Fast and Robust Shortest Paths on Manifolds Learned from Data

Georgios Arvanitidis[†]   **Søren Hauberg**[†]   **Philipp Hennig**[‡]   **Michael Schober**[⋆]

[†]Technical University of Denmark, Lyngby, Denmark
[‡]University of Tübingen, Tübingen, Germany
[‡]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[⋆]Bosch Center for Artificial Intelligence, Renningen, Germany

Abstract

We propose a fast, simple and robust algorithm for computing shortest paths and distances on Riemannian manifolds learned from data. This amounts to solving a system of ordinary differential equations (ODEs) subject to boundary conditions. Here standard solvers perform poorly because they require well-behaved Jacobians of the ODE, and usually, manifolds learned from data imply unstable and ill-conditioned Jacobians. Instead, we propose a fixed-point iteration scheme for solving the ODE that avoids Jacobians. This enhances the stability of the solver, while reduces the computational cost. In experiments involving both Riemannian metric learning and deep generative models we demonstrate significant improvements in speed and stability over both general-purpose state-of-the-art solvers as well as over specialized solvers.

1 Introduction

A long-standing goal in machine learning is to build models that are invariant to irrelevant transformations of the data, as this can remove factors that are otherwise arbitrarily determined. For instance, in nonlinear latent variable models, the latent variables are generally unidentifiable as the latent space is by design not invariant to reparametrizations. Enforcing a Riemannian metric in the latent space that is invariant to reparametrizations alleviate this identifiability issue, which significantly boosts model performance and interpretability (Arvanitidis et al., 2018; Tosi et al., 2014). Irrelevant transformations of the data can alternatively
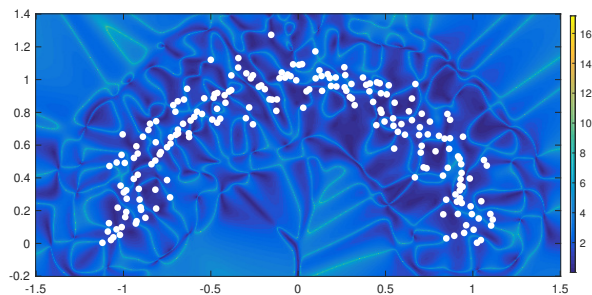
Figure 1: A data manifold and the log-condition number of the Jacobian with fixed velocity (background). High condition numbers can cause failures to converge for off-the-shelf solvers.

be factored out by only modeling local behavior of the data; geometrically this can be viewed as having a locally adaptive inner product structure, which can be learned from data (Hauberg et al., 2012). In both examples, the data is studied under a Riemannian metric, so that can be seen as living on a Riemannian manifold.

While this geometric view comes with strong mathematical support, it is not commonly adopted. The primary concern is that the computational overhead of Riemannian models often outweigh the induced benefits. The main bottleneck herein are *distance computations*, which are often at the core of machine learning algorithms. In Euclidean space, distance evaluations require the computation of a norm. In kernel methods, distances are evaluated via the kernel trick, but many interesting geometries cannot be captured with a positive definite kernel (Feragen et al., 2015). In both cases, distances can be computed with negligible effort.

The distance between two points $\mathbf{x}, \mathbf{y}$ on a Riemannian manifold is defined as the length of the *shortest path* between $\mathbf{x}$ and $\mathbf{y}$ known as the *geodesic*. Computing the geodesic requires the solution of a *boundary value problem (BVP)*. These types of *ordinary differential equations (ODEs)* require specialized numerical methods for their solution, as—unlike *initial value prob-*

lems (IVPs)—they cannot be solved via step-by-step algorithms like Runge–Kutta methods, and thus their solution is computationally *taxing* (Ascher et al., 1994).

Moreover, Riemannian manifolds learned from data usually imply high curvature and an unstable Riemannian metric. The reason is that the metric is estimated only from finite data, so it changes irregularly fast. As a consequence, the Jacobians of the ODE, which are required in many off-the-shelf solvers, are often ill-conditioned which causes additional problems and effort (Ascher et al., 1994, §8.1.2). The example in Fig. 1 shows that the Jacobian associated with a data driven manifold is highly oscillatory. This implies that standard ODE solvers easily break down. Thus, specialized BVP solvers for shortest path computations are required to make distance evaluations on manifolds learned from data as fast and robust as in other models.

In this paper, we propose a novel method for computing shortest paths on Riemannian manifolds learned from data. Combining the theory of smoothing splines/Gaussian process regression (Wahba, 1990; Rasmussen and Williams, 2006) with fixed-point iterations (Mann, 1953; Johnson, 1972), we arrive at an algorithm which is *simple*, *fast* and *more reliable* on challenging geometries. This is achieved by not utilizing the Jacobian which is computationally costly and ill-behaved.

Our work is a significant improvement of an earlier algorithm by Hennig and Hauberg (2014). Their algorithm is an early proof of concept for probabilistic numerics (Hennig et al., 2015). We show below that in the original form, it *provably* does not converge to the true solution. However, their algorithmic structure *is capable* of converging to the true solution. We demonstrate *how* their algorithm needs to be adopted to improve *solution quality* and *convergence speed*, so that the fixed-point algorithm can efficiently compute the shortest path on Riemannian manifolds.

2 A Brief Recap of Riemannian Geometry

We start by defining *Riemannian manifolds* (do Carmo, 1992). These are well-studied metric spaces, where the inner product is only locally defined and changes smoothly throughout space.

Definition 1 *A Riemannian manifold is a smooth manifold $\mathcal{M}$ where each tangent space $T_{\mathbf{x}}\mathcal{M}$ is equipped with an inner product (Riemannian metric) $\langle \mathbf{a}, \mathbf{b} \rangle_{\mathbf{x}} = \mathbf{a}^{\mathsf{T}}\mathbf{M}(\mathbf{x})\mathbf{b}$ that changes smoothly across the manifold.*

This inner product structure is sufficient for defining the length of a smooth curve $\mathbf{c} : [0,1] \rightarrow \mathcal{M}$ in the
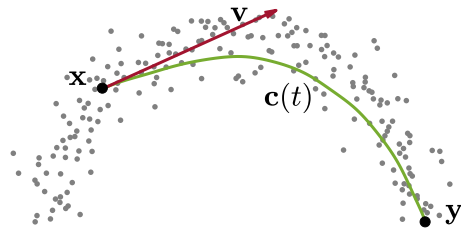

Figure 2: A data manifold with a geodesic and its tangent vector.

usual way as

$$\text{Length}(\mathbf{c}) = \int_0^1 \sqrt{\dot{\mathbf{c}}(t)^{\mathsf{T}}\mathbf{M}(\mathbf{c}(t))\dot{\mathbf{c}}(t)}\,\mathrm{d}t, \qquad (1)$$

where $\dot{\mathbf{c}} = \partial_t \mathbf{c}$ denotes the curve *velocity*. The length of the shortest path connecting two points then constitutes the natural distance measure on $\mathcal{M}$. The shortest curve is known as the geodesic, and can be found through the Euler-Lagrange equations to satisfy a system of 2nd order ODEs (Arvanitidis et al., 2018),

$$\ddot{\mathbf{c}}(t) = \frac{-\mathbf{M}(\mathbf{c}(t))^{-1}}{2}\left[2(\mathbb{I}_D \otimes \dot{\mathbf{c}}(t)^{\mathsf{T}})\frac{\partial\text{vec}[\mathbf{M}(\mathbf{c}(t))]}{\partial\mathbf{c}(t)}\dot{\mathbf{c}}(t)\right.$$
$$\left. -\frac{\partial\text{vec}[\mathbf{M}(\mathbf{c}(t))]}{\partial\mathbf{c}(t)}^{\mathsf{T}}(\dot{\mathbf{c}}(t) \otimes \dot{\mathbf{c}}(t))\right], \quad (2)$$

where $\text{vec}[\cdot]$ stacks the columns of a matrix into a vector and $\otimes$ is the Kronecker product.

Most numerical calculations on Riemannian manifolds are performed in local tangent spaces as these are Euclidean. Key operations are therefore mappings back and forth between the manifold and its tangent spaces. A point $\mathbf{y} \in \mathcal{M}$ can be mapped to the tangent space at $\mathbf{x} \in \mathcal{M}$ by computing the shortest connecting curve and evaluating its velocity $\mathbf{v}$ at $\mathbf{x}$. This is a tangent vector at $\mathbf{x}$ with the property that its length equals the length of the shortest path. By the Picard-Lindelöf theorem (Picard, 1890), this process can be reversed by solving Eq. (2) with initial conditions $\mathbf{c}(0) = \mathbf{x}$ and $\dot{\mathbf{c}}(0) = \mathbf{v}$. Mapping from the manifold to a tangent space, thus, requires solving a boundary value problem, while the inverse mapping is an initial value problem. Practically, the BVPs dominate the computational budget of numerical calculations on manifolds.

In the context of this paper we will focus upon Riemannian manifolds which are *learned* from the data, and capture its underlying geometric structure. Thus, we will consider for the smooth manifold the Euclidean space as $\mathcal{M} = \mathbb{R}^D$, and learn a Riemannian metric $\mathbf{M} : \mathbb{R}^D \rightarrow \mathbb{R}^{D \times D}$. To be clear, this simply changes the way we measure distances, while respecting the structure of the data. An illustrative example can be seen in Fig. 2.

Georgios Arvanitidis[†], Søren Hauberg[†], Philipp Hennig[‡], Michael Schober[⋆]

3   A Fast Fixed-Point Method for Shortest Paths

In order to apply Riemannian models to interesting data sets, we require a fast and robust method to solve the boundary value problem

$$\ddot{\mathbf{c}}(t) = f(\mathbf{c}(t), \dot{\mathbf{c}}(t)), \quad \mathbf{c}(0) = \mathbf{x}, \mathbf{c}(1) = \mathbf{y} \quad (3)$$

where $f(\mathbf{c}(t), \dot{\mathbf{c}}(t))$ is the right-hand side of Eq. (2) and $\mathbf{x}, \mathbf{y} \in \mathcal{M}$. Numerical BVP solvers typically replace the analytic solution $\mathbf{c}(t)$ by an approximant $\underline{\mathbf{c}}(t)$ which is required to fulfill the ODE $\underline{\ddot{\mathbf{c}}}(t_n) = f(\underline{\mathbf{c}}(t_n), \underline{\dot{\mathbf{c}}}(t_n))$ on a discrete *mesh* $\Delta = \{t_0 = 0, t_1, \ldots, t_{N-1} = 1\} \subset [0, 1]$ of evaluation *knots* t_n. Together with the boundary conditions (BC) $\underline{\mathbf{c}}(0) = \mathbf{x}$, $\underline{\mathbf{c}}(1) = \mathbf{y}$, this results in a $D(N+2)$-dimensional nonlinear equation which can be solved for rich enough $D(N+2)$-dimensional parametric models. If the approximant is represented by the posterior mean $\boldsymbol{\mu}(t)$ of a GP regressor $\mathcal{GP}(\tilde{\mathbf{c}}(t); \boldsymbol{\mu}(t), \mathbf{k}(t, s))$, then it can be proven that a solution with small approximation error exists (Wendland, 2004, §11) under suitable conditions on the kernel (Micchelli et al., 2006; Rasmussen and Williams, 2006). This means we have to find a (artificial) data set $\mathcal{D}$ with $(N+2)$ D-dimensional data points such that the posterior mean fulfills the BVP on the discretization mesh Δ and the BC on the curve *boundary* $\mathcal{B} = \{0, 1\}$. The data set $\mathcal{D}$ should be thought of as a *parametrization* for the solution curve.

One way to generate a solution is to define the auxiliary function $F(\mathbf{c}) = \ddot{\mathbf{c}} - f(\mathbf{c}, \dot{\mathbf{c}})$ and apply a variant of the Newton-Raphson method (Deuflhard, 2011) to find a root of F. For example, this type of algorithm is also at the core of MATLAB's `bvp5c`, a state-of-the-art BVP solver. The convergence of these algorithms depends on the Jacobians $\nabla_{\mathbf{c}} f, \nabla_{\dot{\mathbf{c}}} f$ of f at the evaluation knots t_n. In particular, Jacobians with big condition number may cause Newton's method to fail to converge if no precautions are taken (Ascher et al., 1994, §8.1.2). On manifolds learned from data, this is a common problem. Furthermore, in practice these Jacobians are computed with a computationally taxing finite difference scheme. Thus, a method *not* based on Newton's method should be more suitable for the computation of shortest paths.

3.1   Method Description

As mentioned above, we model the approximate solution $\underline{\mathbf{c}}(t)$ with the posterior mean $\boldsymbol{\mu}(t)$ of a (multi-output) *Gaussian process*

$$\mathcal{GP}(\tilde{\mathbf{c}}(t); \boldsymbol{\mu}(t), \mathbf{V} \otimes k(t, s)) \quad (4)$$

with (spatial) kernel k and inter-dimensional covariance matrix $\mathbf{V} \in \mathbb{R}^{D \times D}$. If the kernel k is sufficiently partially differentiable, this implies a covariance between

derivatives of $\tilde{\mathbf{c}}$ as well (Rasmussen and Williams, 2006, § 4.1.1), in particular

$$\text{cov}\left(\frac{\mathrm{d}^m}{\mathrm{d}t^m}\tilde{c}_i(t), \frac{\mathrm{d}^n}{\mathrm{d}s^n}\tilde{c}_j(s)\right) = \mathbf{V}_{ij}\frac{\partial^{m+n}}{\partial t^m \partial s^n}k(t, s) \quad (5)$$

for the covariance between output dimensions i and j, derivatives m and n and spatial inputs t and s. This *(prior) model* class is the same as in Hennig and Hauberg (2014).

The boundary equations fix two parameters $(0, \mathbf{x}), (1, \mathbf{y})$ of the parametrization. The remaining N parameters $(t_n, \ddot{\mathbf{z}}_n)$ approximate the *accelerations* $\ddot{\mathbf{c}}(t_n)$ of the true solution $\mathbf{c}(t_n)$ at knot t_n, i.e., $\ddot{\mathbf{z}}_n \approx \ddot{\mathbf{c}}(t_n)$. The $\ddot{\mathbf{z}}_n$ are updated iteratively and we denote values at the i-th iteration with the superscript (i), e.g., $\underline{\mathbf{c}}^{(i)}(t)$ for the i-th approximation, $\ddot{\mathbf{z}}_n^{(i)}$ for the i-th value of the parameter $\ddot{\mathbf{z}}_n$ and so forth.

At iteration i, the approximation $\underline{\mathbf{c}}^{(i)}(t)$ is the predictive posterior $\mathcal{GP}$

$$
\begin{aligned}
P(\tilde{\mathbf{c}}^{(i)}(t)) &= \mathcal{GP}(\tilde{\mathbf{c}}^{(i)}(t); \boldsymbol{\mu}^{(i)}(t), \mathbf{k}^{(i)}(t, s)) \\
\mathbf{G} &= \mathbf{V} \otimes \left(\begin{bmatrix} k(\mathcal{B}, \mathcal{B}) & \frac{\partial^2}{\partial s^2}k(\mathcal{B}, \Delta) \\ \frac{\partial^2}{\partial t^2}k(\Delta, \mathcal{B}) & \frac{\partial^4}{\partial t^2 \partial s^2}k(\Delta, \Delta) \end{bmatrix} \right. \\
&\qquad \left. + \text{diag}(0, 0, \boldsymbol{\Sigma}, \ldots, \boldsymbol{\Sigma}) \right) \\
\boldsymbol{\omega}^{\mathsf{T}} &= \left(\mathbf{V} \otimes \begin{bmatrix} k(t, \mathcal{B}) & \frac{\partial^2}{\partial s^2}k(t, \Delta) \end{bmatrix} \right) \mathbf{G}^{-1} \\
\boldsymbol{\mu}^{(i)}(t) &= \mathbf{m}(t) + \boldsymbol{\omega}^{\mathsf{T}}\text{vec}\left(\begin{bmatrix} \mathbf{x} - \mathbf{m}(0) \\ \mathbf{y} - \mathbf{m}(1) \\ \ddot{\mathbf{z}}_\Delta^{(i)} - \ddot{\mathbf{m}}(\Delta) \end{bmatrix}^{\mathsf{T}} \right) \\
\mathbf{k}^{(i)}(t, s) &= \mathbf{V} \otimes k(t, s) - \boldsymbol{\omega}^{\mathsf{T}}\left(\mathbf{V} \otimes \begin{bmatrix} k(\mathcal{B}, s) \\ \frac{\partial^2}{\partial t^2}(\Delta, s) \end{bmatrix} \right),
\end{aligned}
\quad (6)
$$

and similarly for the velocity $\underline{\dot{\mathbf{c}}}(t)$ by forming the $\mathbf{G}$ and $\boldsymbol{\omega}$ accordingly. In Eq. (6), $\ddot{\mathbf{z}}_\Delta \in \mathbb{R}^{N \times D}$ represents the accelerations, and $k(\mathcal{B}, \Delta) \in \mathbb{R}^{2 \times N}$ is the matrix of kernel evaluations at boundary points and evaluation knots and similar for $k(\mathcal{B}, \mathcal{B}), k(\Delta, \Delta)$ and $k(\Delta, \mathcal{B})$. Finally, $\boldsymbol{\Sigma} = \varepsilon \mathbb{I}_D$ is the identity matrix times a small regularization parameter $\varepsilon \approx 10^{-10}$, so $\ddot{\boldsymbol{\mu}}^{(i)}(t_n) \to \ddot{\mathbf{z}}_n^{(i)}$. The rationale for its inclusion will be postponed to the end of Sect. 3.2 as it will become more apparent in contrast to the model of Hennig and Hauberg (2014). Details for the choice of $k(\cdot, \cdot)$, $\mathbf{m}(\cdot)$, Δ in Appendix A. Now, we proceed with the description of the algorithm.

Just like a root of the function $F(\mathbf{c}) = \ddot{\mathbf{c}} - f(\mathbf{c}, \dot{\mathbf{c}})$ solves the ODE, so does a fixed point of the mapping $\ddot{\boldsymbol{\mu}}^{(i+1)}(t) = f(\boldsymbol{\mu}^{(i)}(t), \dot{\boldsymbol{\mu}}^{(i)}(t))$. In particular, we can evaluate this mapping on the discretization mesh Δ to map $\ddot{\mathbf{z}}_\Delta^{(i)}$ to $\ddot{\mathbf{z}}_\Delta^{(i+1)}$. The big advantage of this combination of parametrization and update scheme is the simplicity of obtaining closed-form iteration updates (Rasmussen and Williams, 2006, § 9.4). The vector field f is
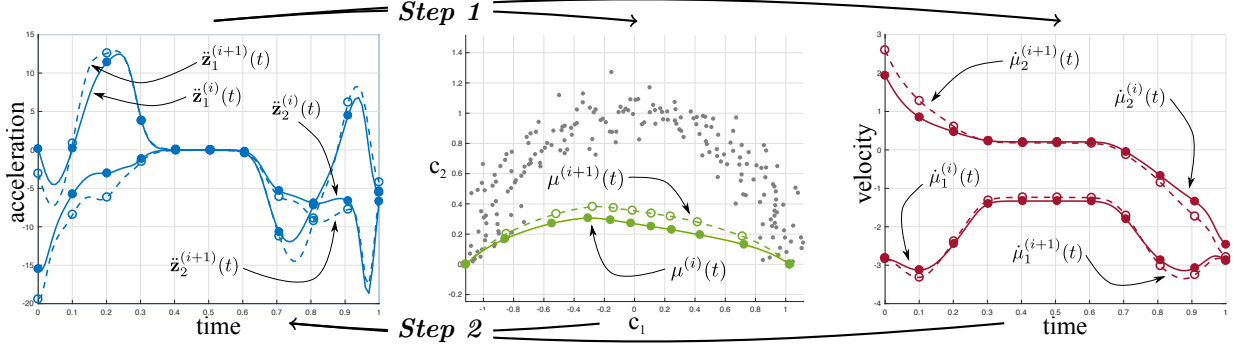
Figure 3: **Step 1**: With the current estimates of $\ddot{\mathbf{z}}_\Delta^{(i)}$ (—●—) we generate using the $\mathcal{GP}$ model Eq. (6), the current posterior curve $\boldsymbol{\mu}^{(i)}$ (—●—) and velocities $\dot{\boldsymbol{\mu}}^{(i)}$ (—●—). **Step 2**: Then, using the proposed fixed-point update scheme Eq. (7), we get the updated parameters $\ddot{\mathbf{z}}_\Delta^{(i+1)}$ (··⊙··). The algorithm iterates until $\| ·⊙· - f(·⊙·, ·⊙·)\|$ small enough.

evaluated using the current iteration $(\boldsymbol{\mu}^{(i)}, \dot{\boldsymbol{\mu}}^{(i)})$ to yield $\ddot{\mathbf{z}}_n^{(i+1)} = f(\boldsymbol{\mu}^{(i)}(t_n), \dot{\boldsymbol{\mu}}^{(i)}(t_n))$, and $\ddot{\mathbf{z}}_n^{(i+1)} \approx \ddot{\boldsymbol{\mu}}^{(i+1)}(t_n)$ because $\varepsilon \to 0$. Forming $\boldsymbol{\mu}^{(i+1)}, \dot{\boldsymbol{\mu}}^{(i+1)}$ from $\ddot{\mathbf{z}}_n^{(i+1)}$ only requires two matrix-vector products (see Eq. (6)). The process is depicted in Fig. 3.

Algorithm 1 The proposed fixed-point method.

Require: BVP $f(\mathbf{c}(t), \dot{\mathbf{c}}(t))$, hyper-parameters Δ, ε,
    tolerance τ
1:   # *Compute* $\boldsymbol{\mu}^{(i)}(t), \dot{\boldsymbol{\mu}}^{(i)}(t)$ *using Eq.* (6) *and* $\ddot{\mathbf{z}}_\Delta^{(i)}$.
2:   Define: $e_n^{(i)} \triangleq \left\| \ddot{\mathbf{z}}_n^{(i)} - f(\boldsymbol{\mu}^{(i)}(t_n), \dot{\boldsymbol{\mu}}^{(i)}(t_n)) \right\|_2^2$
3:   $\ddot{\mathbf{z}}_n^{(0)} \leftarrow \mathbf{0}, \; n = 0, \ldots, N-1$
4:   $i \leftarrow 0$
5:   **while** $\exists \, n : e_n^{(i)} > \tau$ **do**
6:      $\ddot{\mathbf{z}}_n^* \leftarrow f(\boldsymbol{\mu}^{(i)}(t_n), \dot{\boldsymbol{\mu}}^{(i)}(t_n)), \; n = 0, \ldots, N-1$
7:      **for** $j = 0, \ldots, 3$ **do**
8:          $\alpha_j = 3^{-j}$
9:          $\ddot{\mathbf{z}}_\Delta^{(*,j)} \leftarrow \alpha_j \ddot{\mathbf{z}}_\Delta^* + (1 - \alpha_j)\ddot{\mathbf{z}}_\Delta^{(i)}$
10:         **if** $\sum_n e_n^{(*,j)} \leq \sum_n e_n^{(i)}$ **then**
11:            **break**
12:         **end if**
13:      **end for**
14:      $\ddot{\mathbf{z}}_\Delta^{(i+1)} \leftarrow \ddot{\mathbf{z}}_\Delta^{(*,j)}$
15:      $i \leftarrow i + 1$
16: **end while**
17: **return** $\mathcal{GP}(\tilde{\mathbf{c}}(t); \boldsymbol{\mu}^{(i)}, \mathbf{k}^{(i)})$

Variants of this scheme have been repeatedly applied for the creation of probabilistic differential equation solvers (Hennig and Hauberg, 2014; Chkrebtii et al., 2016; Schober et al., 2014; Cockayne et al., 2016; Kersting and Hennig, 2016; Teymur et al., 2016; Schober et al., 2018; Kersting et al., 2018). Of these papers, only Hennig and Hauberg (2014) points out that this can be updated multiple times, but even there the connection between a fixed point of the mapping and an approximate solution

is not stated. Interpreting the iteration as a fixed point search is the key insight of this paper.

We suggest to apply a *Mann iteration* (Mann, 1953; Johnson, 1972) process for the solution of (2) given by

$$\ddot{\mathbf{z}}_n^* = f(\boldsymbol{\mu}^{(i)}(t_n), \dot{\boldsymbol{\mu}}^{(i)}(t_n))$$
$$\ddot{\mathbf{z}}_n^{(i+1)} = \alpha_i \ddot{\mathbf{z}}_n^* + (1 - \alpha_i)\ddot{\mathbf{z}}_n^{(i)} \qquad (7)$$

with "step sizes" $\alpha_i \in [0, 1]$. The results of Mann (1953); Johnson (1972) only apply if $\alpha_i = (i+1)^{-1}$, however we found a backtracking scheme to be effective in practice.

Algorithm 1 presents our method in pseudo-code where $\ddot{\mathbf{z}}_\Delta^{(*,j)}$ denotes the tentative parametrization. Note how the backtracking line search for α_i (Lines 7-13) requires half of the description.

Our method is similar to a recently proposed method by Bello et al. (2017) that is based on the *variational iterative method*[1] by He (2000). Bello et al. (2017) proposed to use this scheme *symbolically* requiring a computer-algebra system for its execution, which makes it inapplicable to practical tasks. More details to these related works can be found in Jafari (2014); Khuri and Sayfy (2014).

3.2 Comparison with Hennig and Hauberg (2014)

The proposed method is inspired by the previous work of Hennig and Hauberg (2014) and we make a direct comparison here.

The algorithm of Hennig and Hauberg (2014) is a proof-of-concept probabilistic numerical method (Hennig et al., 2015) for solving boundary value problems. It is structurally similar to other early probabilistic IVP solvers of Chkrebtii et al. (2016) and Skilling (1991). Since the publication of these early works, the field

---

[1] *calculus of variations* not *variational inference*.

Georgios Arvanitidis[†], Søren Hauberg[†], Philipp Hennig[‡], Michael Schober[⋆]

has matured significantly, providing algorithms with novel functionality (Hauberg et al., 2015; Mahsereci and Hennig, 2015; Oates et al., 2017; Xi et al., 2018) and rigorous analysis (Briol et al., 2015; Chkrebtii et al., 2016; Schober et al., 2018; Kersting et al., 2018).

Their main idea is to treat the vector-field evaluations $\ddot{\mathbf{z}}_n$ as *noisy observations* of the true, but unknown, second derivative $\ddot{\mathbf{c}}(t_n)$. For a concrete suggestion, they propose a Gaussian likelihood $P(\ddot{\mathbf{z}}_n) = \mathcal{N}(\ddot{\mathbf{z}}_n; \ddot{\mathbf{c}}(t_n), \boldsymbol{\Lambda}_n)$. Together with a GP prior on $\mathbf{c}(t_n)$, they arrive at an inference algorithm. Heuristically, they propose to add mesh observations sequentially, refine them iteratively for a fixed number of steps, and they repeat the overall process until they find a set of hyper-parameters of the GP which maximizes the data likelihood of the final approximation.

However, the algorithm of Hennig and Hauberg (2014) cannot converge to the true solution in general. A convergent method is required to satisfy $f(\boldsymbol{\mu}^{(i)}(t_n), \dot{\boldsymbol{\mu}}^{(i)}(t_n)) \to \ddot{\boldsymbol{\mu}}^{(i)}(t_n)$ as $i \to \infty$. However, as $\boldsymbol{\Sigma}_n \neq \mathbf{0}$ in Hennig and Hauberg (2014), $\ddot{\boldsymbol{\mu}}^{(i)}(t_n)$ is *not* an interpolant of $\ddot{\mathbf{z}}_n^{(i)}$ (Kimeldorf and Wahba, 1970, Thm. 3.2; Kanagawa et al., 2018, Prop. 3.6) implying that the *true accelerations* $\ddot{\mathbf{c}}(t_n)$ cannot be a parameterization in the model of Hennig and Hauberg (2014) contradicting the fixed point requirement.

The same criticism could be applied to our model, as we propose $\boldsymbol{\Sigma}_n = \boldsymbol{\Sigma} = \varepsilon \mathbb{I}_D \neq \mathbf{0}$. We have experimented with annealing schemes for this hyper-parameter $\boldsymbol{\Sigma}^{(i)} = i^{-1}\boldsymbol{\Sigma}$, but the benefit of $\varepsilon > 0$ for the stability of the Gram matrix $\mathbf{G}$ is bigger than induced numerical inprecision, in particular when the tolerance τ is considerably larger than ε.

Although the algorithm of Hennig and Hauberg (2014) cannot converge, three insights and resulting modifications lead to our proposed method:

1. Hennig and Hauberg (2014) did not propose a principled scheme to determine the number of refinements S, but treated it as a hyper-parameter that must be provided by the user. However, it can be easily checked whether the posterior mean $\boldsymbol{\mu}^{(i)}(t)$ fulfills the differential equation at any point t. This not only removes a hyper-parameter, but can also gives more confidence in the returned solution. In principle, the error $e_n^{(i)} = \left\| \ddot{\boldsymbol{\mu}}^{(i)}(t_n) - f(\boldsymbol{\mu}^{(i)}(t_n), \dot{\boldsymbol{\mu}}^{(i)}(t_n)) \right\|_2^2$ could even be used to construct adaptive meshes $\Delta^{(i)}$ (Mazzia and Trigiante, 2004).

2. Using an universal kernel (Micchelli et al., 2006) and a fine enough mesh Δ^*, it is known that any curve can be fitted. While sub-optimal kernel pa-

rameters θ might require an exponential bigger mesh (Vaart and Zanten, 2011, Thm. 10), the property of universality holds *regardless* of the hyper-parameter θ used to find the approximation. As a consequence, tuning hyper-parameters is *purely optional* and certainly *does not require restarts*. This also improves runtime significantly as the Gram matrix needs only be to inverted *once*. In practice, we have observed negligible solution improvements after type-II maximum likelihood optimization after the end of the algorithm.

3. Currently, there is no analysis when or if coordinate-wise updates offer improvements over simultaneous updates for all parameters $\ddot{\mathbf{z}}_\Delta^{(i)}$. There is, however, a strong argument *for* updating simultaneously: runtime. All predictive posterior parameters can be pre-computed and kept fixed throughout the runtime of the algorithm, if the mesh is not adapted throughout. In particular, the regression weights $\boldsymbol{\omega}^\mathsf{T}$ can be kept fixed, so each update requires only two vector-vector products.

Finally, while our derivation does not make use of its probabilistic interpretation, further steps in this direction could potentially unlock novel functionality which has repeatedly been the case with other probabilistic numerical methods (Xi et al., 2018; Oates et al., 2017; Hauberg et al., 2015).

4 Experiments

In this section we demonstrate the advantages of our method compared to MATLAB's `bvp5c`, and the algorithm in Hennig and Hauberg (2014) denoted in the experiments as H&H. Since all the methods depend on a set of parameters, we will come up with a *default* setting. For the proposed method, we will use for the Δ a uniform grid of $N = 10$ points including the boundaries. The corresponding noise term of the points $\ddot{\mathbf{z}}$ will be kept fixed to $\boldsymbol{\Sigma} = 10^{-7}\mathbb{I}_D$. For the GP we will use the Squared Exponential kernel $k(t,t') = \exp(-(2\lambda^2)^{-1}|t-t'|^2)$. We fix the amplitude $\mathbf{V}$ in a Bayesian fashion as Hennig and Hauberg (2014), and the length-scale $\lambda^2 \approx 2^{-1}|t_{n+1} - t_n|$ which provides enough degrees-of-freedom while covering the entire interval at the same time. The prior mean is set to the straight line $\mathbf{m}(t) = \mathbf{c}(0) + t \cdot (\mathbf{c}(1) - \mathbf{c}(0))$, and the derivative accordingly. For the method of Hennig and Hauberg (2014), we use the same parameters as for our method. We set the bounds on the Jacobian to $U = \dot{U} = 10$ and run the method with one refinement iteration. We set the maximum mesh size for the `bvp5c` to 1000, and use for the starting mesh uniformly 10 points on the straight line connecting the boundary
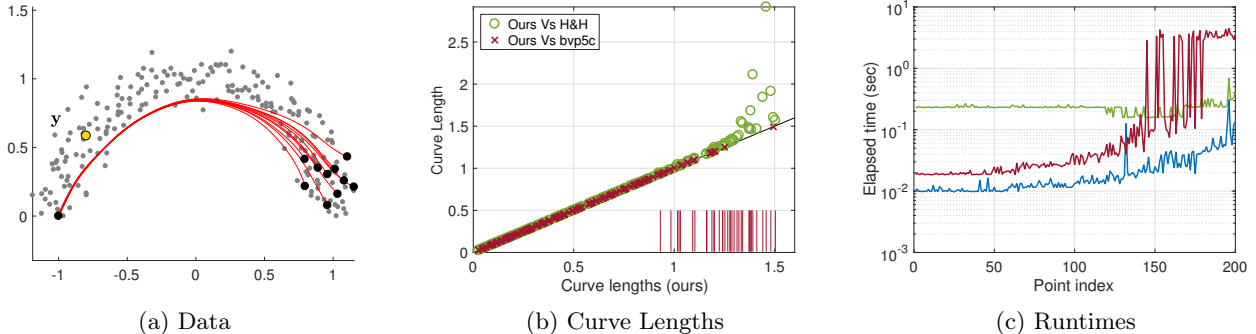
| (a) Data | (b) Curve Lengths | (c) Runtimes |

Figure 4: *Left*: Generated data on a semi-circle, together with some challenging geodesics computed by our method, and a point **y**. *Middle*: The curve lengths of the geodesics between the given data and the point **y**, on the horizontal axis we have the point index, and the vertical lines represent the failures of the `bvp5c` to converge. *Right*: The runtime for the corresponding geodesic problems.

points. For all the methods we consider the resulting curve as correct if $\|\ddot{\mathbf{c}}(t_n) - f(\mathbf{c}(t_n), \dot{\mathbf{c}}(t_n))\|_2^2 \leq 0.1$, $\forall n$.

4.1 Experiments with a Non-parametric Riemannian Metric

We first consider the case of Riemannian metric learning as proposed by Arvanitidis et al. (2016). This can be seen as a way to capture the local density of the data, and thus to uncover the underlying geometric structure. The metric at a given point is computed in three steps: 1) use a kernel to assign weight to given data, 2) compute the local diagonal covariance matrix, and 3) use its inverse as the metric tensor. More formally, the metric is

$$\mathbf{M}_{dd}(\mathbf{x}) = \left(\sum_{n=1}^{N} w_n(\mathbf{x})(x_{nd} - x_d)^2 + \rho \right)^{-1}, \quad (8)$$

$$\text{where} \quad w_n(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}_n - \mathbf{x}\|_2^2}{2\sigma_{\mathcal{M}}^2} \right).$$

The parameter $\sigma_{\mathcal{M}} \in \mathbb{R}$ controls the curvature of the manifold, since it regulates the rate of the metric change i.e. when the $\sigma_{\mathcal{M}}$ is small then the metric changes fast so the curvature increases. The parameter $\rho \in \mathbb{R}_{>0}$ is fixed such that to prevent zero diagonal elements.

We generated 200 data points along the upper semi-circle and added Gaussian noise $\mathcal{N}(0, 0.01)$ as it is shown in Fig. 4a. The $\rho = 0.01$ is kept fixed in all the experiments. We set the parameter $\sigma_{\mathcal{M}} = 0.15$, and after fixing the point $\mathbf{c}(0) = \mathbf{y}$ we compute the geodesics to the given data. From the results (Fig. 4b) we see that all three methods perform well when the distance from the starting point is small. However, when distances increase, we see that only our method manages to find the correct curve, while `bvp5c` is not able to solve the problem, and H&H finds too long

curves. Also, we see that the runtime of our method (Fig. 4c) is increased only slightly for the difficult problems, while `bvp5c` is always slower and especially for the difficult problems increases the mesh size to the maximum and fails to converge. The performance of the H&H remains almost constant, since it is essentially based on the converge of the model's parameters and not the difficulty of the problem.
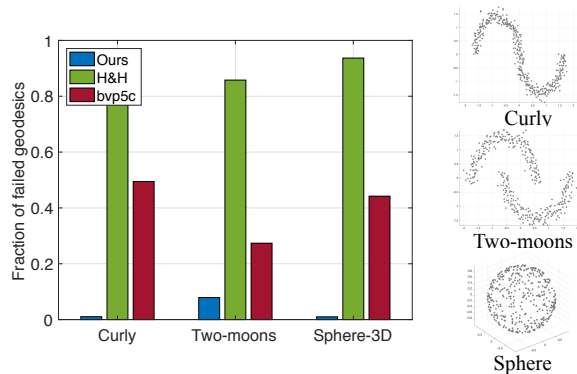

Figure 5: Failed geodesics.

Next, we generated three challenging datasets consisting of 400 points each. For the first one we generate a circle and we flip the lower half along the y axis, we refer to it as Curly in the results. For the second we move the lower half of the circle such that to get the two moons. The third one is a 2-dimensional sphere in $\mathbb{R}^3$. Finally, we add Gaussian noise $\mathcal{N}(0, 0.01)$, and we standardize to zero mean and unit variance each dimension. We keep the same parameters for the metric. Note that the resulting manifold implies high curvature, so we increased the flexibility of the methods. For the proposed model and H&H we used a grid of 50 points, and the maximum mesh size of `bvp5c` was set to 5000. We pick randomly 40 points for each dataset, and compute the pairwise distances. In Fig. 5 we see

Georgios Arvanitidis[†], Søren Hauberg[†], Philipp Hennig[‡], Michael Schober[⋆]

that our method manages to solve almost all of the shortest path problems, while `bvp5c` fails in almost half of them. H&H is expected to fail in many cases, since it is not designed to converge to the correct solution.
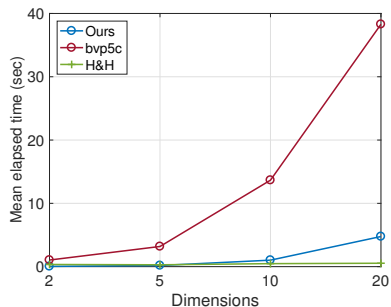

D	2	5	10	20
Ours	0%	0%	0%	9%
bvp5c	9%	24%	42%	50%
H&H	69%	95%	100%	100%

Figure 6: Scalability in higher dimensions and failures.

Furthermore, we tested the scalability of the methods with respect to dimensionality. We generate 1000 points on a semi-circle in 2 dimensions, and standardized to zero mean and unit variance. We fix a point $\mathbf{y}$ and a subset $\mathcal{S}$ of 100 points. Then, for every $D = [2, 5, 10, 20]$ we construct an orthogonal basis to project the data in $\mathbb{R}^D$, where we standardize the data again and add Gaussian noise $\mathcal{N}(0, 0.01)$. Then, for each dimension D we compute the geodesics between the $\mathbf{y}$ and the subset $\mathcal{S}$ of the points. Keep in mind that the parameter $\sigma_{\mathcal{M}} = 0.25$ is kept fixed, so as the dimensions increase the sparsity of the data increase, and so does the curvature of the manifold. In Fig. 6 we show the average runtime for every dimensionality and the percentage of failures for each method. Our method remains fast in higher dimensions, even if the curvature of the manifold is increased. On the other hand, we observe that `bvp5c` fails more often, which causes the overhead in the runtime. H&H remains fast, but, the resulting curves cannot be trusted as the criterion of correct solution is almost never met.
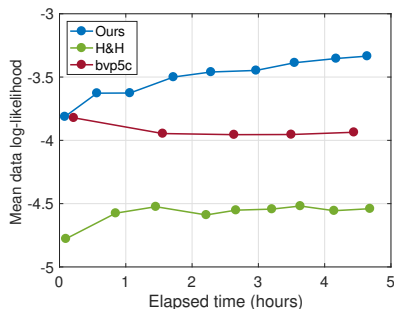

Figure 7: LAND experiment.

Also, we fitted a mixture of LANDs (Arvanitidis et al., 2016) using the three models, on the two moons dataset generated by flipping and translating the data in Fig. 4a. Note that we fix $\sigma_{\mathcal{M}} = 0.1$ since we want our metric to capture precisely the underlying structure of the data, which implies that the curvature is increased. From the results in Fig. 7 we see that the proposed solver faster achieves higher log-likelihood. Additionally, in the same time interval, it manages to run more iterations (dots in the figure).

4.2 Experiments with a Riemannian Metric of Deep Generative Models

The Variational Auto-Encoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014), provides a systematic way to learn a low dimensional latent representation of the data, together with a generator that learns to interpolate the data manifold in the input space. Usually, deep neural networks are used to model the generator. These flexible models are able to compensate for any reparametrization of the latent space, which renders the latent space unidendifiable. Recently, Arvanitidis et al. (2018) defined a Riemannian metric in the latent space, which is induced by the generator and is invariant to the parametrization of the latent space. This resolves the identifiability issue and makes computations in the latent space parametrization invariant. More specifically, the VAEs utilizes a stochastic generator that maps a point $\mathbf{x}$ from the latent space $\mathcal{X}$ to a point $\mathbf{y}$ in the input space $\mathcal{Y}$, and it consists of two parts: the mean and the variance function as $\mathbf{y}(\mathbf{x}) = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\sigma}(\mathbf{x}) \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, \mathbb{I}_{\dim(\mathcal{Y})})$ and $\odot$ is the pointwise multiplication. This stochastic mapping introduces a random Riemannian metric in the latent space. However, as it is shown (Tosi et al., 2014) we are able to use the expectation of the metric which has the appealing form

$$\mathbf{M}(\mathbf{x}) = \mathbf{J}_{\boldsymbol{\mu}}(\mathbf{x})^{\intercal} \mathbf{J}_{\boldsymbol{\mu}}(\mathbf{x}) + \mathbf{J}_{\boldsymbol{\sigma}}(\mathbf{x})^{\intercal} \mathbf{J}_{\boldsymbol{\sigma}}(\mathbf{x}), \qquad (9)$$

where $\mathbf{J}$ stands for the Jacobian of the corresponding functions. The interpretation of the metric is relatively simple. It represents the distortions due to the mean function and the uncertainty of the generator.

In this context, for the data in the input space we generated the upper half of a 2-dim sphere in $\mathbb{R}^3$, added Gaussian noise $\mathcal{N}(0, 0.01)$ and scaled the data in the interval $[-1, 1]^3$. Then, we trained a VAE, using for the generator a simple deep network consisted of two hidden layers with 16 units per layer, the `softplus` as activation functions, and `tanh` for the output layer. We used a 2-dimensional latent space, and the encoded data can be seen in Fig. 8a. There, we show the computed geodesic for the 3 methods. Interestingly, we see that the 3 resulting curves differ, and the estimated
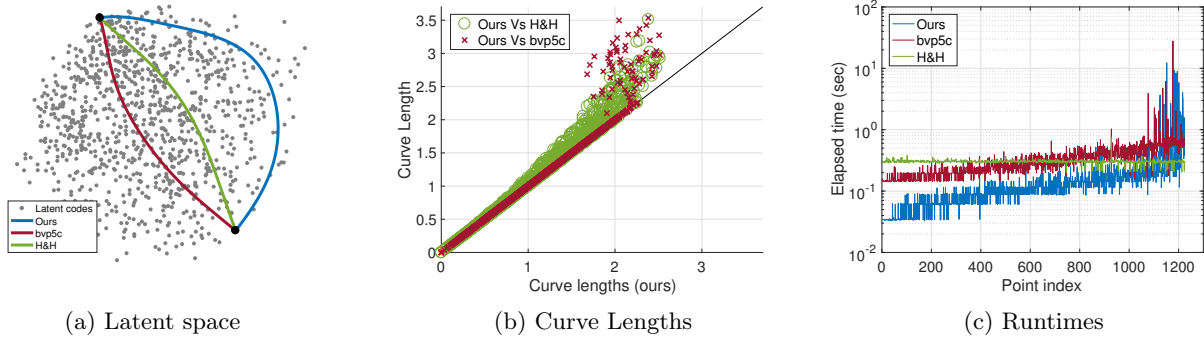
(a) Latent space   (b) Curve Lengths   (c) Runtimes

Figure 8: *Left*: The latent space together with the computed geodesics between two points. *Middle*: The curve lengths, on the x axis we show the curve length of the proposed model. The results are sorted with respect to our model. *Right*: The corresponding runtimes.

curve lengths are: proposed (2.52), `bvp5c` (3.65) and H&H (3.30). Our model, manages to find the shortest path, which is a particularly curved path but the second derivative remains relatively smooth, while `bvp5c` finds a simpler curve with larger length. This is not surprising since `bvp5c` prefers solutions where the curve is smoother, while our method prefers curves where the second derivative is smoother. In order to further analyse this behavior, we randomly pick 50 points and compute all pairwise distances. The results in Fig. 8b shows that the proposed method manages to find always the shortest path, while the other methods when the distances increase, provide a suboptimal solution. Comparing the runtimes (Fig. 8c) we see that our method is faster in the simple problems, and has only a small overhead in the difficult problems, however, it manages always to find the shortest path.
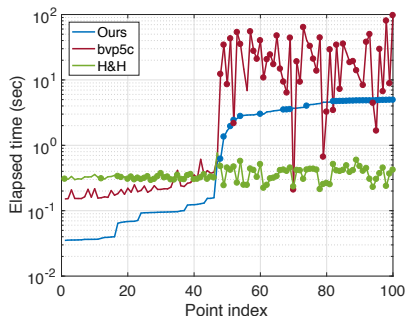

Figure 9: Runtime comparison

As a last experiment, we generated a 2-dimensional sphere in $\mathbb{R}^3$ and moved the upper half by 1, and again added Gaussian noise and scaled the data to $[-1, 1]^3$. Instead of `softplus` in the hidden layers, we used the `tanh` activation functions which increases the curvature. Here the latent space is 2-dimensional. We fix randomly a point in the latent space and compute the geodesic to 100 randomly chosen points. As we observe from the

results in Fig. 9, as long as we compute the distance between points of the same semi-sphere the runtimes of our method and `bvp5c` are comparable. However, when the points belong in different semi-spheres the runtimes increase significantly. The reason is that curvature increases dramatically when we cross parts of the latent space where the generator is uncertain. That is also the reason why many problems are not solved (dots in the figure), but even in this challenging setting our model is more robust.

5 Conclusions

We proposed a simple, fast and robust algorithm to compute shortest paths on Riemannian manifolds learned from data. Here, standard solvers often fail due to ill-conditioned Jacobians of the associated BVP. Instead, our method applies a Jacobian-free fixed-point iterative scheme. The assumption is that the true path can be approximated smoothly by the predictive GP posterior. This solver makes the Riemannian methods more feasible since robustly in reasonable time, complex statistical models (Arvanitidis et al., 2016) can be fitted, as well as distances can be computed in challenging deep metric scenarios (Arvanitidis et al., 2018).

This has been achieved by analyzing and extending an existing probabilistic numerical solver (Hennig and Hauberg, 2014), turning it from a proof-of-concept into a principled algorithm. The presented method thus contributes both to Riemannian methods and to probabilistic numerics. Further improvements might be achieved with more complex fixed-point iterations (Ishikawa, 1974), advanced line searches (Mahsereci and Hennig, 2015), adaptive mesh selection (Mazzia and Trigiante, 2004), and improved model selection (Vaart and Zanten, 2011). Finally, by quantifying the uncertainty of the approximate solution, we can turn the model into a proper probabilistic numerical algorithm.

Georgios Arvanitidis[†], Søren Hauberg[†], Philipp Hennig[‡], Michael Schober[⋆]

Acknowledgments

References

G. Arvanitidis, L. K. Hansen, and S. Hauberg. A locally adaptive normal distribution. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

G. Arvanitidis, L. K. Hansen, and S. Hauberg. Latent space oddity: on the curvature of deep generative models. In *International Conference on Learning Representations (ICLR)*, 2018.

U. M. Ascher, R. M. Mattheij, and R. D. Russell. *Numerical solution of boundary value problems for ordinary differential equations*, volume 13. Siam, 1994.

N. Bello, A. J. Alkali, and A. Roko. A fixed point iterative method for the solution of two-point boundary value problems for a second order differential equations. *Alexandria Engineering Journal*, 2017. doi: 10.1016/j.aej.2017.09.010.

F.-X. Briol, C. Oates, M. Girolami, and M. A. Osborne. Frank-wolfe bayesian quadrature: Probabilistic integration with theoretical guarantees. In *Advances in Neural Information Processing Systems*, pages 1162–1170, 2015.

O. A. Chkrebtii, D. A. Campbell, B. Calderhead, and M. A. Girolami. Bayesian solution uncertainty quantification for differential equations. *Bayesian Anal.*, 11(4):1239–1267, 12 2016. doi: 10.1214/16-BA1017.

J. Cockayne, C. Oates, T. Sullivan, and M. Girolami. Probabilistic meshless methods for partial differential equations and bayesian inverse problems. *arXiv preprint arXiv:1605.07811*, 2016.

P. Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*. Springer Science & Business Media, 2011.

M. do Carmo. *Riemannian Geometry*. Mathematics (Boston, Mass.). Birkhäuser, 1992.

A. Feragen, F. Lauze, and S. Hauberg. Geodesic exponential kernels: When curvature and linearity conflict. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3032–3042, 2015.

S. Hauberg, O. Freifeld, and M. Black. A Geometric Take on Metric Learning. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 2033–2041. MIT Press, 2012.

S. Hauberg, M. Schober, M. Liptrot, P. Hennig, and A. Feragen. A random riemannian metric for probabilistic shortest-path tractography. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, volume 18, Munich, Germany, Sept. 2015. Springer.

J.-H. He. Variational iteration method for autonomous ordinary differential systems. *Applied Mathematics and Computation*, 114(2):115 – 123, 2000. ISSN 0096-3003. doi: 10.1016/S0096-3003(99)00104-6.

P. Hennig and S. Hauberg. Probabilistic Solutions to Differential Equations and their Application to Riemannian Statistics. In *Proc. of the 17th int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, volume 33. JMLR, W&CP, 2014.

P. Hennig, M. A. Osborne, and M. Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471(2179), 2015.

S. Ishikawa. Fixed points by a new iteration method. *Proceedings of the American Mathematical Society*, 44(1): 147–150, 1974.

H. Jafari. A comparison between the variational iteration method and the successive approximations method. *Applied Mathematics Letters*, 32:1 – 5, 2014. ISSN 0893-9659. doi: 10.1016/j.aml.2014.02.004.

G. G. Johnson. Fixed points by mean value iterations. *Proceedings of the American Mathematical Society*, 34(1): 193–194, 1972.

M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur. Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences. *ArXiv e-prints*, July 2018.

H. Kersting, T. J. Sullivan, and P. Hennig. Convergence Rates of Gaussian ODE Filters. *ArXiv e-prints*, July 2018.

H. P. Kersting and P. Hennig. Active uncertainty calibration in Bayesian ODE solvers. In Janzing and Ihlers, editors, *Uncertainty in Artificial Intelligence (UAI)*, volume 32, 2016.

S. Khuri and A. Sayfy. Variational iteration method: Green's functions and fixed point iterations perspective. *Applied Mathematics Letters*, 32:28 – 34, 2014. ISSN 0893-9659. doi: 10.1016/j.aml.2014.01.006.

G. S. Kimeldorf and G. Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41 (2):495–502, 1970.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

M. Mahsereci and P. Hennig. Probabilistic line searches for stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 181–189, 2015.

W. R. Mann. Mean value methods in iteration. *Proceedings of the American Mathematical Society*, 4(3):506–510, 1953.

F. Mazzia and D. Trigiante. A hybrid mesh selection strategy based on conditioning for boundary value ode problems. *Numerical Algorithms*, 36(2):169–187, 2004.

C. A. Micchelli, Y. Xu, and H. Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(Dec):2651–2667, 2006.

C. Oates, J. Cockayne, R. G. Aykroyd, et al. Bayesian probabilistic numerical methods for industrial process monitoring. *arXiv preprint arXiv:1707.06107*, 2017.

E. Picard. Mémoire sur la théorie des équations aux dérivées partielles et la méthode des approximations successives. *Journal de Mathématiques Pures et Appliqués*, 6:145–210, 1890.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT, 2006.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, Bejing, China, 2014.

M. Schober, D. Duvenaud, and P. Hennig. Probabilistic ODE Solvers with Runge-Kutta Means. *Advances in Neural Information Processing Systems (NIPS)*, 2014.

M. Schober, S. Särkkä, and P. Hennig. A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, Jan 2018. doi: 10.1007/s11222-017-9798-7.

J. Skilling. Bayesian solution of ordinary differential equations. *Maximum Entropy and Bayesian Methods, Seattle*, 1991.

O. Teymur, K. Zygalakis, and B. Calderhead. Probabilistic linear multistep methods. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4314–4321. Curran Associates, Inc., 2016.

A. Tosi, S. Hauberg, A. Vellido, and N. D. Lawrence. Metrics for Probabilistic Geometries. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, July 2014.

A. v. d. Vaart and H. v. Zanten. Information rates of nonparametric gaussian process methods. *Journal of Machine Learning Research*, 12(Jun):2095–2119, 2011.

G. Wahba. *Spline models for observational data*. Number 59 in CBMS-NSF Regional Conferences series in applied mathematics. SIAM, 1990.

H. Wendland. *Scattered data approximation*. Cambridge University Press, 2004.

X. Xi, F.-X. Briol, and M. Girolami. Bayesian quadrature for multiple related integrals. In *International Conference on Machine Learning*, 2018.