik}}{\partial \gamma^j}\dot{\gamma}^i\dot{\gamma}^j + M_{ik}\ddot{\gamma}^i. \tag{15}$$

We expand this using a small index rearrangement trick

$$\mathrm{II} = \frac{1}{2}\frac{\partial M_{ik}}{\partial \gamma^j}\dot{\gamma}^i\dot{\gamma}^j + \frac{1}{2}\frac{\partial M_{jk}}{\partial \gamma^i}\dot{\gamma}^i\dot{\gamma}^j + M_{ik}\ddot{\gamma}^i. \tag{16}$$

This allows us to write $\mathrm{I} = \mathrm{II} \Leftrightarrow \mathrm{II} - \mathrm{I} = 0$ as

$$M_{ik}\ddot{\gamma}^i + \frac{1}{2}\left(\frac{\partial M_{ik}}{\partial \gamma^j} + \frac{\partial M_{jk}}{\partial \gamma^i} - \frac{\partial M_{ij}}{\partial \gamma^k}\right)\dot{\gamma}^j = 0. \tag{17}$$

the next step is to left multiply with the inverse metric tensor and plug in the *Christoffel symbols* defined as follows

$$\Gamma^k_{ij} = \frac{1}{2}M^{-1}_{kh}\left(\frac{\partial M_{ih}}{\partial \gamma^j} + \frac{\partial M_{jh}}{\partial \gamma^i} - \frac{\partial M_{ij}}{\partial \gamma^h}\right), \tag{18}$$

so we finally obtain the geodesic equations in the canonical form

$$\ddot{\gamma}^k + \Gamma^k_{ij}\dot{\gamma}^j\dot{\gamma}^j = 0, \quad \text{for } k \in 1, \ldots, D. \tag{19}$$

We assume our manifold to be *geodesically complete* (Pennec, 2006), which means that geodesics can be infinitely extended, i.e., their domain is $\mathbb{R}$. As a consequence, the exponential map is then defined on the whole tangent space. In theory, the exponential map $\mathrm{Exp}_{\boldsymbol{\mu}}(\cdot)$ is a *diffeomorphism* only in some open neighborhood around $\boldsymbol{\mu}$ and thus it only admits a smooth inverse, i.e., $\mathrm{Log}_{\boldsymbol{\mu}}(\cdot)$, in said neighborhood. However, we assume this to be true on the whole manifold in practice to keep the analysis tractable. For long geodesics on high-curvature data manifolds, often $\mathrm{Log}_{\boldsymbol{\mu}}(\mathrm{Exp}_{\boldsymbol{\mu}}(\mathbf{v})) \neq \mathbf{v}$. This is rather unproblematic since if $\| \mathrm{Log}_{\boldsymbol{\mu}_l}(\mathbf{x}_n)\|$ is high, the responsibility r_{nl} will be low (see A.2), so this logarithmic map will play a minimal role in the Mahalanobis distance of the LAND density. Thus, the optimization process on its own favors mean and covariances such that the density is concentrated in sufficiently small neighborhoods where the exponential map approximately admits an inverse.

A.1.2. Covariance and Precision Matrices

We here elaborate on Footnote 1 of the paper. The Riemannian normal distribution (Pennec, 2006) is defined using the precision matrix $\boldsymbol{\Gamma}$. This matrix lives on the tangent space $\mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$, i.e., it may be represented as a matrix in $\mathbb{R}^{D\times D}$, where D is the dimension of the tangent space, which is equal to the topological dimension of the manifold. In our applied setting, D matches the dimension of the data space, as we view the whole $\mathbb{R}^D$ as the manifold. We can use the tangent space "covariance" matrix $\boldsymbol{\Sigma} = \boldsymbol{\Gamma}^{-1}$ for our reasoning and the optimization process. However, to obtain the true covariance on the manifold $\mathcal{M}$, a subtle correction is necessary (Pennec, 2006)

$$\boldsymbol{\Sigma}_{\mathcal{M}} = \mathbb{E}\left[\mathrm{Log}_{\boldsymbol{\mu}}(\mathbf{x})\,\mathrm{Log}_{\boldsymbol{\mu}}(\mathbf{x})^\intercal\right] = \frac{1}{\mathcal{C}}\int_{\mathcal{M}} \mathrm{Log}_{\boldsymbol{\mu}}(\mathbf{x})\,\mathrm{Log}_{\boldsymbol{\mu}}(\mathbf{x})^\intercal \exp\left(-\frac{1}{2}\left\langle \mathrm{Log}_{\boldsymbol{\mu}}(\mathbf{x}), \boldsymbol{\Gamma}\,\mathrm{Log}_{\boldsymbol{\mu}}(\mathbf{x})\right\rangle\right) \mathrm{d}\mathcal{M}(\mathbf{x}), \tag{20}$$

with respect to the density on the manifold. For conceptual ease, we focus on the tangent space view in the paper. To plot the eigenvectors of the ADK LAND covariance (Fig. 10), we used the exponential map on the tangent space covariance matrix, i.e., we evaluate and plot $\mathrm{Exp}_{\boldsymbol{\mu}}(\mathbf{v}_{1:2})$, where $\mathbf{v}_{1:2}$ are the eigenvectors of $\boldsymbol{\Sigma}$.

A.1.3. Geodesic Solvers

To solve the geodesic equations, we combine two solvers, which have different strengths and weaknesses. By chaining them together, we obtain a more robust computational pipeline.

First, we make use of the fast and robust fixed-point solver (FP) introduced by Arvanitidis et al. (2019a). This solver pursues a GP-based approach that avoids the often ill-behaved Jacobians of the geodesic ODE system. However, the resulting logarithmic maps are subject to significant approximation error, depending on the curvature of the manifold. The parameters of this solver are as follows:

Parameter	Value	Description
$\texttt{iter}_{\texttt{max}}$	1000	maximum number of iterations
`N`	10	number of mesh nodes.
`tol`	0.1	tolerance used to evaluate solution correctness.
σ	10^{-4}	noise of the GP.

For MNIST, we set $\texttt{iter}_{\texttt{max}} = 500$, and $\texttt{tol} = 0.2$, since this high-curvature manifold easily leads to failing geodesics.

The second solver we employ is a precise, albeit less robust one. This is the BVP solver available in the module `scipy.integrate.solve_bvp`. On high-curvature manifolds, this solver often fails (especially for long curves) and takes a significant amount of time to run. When it succeeds, however, the logarithmic maps are reliable. For this solver, we set the maximum number of mesh nodes to 100 and the tolerance to 0.1. We empirically found that choosing a high maximum number of mesh nodes (e.g., 500) can lead to high runtimes for failing geodesic computations.

To obtain fast and robust geodesics, these solvers may be chained together, i.e., we initialize the BVP solver with the FP solution, which is often worth the extra effort for speedup and improved robustness. For initialization, we use 20 mesh nodes, evenly spaced on the FP solution. If the FP solver already failed, it is very unlikely for the BVP solver to succeed, so we abort the computation.

Algorithm 1 LAND mixture main loop

Input: data $\mathbf{x}_{1:N}$, manifold $\mathcal{M}$ with Exp and Log operators, max. number of iterations t_{max},
initial stepsize $\alpha^1_{\boldsymbol{\mu}} \in \mathbb{R}$, gradient tolerance $\epsilon_{\nabla_{\boldsymbol{\mu}}}$, likelihood tolerance $\epsilon_{\mathcal{L}}$
Output: estimates $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \mathcal{C}_k, \pi_k)_{1:K}$
Initialize LAND parameters $\left(\boldsymbol{\mu}^1_k, \boldsymbol{\Sigma}^1_k, \mathcal{C}^1_k, \pi^1_k\right)_{1:K}$, $t \leftarrow 1$.
repeat
  Expectation step: $r_{nk} = \frac{\pi_k p(x_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^K \pi_l p(x_n|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}$
  Maximization step:
  for $k = 1$ **to** K **do**
    Compute $\mathcal{C}^t_k(\boldsymbol{\mu}^t_k, \boldsymbol{\Sigma}^t_k)$
    Compute $d_{\boldsymbol{\mu}_k}\mathcal{L}(\boldsymbol{\mu}^t_k, \boldsymbol{\Sigma}^t_k)$ using Eq. (22)
    if $||d_{\boldsymbol{\mu}_k}\mathcal{L}|| < \epsilon_{\nabla_{\boldsymbol{\mu}}}$ **then**
      Continue
    end if
    $\boldsymbol{\mu}^{t+1}_k \leftarrow \mathrm{Exp}_{\boldsymbol{\mu}^t_k}(\alpha^t_{\boldsymbol{\mu}} d_{\boldsymbol{\mu}_k}\mathcal{L})$
    Compute $\mathrm{Log}_{\boldsymbol{\mu}^{t+1}_k}(\mathbf{x}_{1:N})$
    Compute $\mathcal{C}^{t+1}_k(\boldsymbol{\mu}^{t+1}_k, \boldsymbol{\Sigma}^t_k)$
    $\boldsymbol{\Sigma}^{t+1}_k \leftarrow \mathrm{update}_{\boldsymbol{\Sigma}^t_k}$ using Alg. 2
    $\pi^t_k = \frac{1}{N}\sum_{n=1}^N r_{nk}$
  end for
  if $\mathcal{L}^{t+1} < \mathcal{L}^t$ **then**
    $\alpha^{t+1}_{\boldsymbol{\mu}} \leftarrow 1.1 \cdot \alpha^t_{\boldsymbol{\mu}}$ {optimism}
  else
    $\alpha^{t+1}_{\boldsymbol{\mu}} \leftarrow 0.75 \cdot \alpha^t_{\boldsymbol{\mu}}$ {pessimism}
  end if
  $t \leftarrow t + 1$
until $||\mathcal{L}^{t+1} - \mathcal{L}^t|| \leq \epsilon_{\mathcal{L}}$ **or** $t = t_{max}$

Furthermore, we exploit previously computed BVP solutions: assume we want to compute $\mathrm{Log}_{\boldsymbol{\mu}_t}(\mathbf{x})$. We search for past results $\mathrm{Log}_{\boldsymbol{\mu}^*_t}(\mathbf{x})$, with $t^* < t$, $t^* = \arg\min \|\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t*}\|$ and $\|\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t*}\| < \epsilon_d$, where we choose $\epsilon_d = 0.5$. Since we compute logarithmic maps for data points $\mathbf{x}_{1:N}$, which do not change during LAND optimization, we can use them as hash keys in a dictionary, where we store the solutions. Looking up the solution is then linear in the number of previous LAND iterations. If such a solution is found, the FP is skipped and the solution is used to directly initialize the BVP solver.

For the exponential maps, we use `scipy.integrate.solve_ivp` with a tolerance of 10^{-3}.

A.2. The LAND Objective and Gradients

Given a dataset $\mathbf{x}_{1:N}$ assumed to be i.i.d., the negative log-likelihood of the *Locally Adaptive Normal Distribution* (LAND) mixture can be stated as (Arvanitidis et al., 2016)

$$\mathcal{L}\left(\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{1:K}\right) = \sum_{k=1}^K \sum_{n=1}^N r_{nk}\left[\frac{1}{2}\langle \mathrm{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}_n), \boldsymbol{\Sigma}_k^{-1}\,\mathrm{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}_n)\rangle + \log\left(\mathcal{C}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) - \log(\pi_k)\right] \tag{21}$$

where π_k is the weight of the k^{th} component, $\sum_{k=1}^K \pi_k = 1$ and $r_{nk} = \frac{\pi_k p(x_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^K \pi_l p(x_n|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}$ is the responsibility of the k^{th} component for the n^{th} datum. The maximum likelihood solution can be obtained by non-convex optimization, alternating between gradient descent updates of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and cycling through the components k, as described in Alg. 1.

For $\boldsymbol{\mu}$, we use the steepest descent direction as in Arvanitidis et al. (2016)

$$d_{\boldsymbol{\mu}_k}\mathcal{L} = \sum_{n=1}^{N} r_{nk} \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}_n) - \frac{\mathcal{Z}_k \cdot R_k}{\mathcal{C}_k(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \int_{\mathcal{T}_{\boldsymbol{\mu}_k}\mathcal{M}} \mathbf{v} g_{\boldsymbol{\mu}_k}(\mathbf{v}) \mathcal{N}(\mathbf{v}; \mathbf{0}, \boldsymbol{\Sigma}_k)\, \mathrm{d}\mathbf{v}, \tag{22}$$

where the vector-valued integral stems from BQ and $R_k = \sum_{n=1}^{N} r_{nk}$, $\mathcal{Z}_k = \sqrt{(2\pi)^d |\boldsymbol{\Sigma}_k|}$.

Arvanitidis et al. (2016) decomposed the precision $\boldsymbol{\Sigma}_k^{-1} = \mathbf{A}^\intercal \mathbf{A}$ for unconstrained optimization using gradient descent. We opt for a more principled approach by exploiting geometric structure of the symmetric positive definite (SPD) manifold, to which the covariance is confined. More specifically, we use the *bi-invariant* metric (Bhatia, 2009). Under this metric, geodesics from $\mathbf{A}$ to $\mathbf{B}$ may be parameterized as $\gamma(t) = \mathbf{A}^{\frac{1}{2}} \left(\mathbf{A}^{-\frac{1}{2}} \mathbf{B}^{\frac{1}{2}} \mathbf{A}^{-\frac{1}{2}}\right)^t \mathbf{A}^{\frac{1}{2}}, \quad 0 \leq t < 1$, and the distance from $\mathbf{A}$ to $\mathbf{B}$ is $d(\mathbf{A}, \mathbf{B}) = \left\| \log \mathbf{A}^{-\frac{1}{2}} \mathbf{B}^{\frac{1}{2}} \mathbf{A}^{-\frac{1}{2}} \right\|_2$. The name stems from the fact that this distance is invariant under multiplication with any invertible square matrix $\boldsymbol{\Xi}$, i.e., $d(\mathbf{A}, \mathbf{B}) = d(\boldsymbol{\Xi} \cdot \mathbf{A}, \boldsymbol{\Xi} \cdot \mathbf{B})$. For manifold gradient descent, we calculate the Euclidean gradient and then project it onto the manifold. We begin with the first term

$$\nabla_{\boldsymbol{\Sigma}_k} \left(\sum_{n=1}^{N} r_{nk} \left[\frac{1}{2} \langle \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}_n), \boldsymbol{\Sigma}_k^{-1} \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}_n) \rangle \right] \right) = -\frac{1}{2} \sum_{n=1}^{N} r_{nk} \boldsymbol{\Sigma}_k^{-\intercal} \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}_n) \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}_n)^\intercal \boldsymbol{\Sigma}_k^{-\intercal}. \tag{23}$$

For the gradient of the normalization constant we get

$$\begin{aligned}
\nabla_{\boldsymbol{\Sigma}_k} \log(\mathcal{C}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)) &= \frac{1}{\mathcal{C}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \int_{\mathcal{M}} \nabla_{\boldsymbol{\Sigma}_k} \exp\left(\frac{1}{2} \langle \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}), \boldsymbol{\Sigma}^{-1} \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}) \rangle \right) \mathrm{d}\mathcal{M}_{\mathbf{x}} \\
&= \frac{1}{2 \cdot \mathcal{C}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \int_{\mathcal{M}} \boldsymbol{\Sigma}_k^{-\intercal} \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}) \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x})^\intercal \boldsymbol{\Sigma}_k^{-\intercal} \exp\left(-\frac{1}{2} \langle \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}), \boldsymbol{\Sigma}^{-1} \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}) \rangle \right) \mathrm{d}\mathcal{M}_{\mathbf{x}} \\
&= \frac{1}{2 \cdot \mathcal{C}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \int_{\mathcal{T}_{\boldsymbol{\mu}_k}\mathcal{M}} \boldsymbol{\Sigma}_k^{-\intercal} \mathbf{v}\mathbf{v}^\intercal g_{\boldsymbol{\mu}_k}(\mathbf{v}) \boldsymbol{\Sigma}_k^{-\intercal} \exp\left(-\frac{1}{2} \langle \mathbf{v}, \boldsymbol{\Sigma}^{-1} \mathbf{v} \rangle \right) \mathrm{d}\mathbf{v}.
\end{aligned} \tag{24}$$

Taking this together, we obtain the gradient

$$\begin{aligned}
\nabla_{\boldsymbol{\Sigma}_k}\mathcal{L} = &-\frac{1}{2} \sum_{n=1}^{N} r_{nk} \boldsymbol{\Sigma}_k^{-\intercal} \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}_n) \operatorname{Log}_{\boldsymbol{\mu}_k}(\mathbf{x}_n)^\intercal \boldsymbol{\Sigma}_k^{-\intercal} \\
&+ \frac{R_k}{2 \cdot \mathcal{C}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \int_{\mathcal{T}_{\boldsymbol{\mu}_k}\mathcal{M}} \boldsymbol{\Sigma}_k^{-\intercal} \mathbf{v}\mathbf{v}^\intercal g_{\boldsymbol{\mu}_k}(\mathbf{v}) \boldsymbol{\Sigma}_k^{-\intercal} \exp\left(-\frac{1}{2} \langle \mathbf{v}, \boldsymbol{\Sigma}^{-1} \mathbf{v} \rangle \right) \mathrm{d}\mathbf{v},
\end{aligned} \tag{25}$$

where the matrix-valued integral again stems from BQ. To project the Euclidean gradient $\nabla_{\boldsymbol{\Sigma}_k}$ onto the tangent space of a SPD matrix $\boldsymbol{\Sigma}_k$, we simply calculate $\frac{1}{2}\boldsymbol{\Sigma}_k \left(\nabla_{\boldsymbol{\Sigma}_k} + \nabla_{\boldsymbol{\Sigma}_k}^\intercal\right) \boldsymbol{\Sigma}_k$. We optimize with gradient descent and a deterministic manifold linesearch as a subroutine, which adaptively chooses its step lengths. This procedure as well as the SPD manifold are conveniently available in the `Pymanopt` (Townsend et al., 2016) library.

In sum, the optimization process is as follows: we cycle through the components K. After taking a single steepest-direction step for $\boldsymbol{\mu}_k$, we perform two gradient descent steps for $\boldsymbol{\Sigma}_k$, each of which may use up to 4 steps in the linesearch subroutine to satisfy a sufficient decrease criterion. We provide pseudocode for the covariance update in Alg. 2. The optimizer has the following hyperparameters:

Parameter	Value	Description
t_{max}	-	update each component $\mathtt{t_{max}}$ times.
$\alpha_{\boldsymbol{\mu}}^1$	-	initial stepsize for mean updates.
$\epsilon_{\nabla_{\boldsymbol{\mu}}}$	-	tolerance for mean gradients
$\epsilon_{\mathcal{L}}$	2	likelihood tolerance
$t_{max,\boldsymbol{\Sigma}}$	4	max. $\boldsymbol{\Sigma}$ linesearch steps.
α_1	1.0	initial step size ($\boldsymbol{\Sigma}$ linesearch).
c_0	0.5	sufficient decrease factor ($\boldsymbol{\Sigma}$ linesearch).
c_1	0.5	contraction factor ($\boldsymbol{\Sigma}$ linesearch)

Cells with unspecified values (-) imply that the value of the respective parameter is not equal across all experiments and problems. Experiment-specific parameter details are in A.4.

A.3. More Details on BQ

A.3.1. General BQ

Since we use the Matérn-5/2 kernel and we require further integrals for the LAND objective gradients, we use the GP as an emulator of the function we wish to model; that is, we do not calculate integrals analytically, but use extensive Monte Carlo (MC) sampling on top of the GP, which implies evaluating the posterior mean at the locations randomly drawn from the integration measure. To compute the integral without loss of precision, we use $S = 30{,}000$ samples to estimate the integrals. The time overhead and approximation error of this procedure are negligible in practice.

We optimize the marginal likelihood of the GP with respect to the hyperparameters and use their final values to initialize the next iteration, since during the optimization the function changes smoothly from each step to the next. This information is not shared across the K components, but kept separately.

Our implementation of BQ builds upon the `bayesquad` python library (Wagstaff et al., 2018), which is available at `https://github.com/OxfordML/bayesquad`.

A.3.2. DCV - Derivations and Technical Details

The DCV acquisition function is

$$\bar{u}(\mathbf{r}) = \int_0^\infty u(\beta\mathbf{r})\ \mathrm{d}\beta = \int_0^\infty \tilde{k}_{\mathcal{D}}(\beta\mathbf{r}, \beta\mathbf{r})\pi(\beta\mathbf{r})^2\ \mathrm{d}\beta, \tag{26}$$

with derivative

$$\frac{\partial}{\partial \mathbf{r}}\bar{u}(\mathbf{r}) = \int_0^\infty \beta\pi(\beta\mathbf{r})\left[2\tilde{k}_{\mathcal{D}}(\beta\mathbf{r}, \beta\mathbf{r})\frac{\partial}{\partial\beta\mathbf{r}}\pi(\beta\mathbf{r}) + \pi(\beta\mathbf{r})\frac{\partial}{\partial\beta\mathbf{r}}\tilde{k}_{\mathcal{D}}(\beta\mathbf{r}, \beta\mathbf{r})\right]\mathrm{d}\beta. \tag{27}$$

Since the integration measure is Gaussian, i.e., $\pi(\beta\mathbf{r}) = \mathcal{N}(\beta\mathbf{r}; 0, \boldsymbol{\Sigma})$, its derivative is

$$\frac{\partial}{\partial\beta\mathbf{r}}\pi(\beta\mathbf{r}) = -\pi(\beta\mathbf{r})\boldsymbol{\Sigma}^{-1}\beta\mathbf{r}. \tag{28}$$

For simplicity, we always use WSABI-L in combination with DCV, so the derivative of the variance of the warped GP is

$$\frac{\partial}{\partial\beta\mathbf{r}}\tilde{k}_{\mathcal{D}}(\beta\mathbf{r}, \beta\mathbf{r}) = \frac{\partial}{\partial\beta\mathbf{r}}\left[m_{\mathcal{D}}(\beta\mathbf{r})^2 k_{\mathcal{D}}(\beta\mathbf{r}, \beta\mathbf{r})\right] = 2m_{\mathcal{D}}(\beta\mathbf{r})k_{\mathcal{D}}(\beta\mathbf{r}, \beta\mathbf{r})\frac{\partial}{\partial\beta\mathbf{r}}m_{\mathcal{D}}(\beta\mathbf{r}) + \frac{\partial}{\partial\beta\mathbf{r}}k_{\mathcal{D}}(\beta\mathbf{r}, \beta\mathbf{r})m_{\mathcal{D}}(\beta\mathbf{r})^2. \tag{29}$$

The derivative of the DCV acquisition function is significantly more costly to evaluate than the objective, because it requires predictive gradients of the underlying GP. Instead of using a quadrature routine like `scipy.quad`, which would evaluate the integral for every dimension sequentially, we use Simpson's rule on 50 evenly spaced points between 0 and α_{max} (defined below). Since these are multiple univariate integrals of a smooth function, the errors are practically negligible.

The scalar α_{max} simultaneously constitutes an upper bound for the integration and the length of the exponential map. A bound is reasonable since longer exponential maps are slower to compute and the integration measure concentrates the mass near the center, so very far-away locations become irrelevant. For a sensible bound, we use the chi-square distribution:

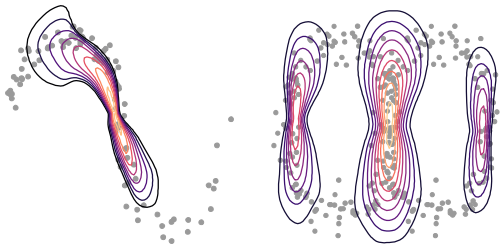
$$\langle \alpha\cdot\mathbf{r}, \boldsymbol{\Sigma}^{-1}\alpha\cdot\mathbf{r}\rangle = \chi^2_p \tag{30}$$

by choosing a high value $p = 99.5\%$, we make sure that there is no significant amount of mass outside of this isoprobability contour. Note that this limit applies only to the computation of exponential maps and the collection of observations, not to the main quadrature itself.

Since $\mathbf{r}$ is constrained to lie on the unit hypersphere, we employ manifold gradient descent with a linesearch subroutine. Conveniently, the linesearch only evaluates the objective and not its gradient, which saves a significant amount of time. Overall, optimizing this acquisition function is costly, however.

Table 1. Mean exponential map runtime in milliseconds, obtained by averaging over MC runtimes on the entire LAND fit.

CIRCLE	CIRCLE 5D	MNIST	ADK	CIRCLE 3D	CIRCLE 4D	CURLY	2-CIRCLES
60	50	238	68	32	45	62	36



(a) CURLY (b) 2-CIRCLES

Figure 12. Toy data LAND fits

For completeness, we briefly describe the geometry of the unit (hyper)sphere. If the tangent space of our data manifold is $\mathcal{T}_{\boldsymbol{\mu}}\mathcal{M} = \mathbb{R}^D$, then a direction in this tangent space is a point on $\mathbb{S}^{D-1}$, which we represent as a unit norm vector in $\mathbb{R}^D$. For a point $\mathbf{x}$ on the sphere and a tangent vector $\boldsymbol{\xi}$, which lies in the plane touching the sphere tangentially, the exponential map is $\mathrm{Exp}_{\mathbf{x}}(\boldsymbol{\xi}) = \cos(\|\boldsymbol{\xi}\|_2)\mathbf{x} + \sin(\|\boldsymbol{\xi}\|_2)\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|_2}$. However, the optimizer uses a *retraction map* $\mathrm{Retr}_{\mathbf{x}}(\boldsymbol{\xi}) = \frac{\mathbf{x}+\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|_2}$ instead of the exponential map to take a descent step. To obtain the gradient on the manifold, the Euclidean gradient is orthogonally projected onto the tangent plane.

The gradient descent is allowed a maximum of 15 steps in the "error vs. runtime experiment", whereas in the boxplot experiment we decrease this number to 5, as this experiment focuses more on speed given a fixed number of samples. The linesearch may use up to 5 steps. We set the optimism of the linesearch to 2.0 and the initial stepsize to 1.0. If a descent step has norm less than 10^{-10}, the optimization is aborted.

After an exponential map is computed according to DCV, we discretize the resulting straight line in the tangent space into 30 evenly spaced points and sequentially select 6 points using the standard WSABI objective, updating the GP after each observation.

A.4. More Details on the Experiments

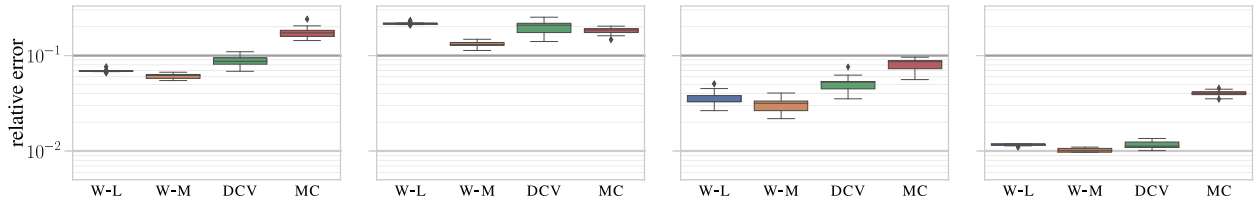
In this supplementary section, we give details about the conducted experiments and report further results, not included in the main paper due to space limitations. Fig. 13 and Fig. 15 follow the methodology as sketched in the main paper. The runtimes belonging to Fig. 13 are displayed in Fig. 14. We here also report mean runtimes of exponential maps (Tab. 1).

A.4.1. Synthetic Experiments

We created two further synthetic datasets CURLY and 2-CIRCLES that are not shown in the main paper (Fig. 12).

A.4.2. MNIST

We sampled 5,504 random data points from the first three digits of MNIST (LeCun et al., 1998), which were preprocessed by normalizing them feature-wise to $[-1, +1]$ using `sklearn.preprocessing.MinMaxScaler`. We trained a simple Variational-Autoencoder (VAE) to embed the 784 dimensional input in a latent space of dimension 2. The architecture uses separate encoders $\boldsymbol{\mu}_\phi$, $\boldsymbol{\sigma}_\phi$ and decoders $\boldsymbol{\mu}_\theta$, $\boldsymbol{\sigma}_\theta$. In summary:



(a) CIRCLE 3D (b) CIRCLE 4D (c) CURLY (d) 2-CIRCLES

Figure 13. Boxplot error comparison (log scale, shared y-axis) of BQ and MC on whole LAND fit for different manifolds. For MC, we allocate the runtime of the mean slowest BQ method. Each box contains 16 independent runs.

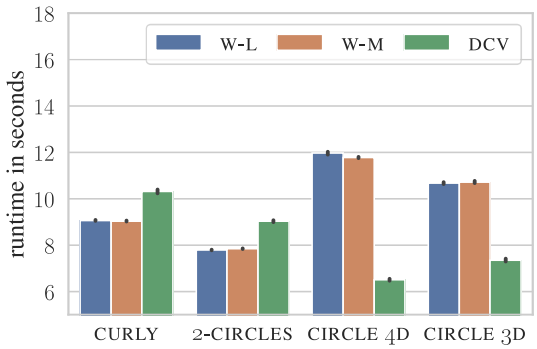
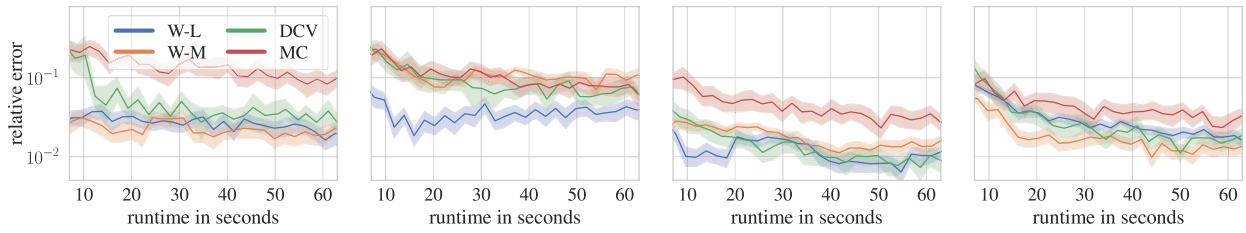


Figure 14. Mean runtime comparison (for a single integration) of the BQ methods. Errorbars indicate 95% confidence intervals w.r.t the 16 runs on each LAND fit. The reported runtimes belong to the boxplots in Fig. 13.



(a) CIRCLE 3D (b) CIRCLE 4D (c) CURLY (d) 2-CIRCLES

Figure 15. Comparison of BQ and MC errors (vertical log scale, shared legend and axes) for different manifolds, on the first integration problem of the respective LAND fit. Shaded regions indicate 95% confidence intervals w.r.t. the 30 independent runs.

Encoder/Decoder	Layer 1	Layer 2	Layer 3
$\boldsymbol{\mu}_\phi$	128 (tanh)	64 (tanh)	2 (linear)
$\boldsymbol{\sigma}_\phi$	128 (tanh)	64 (tanh)	2 (softplus)
$\boldsymbol{\mu}_\theta$	64 (linear)	128 (linear)	784 (linear)
$\boldsymbol{\sigma}_\theta$	64 (linear)	128 (linear)	784 (softplus)

We trained the network for 200 epochs using ADAM with a learning rate of 10^{-3}. The resulting latent codes were used to construct the *Aggregated Posterior Metric*, with $\rho = 0.001$, such that the measure far from the data is 1000. The small variances cause high curvature, which makes the integration tasks challenging and geodesic computations slow. To fit the LAND, we used 250 subsampled points to lower the amount of time spent on BVPS. In contrast, the GMM was fitted on the whole 5,504 points. Note that Fig. 9 shows this training data.

A.4.3. ADK

We obtained protein trajectory data of adenylate kinase from

https://www.mdanalysis.org/MDAnalysisData/adk_transitions.html#adk-dims-transitions-ensemble-dataset

(Seyler et al., 2015). We use the DIMS variant, a dataset which comprises 200 trajectories and select a subset consisting of the trajectories $160 - 200$, which contain in total 2,038 data points. To model the assumed high curvature of the trajectory space, we choose the kernel metric with $\sigma = 0.035$ and $\rho = 10^{-5}$. To visualize spatial protein structure, we used the software VMD (Humphrey et al., 1996) with the "new cartoon" representation, colored according to "residue type".

According to Seyler et al. (2015), "AdK's closed/open transition [..] is a standard test case that captures general, essential features of conformational changes in proteins". This well-studies transition involves the movement of the *LID* and *NMP* domains against the rather stable core domain. As a consequence, it can be described by two angles θ_{LID} and θ_{NMP}. In Fig. 11, it is visible how the *LID* opens to the top, whereas the *NMP* domain moves towards the bottom right (from this particular perspective).

A.4.4. General Methodology

For the aforementioned manifolds we fitted the LAND mixture model with a pre-determined component number K.

As the ground truth, we obtained $S = 40{,}000$ MC samples on each integration problem. Since obtaining a large number of exponential maps is computationally extremely expensive, we subsampled from this pool of ground truth samples when MC samples were required in the experiments, instead of running MC again. For example, in the "error vs. runtime" experiment, we calculated the mean MC runtime per sample from the ground truth pool of this particular problem and then subsampled as many samples as the given runtime limit affords. For the boxplot experiments, we averaged the MC runtimes over the whole LAND fit and always obtained the same number of samples per integration. Note that the MC runtime practically corresponds to the runtime of the exponential maps, since the overhead is minimal.

All experiments were run in a cloud setting on 8 virtual CPUs. We restricted the core usage of BLAS linear algebra subroutines to a single core, so as not to create interference between multiple processes.

A.4.5. Manifold and Optimization Hyperparameters

In Table 2, we report the relevant hyperparameters for the metrics (σ, ρ), which were used to construct the manifolds, and those optimization parameters which are not equal across all problems.

Parameter	CIRCLE	CIRCLE 3D	CIRCLE 4D	CIRCLE 5D	MNIST	ADK	CURLY	2-CIRCLES
σ	0.1	0.25	0.25	0.25	-	0.035	0.2	0.15
ρ	0.001	0.01	0.0316	0.063	0.001	0.00001	0.01	0.01
K	2	2	2	2	3	1	1	3
t_{max}	7	4	4	4	7	7	7	7
$\alpha^1_{\boldsymbol{\mu}}$	0.3	0.3	0.3	0.3	0.3	0.2	0.3	0.3
ϵ_{∇_μ}	0.01	0.01	0.01	0.01	0.015	0.01	0.01	0.01
integrations	67	39	40	34	105	36	33	111

Table 2. Manifold and LAND optimization hyperparameters and resulting number of integrations.

A.4.6. Boxplot Experiments (Fig. 5, Fig. 13)

These experiment were conducted on whole LAND fits, with 16 independent runs for each of the 3 BQ methods. From Table 2, we can easily calculate the total number of runs as $48 \cdot (67 + 39 + 40 + 34 + 105 + 36 + 33 + 111) = 22{,}320$.

A.4.7. Error vs. Runtime Experiments (Fig. 7, Fig. 15)

We evenly space 30 runtime limits between 5 and 65 seconds using `np.linspace(5., 65., 30)`. For each of these runtime limits, we let each BQ method run 30 times. BQ will stop collecting more samples as soon as the runtime limit is reached. After this, however, it will take some more time to finalize, as an ongoing computation is not interrupted. We then record the actually resulting runtimes and average over the 30 runs. These averages are then used for the x-axes of the plots, whereas the mean relative error is on the y-axes. In total, each BQ method thus has 900 runs on each problem. The 8 plots, 4 in the main paper and 4 in the supplementary, contain $3 \cdot 900 \cdot 8 = 21{,}600$ runs. Together with the boxplot experiments, we obtain $21{,}600 + 22, = 43{,}920$ BQ runs, that is, 14,640 for each of the 3 methods.

In Fig. 7(c), we removed 4 extreme DCV outliers, where seemingly the GP "broke". This amounts to $\frac{4}{21{,}600} = 0.01852\%$ of the BQ runs in the 8 plots.

Algorithm 2 LAND update$_{\mathbf{\Sigma}_k}$ on the symmetric positive definite manifold $\mathcal{S}_+$

Input: Covariance $\mathbf{\Sigma}_k^t$, mean $\boldsymbol{\mu}_k$, max. linesearch iterations $t_{max,\mathbf{\Sigma}}$, last stepsize α_k, initial stepsize $\alpha_1 = 1.0$, sufficient decrease factor $c_0 = 0.5$, contraction factor $c_1 = 0.5$
Output: $\mathbf{\Sigma}_k^{t+1}$, α_k (for reuse)

{define the exp. map on the $\mathcal{S}_+$ manifold, where $\mathbf{X}$ is an SPD matrix and $\mathbf{\Xi}$ is a tangent vector, i.e., a symmetric matrix}
Function $\mathrm{Exp}^+_{\mathbf{X}}(\mathbf{\Xi})$:
  return $\mathbf{X}^{\frac{1}{2}} \exp\left(\mathbf{X}^{-\frac{1}{2}}\mathbf{\Xi}\mathbf{X}^{-\frac{1}{2}}\right)\mathbf{X}^{\frac{1}{2}}$, where exp denotes the matrix exponential.
EndFunction
{define the norm of a vector $\mathbf{\Xi}$ in the tangent space of $\mathbf{X} \in \mathcal{S}_+$}
Function $\left(\|\cdot\|^+_{\mathbf{X}}\right)(\mathbf{\Xi})$:
  $\mathbf{X} \leftarrow \mathbf{L}\mathbf{L}^\intercal$ {cholesky decomposition}
  return $\|\mathbf{L}^{-1}\mathbf{\Xi}\mathbf{L}^{-\intercal}\|_2$
EndFunction

for $i = 1$ **to** 2 {outer gradient descent loop} **do**
  Compute (or retrieve from cache) $\mathcal{L}(\mathbf{\Sigma}_k^t)$
  Compute (or retrieve from cache) Euclidean gradient $\nabla_{\mathbf{\Sigma}_k^t}\mathcal{L}(\Sigma_k^t)$ using Eq. (25)
  Obtain manifold gradient: $g := \nabla_{\mathbf{\Sigma}_k^t;\mathcal{S}_+} = \frac{1}{2}\mathbf{\Sigma}_k^t\left(\nabla_{\mathbf{\Sigma}_k^t} + \nabla^\intercal_{\mathbf{\Sigma}_k^t}\right)\mathbf{\Sigma}_k^t$
  if α_k **is None or** $\alpha_k = 0$ **then**
    $\alpha_k \leftarrow \frac{\alpha_0}{\|g\|}$
  end if
  $\mathbf{\Sigma}_k^{t+1} \leftarrow \mathrm{Exp}^+_{\mathbf{\Sigma}_k^t}(-\alpha_k \cdot g)$
  Compute $\mathcal{C}_k(\boldsymbol{\mu}_k, \mathbf{\Sigma}_k^{t+1})$
  Evaluate LAND objective $\mathcal{L}(\mathbf{\Sigma}_k^{t+1})$
  {Linesearch subroutine}
  $j \leftarrow 1$
  while $\mathcal{L}(\mathbf{\Sigma}_k^{t+1}) > \mathcal{L}(\mathbf{\Sigma}_k^t) - c_0 \cdot \alpha_k \cdot \|g\|^2$ and $j \leq t_{max,\mathbf{\Sigma}}$ **do**
    {while no sufficient decrease, contract}
    $\alpha_k \leftarrow \alpha_k \cdot c_1$
    $\mathbf{\Sigma}_k^{t+1} \leftarrow \mathrm{Exp}^+_{\mathbf{\Sigma}_k^t}(-\alpha_k \cdot g)$
    Compute $\mathcal{C}_k(\boldsymbol{\mu}_k, \mathbf{\Sigma}_k^{t+1})$
    Evaluate LAND objective $\mathcal{L}(\mathbf{\Sigma}_k^{t+1})$
    $j \leftarrow j + 1$
  end while
  if $\mathcal{L}(\mathbf{\Sigma}_k^{t+1}) > \mathcal{L}(\mathbf{\Sigma}_k^t)$ **then**
    $\alpha_k \leftarrow 0$
  end if
  if $j \neq 2$ **then**
    $\alpha_k = 1.3 \cdot \alpha_k$ {optimism}
  end if
  $t \leftarrow t + 1$
end for