Probabilistic Solutions to Differential Equations and their Application to Riemannian Statistics

Philipp Hennig   **Søren Hauberg**

Max Planck Institute for Intelligent Systems, Spemannstraße 38, 72076 Tübingen, Germany

[philipp.hennig|soren.hauberg]@tue.mpg.de

Abstract

We study a probabilistic numerical method for the solution of both boundary and initial value problems that returns a joint Gaussian process posterior over the solution. Such methods have concrete value in the statistics on Riemannian manifolds, where non-analytic ordinary differential equations are involved in virtually all computations. The probabilistic formulation permits marginalising the uncertainty of the numerical solution such that statistics are less sensitive to inaccuracies. This leads to new Riemannian algorithms for mean value computations and principal geodesic analysis. Marginalisation also means results can be less precise than point estimates, enabling a noticeable speed-up over the state of the art. Our approach is an argument for a wider point that uncertainty caused by numerical calculations should be tracked throughout the pipeline of machine learning algorithms.

1 Introduction

Systems of ordinary differential equations (ODEs), defined here as equations of the form

$$c^{(n)}(t) = f(c(t), c'(t), \ldots, c^{(n-1)}(t), t) \qquad (1)$$

with $c : \mathbb{R} \to \mathbb{R}^D$ and $f : \mathbb{R}^{nD} \times \mathbb{R} \to \mathbb{R}^D$ are essential tools of numerical mathematics. In machine learning and statistics, they play a role in the prediction of future states in dynamical settings, but another area where they have a particularly central role are statistical computations on Riemannian manifolds. On such

manifolds, the distance between two points is defined as the length of the shortest path connecting them, and calculating this path requires solving ordinary differential equations. Distances are elementary, so virtually all computations in Riemannian statistics involve numerical ODE solvers (Sec. 2). Just calculating the mean of a dataset of P points, trivial in Euclidean space, requires optimisation of P geodesics, thus the repeated solution of P ODEs. Errors introduced by this heavy use of numerical machinery are structured and biased, but hard to quantify, mostly because numerical solvers do not return sufficiently structured error estimates.

The problem of solving ordinary differential equations is very well studied (Hairer et al. [1987, §1] give a historical overview), and modern ODE solvers are highly evolved. But at their core they are still based on the seminal work of Runge [1895] and Kutta [1901], a carefully designed process of iterative extrapolation. In fact, ODE solvers, like other numerical methods, can be interpreted as inference algorithms [Diaconis, 1988, O'Hagan, 1992], in that they estimate an intractable quantity (the ODE's solution) given "observations" of an accessible quantity (values of f). Like all estimators, these methods are subject to an error between estimate and truth. Modern solvers can estimate this error [Hairer et al., 1987, §II.3 & §II.4], but generally only as a numerical bound at the end of each local step, not in a functional form over the space of solutions. In this paper we study and extend an idea originally proposed by John Skilling [1991] to construct (Sec. 3) a class of probabilistic solvers for both boundary value problems (BVPs) and initial value problems (IVPs). Our version of Skilling's algorithm returns nonparametric posterior probability measures over the space of solutions of the ODE. This kind of structured uncertainty improves Riemannian statistics, where the solution of the ODE is a latent "nuisance" variable. As elsewhere, it is beneficial to replace point estimates over such variables with probability distributions and marginalise over them (Sec. 4). A second, remarkable result is that, although the probabilistic solver presented here provides additional functionality, when applied to concrete

problems in Riemannian statistics, it can actually be faster than standard boundary value problem solvers (as implemented in Matlab). One reason is that the answers returned by the probabilistic solver can be less precise, because small errors are smoothed out when marginalised over the posterior uncertainty. As the cost of Riemannian statistics is dominated by that of solving ODEs, time saved by being "explicitly sloppy" in the probabilistic sense is of great value. Sec. 5 offers some empirical evaluations. A Matlab implementation of the solver is published alongside this manuscript[1].

The focus of this text is on the concrete case of Riemannian statistics and ODEs. But we believe it to be a good example for the value of probabilistic numerical methods to machine learning and statistics more generally. Treating numerical algorithms as black boxes ignores their role as error sources and can cause unnecessary computational complexity. Our results show that algorithms designed with probabilistic calculations in mind can offer qualitative and quantitative benefits. In particular, they can propagate uncertainty through a numerical computation, providing a joint approximate distribution between input and output of the computation.

2 Riemannian Statistics

Statistics on Riemannian manifolds is becoming increasingly acknowledged as it allows for statistics in constrained spaces, and in spaces with a smoothly changing metric. Examples include analysis of shape [Freifeld and Black, 2012, Fletcher et al., 2004], human poses [Hauberg et al., 2012b] and image features [Porikli et al., 2006]. Here we focus on Riemannian statistics in *metric learning*, where a smoothly changing metric locally adapts to the data [Hauberg et al., 2012a].

Let $M(x) \in \mathbb{R}^{D \times D}$ denote a positive definite, smoothly changing, metric tensor function at $x \in \mathbb{R}^D$. It locally defines a norm $\|y\|^2_{M(x)} = y^\top M(x) y$. This norm in turn defines [Berger, 2007] the *length* of a curve $c :$ $[0,1] \to \mathbb{R}^D$ as $\mathrm{Length}(c) := \int_0^1 \|c'(t)\|_{M(c(t))} \mathrm{d}t$. For two points a, b, the shortest connecting curve, $c = \arg\min_{\hat{c}} \mathrm{Length}(\hat{c})$ is known as a *geodesic*, and defines the *distance* between a and b on the manifold. Using the Euler-Lagrange equation, it can be shown [Hauberg et al., 2012a] that geodesics must satisfy the system of homogenous second-order boundary value problems

$$c''(t) = -\frac{1}{2} M^{-1}(c(t)) \left[\frac{\partial \vec{M}(c(t))}{\partial c(t)} \right]^\top [c'(t) \otimes c'(t)]$$

$$=: f(c(t), c'(t)), \text{ with } c(0) = a, c(1) = b. \quad (2)$$

where the notation $\vec{M}$ indicates the vectorization operation returning a vector $\in \mathbb{R}^{D^2}$, formed by stacking the elements of the $D \times D$ matrix M row by row, and $\otimes$ is the Kronecker product, which is associated with the vectorization operation [van Loan, 2000]. The shorthand $f(c(t), c'(t))$ is used throughout the paper. For simple manifolds this BVP can be solved analytically, but most models of interest require numerical solutions.

The geodesic starting at a with initial direction v is given by the initial value problem

$$c''(t) = f(c(t), c'(t)), \quad c(0) = a \quad \text{and} \quad c'(0) = v. \quad (3)$$

By the Picard–Lindelöf theorem [Tenenbaum and Pollard, 1963], this curve exists and is locally unique. As v is the derivative of a curve on the manifold at a, it belongs to the *tangent space* of the manifold at a. Eq. (3) implies that geodesics can be fully described by their tangent v at $c(0)$. This observation is a corner-stone of statistical operations on Riemannian manifolds, which typically start with a point μ (often the mean), compute geodesics from there to all data points, and then perform *Euclidean* statistics on the tangents $v_{1:P}$ at μ. This implicitly gives statistics of the geodesics.

These concepts are formalised by the *exponential* and *logarithm* maps, connecting the tangent space and the manifold. The exponential map $\mathrm{Exp}_a(v)$ returns the end-point of the geodesic c satisfying Eq. (3) such that $\mathrm{Length}(c) = \|v\|$. The logarithm map $\mathrm{Log}_a(b)$ is the inverse of the exponential map. It returns the initial direction of the geodesic from a to b, normalised to the length of the geodesic, i.e.

$$\mathrm{Log}_a(b) = \frac{c'(0)}{\|c'(0)\|} \mathrm{Length}(c). \quad (4)$$

The following paragraphs give examples for how this theory is used in statistics, and how the use of numerical solvers in these problems affect the results and their utility, thus motivating the use of a probabilistic solver.

Means on Manifolds The most basic statistic of interest is the mean, which is defined as the minimiser of the sum of squared distances on the manifold. A global minimiser is called the *Frechét mean*, while a local minimiser is called a *Karcher mean* [Pennec, 1999]. In general, closed-form solutions do not exist and gradient descent is applied; the update rule of μ_k with learning rate α is [Pennec, 1999]

$$\mu_{k+1} = \mathrm{Exp}_{\mu_k} \left(\alpha \frac{1}{P} \sum_{i=1}^{P} \mathrm{Log}_{\mu_k}(x_i) \right). \quad (5)$$

One practical issue is that, in each iteration, the exponential and logarithm maps need to be computed numerically. Each evaluation of Eq. (5) requires the

solution of P BVPs and one IVP. Practitioners thus perform only a few iterations rather than run gradient descent to convergence. Intuitively, it is clear that these initial steps of gradient descent do not require high numerical precision. The solver should only control their error sufficiently to find a rough direction of descent.

Principal Geodesic Analysis A generalisation of principal component analysis (PCA) to Riemannian manifolds is *principal geodesic analysis (PGA)* [Fletcher et al., 2004]. It performs ordinary PCA in the tangent space at the mean, i.e. the *principal directions* are found as the eigenvectors of $\text{Log}_\mu(x_{1:P})$, where μ is the mean as computed by Eq. (5). The principal direction v is then mapped back to the manifold to form *principal geodesics* via the exponential map, i.e. $\gamma(t) = \text{Exp}_\mu(tv)$.

Again, this involves solving for P geodesics: one from each x_i to the mean. As the statistics are performed on the (notoriously unstable) derivatives of the numerical solutions, here, high-accuracy solvers are required. These computational demands only allow for low-sample applications of PGA.

Applications in Metric Learning The original appeal of Riemannian statistics is that many types of data naturally live on known Riemannian manifolds [Freifeld and Black, 2012, Fletcher et al., 2004, Porikli et al., 2006], for example certain Lie groups. But a recent generalisation is to estimate a smoothly varying metric tensor from the data, using metric learning, and perform Riemannian statistics on the manifold implied by the smooth metric [Hauberg et al., 2012a]. The idea is to regress a smoothly changing metric tensor from learned metrics $M_{1:R}$ as

$$M(x) = \sum_{r=1}^{R} \frac{\tilde{w}_r(x)}{\sum_{j=1}^{R} \tilde{w}_j(x)} M_r, \qquad \text{where} \qquad (6)$$

$$\tilde{w}_r(x) = \exp\left(-\frac{\rho}{2}(x - \mu_r)^\top M_r (x - \mu_r)\right). \qquad (7)$$

As above, geodesics on this manifold require numeric solutions; hence this approach is currently only applicable to fairly small problems. The point estimates returned by numerical solvers, and their difficult to control computational cost, are both a conceptual and computational roadblock for the practical use of Riemannian statistics. A method returning explicit probabilistic error estimates, at low cost, is needed.

3 Solving ODEs with Gaussian process priors

Writing in 1991, John Skilling proposed treating initial value problems as a recursive Gaussian regression problem: Given a Gaussian estimate for the solution of the

ODE, one can evaluate a linearization of f, thus construct a Gaussian uncertain evaluation of the derivative of c, and use it to update the belief on c. The derivations below start with a compact introduction to this idea in a contemporary notation using Gaussian process priors, which is sufficiently general to be applicable to both boundary and initial values. Extending Skilling's method, we introduce a way to iteratively refine the estimates of the method similar to implicit Runge-Kutta methods, and a method to infer the hyperparameters of the algorithm.

The upshot of these derivations is a method returning a nonparametric posterior over the solution of both boundary and initial value problems. Besides this more expressive posterior uncertainty, another functional improvement over contemporary solvers is that these kind of algorithms allow for Gaussian uncertainty on the boundary or initial values, an essential building block of probabilistic computations.

In parallel work, Chkrebtii et al. [2013] have studied several theoretical aspects of Skilling's method thoroughly. Among other things, they provide a proof of consistency of the probabilistic estimator and argue for the uniqueness of Gaussian priors on ODE and PDE solutions. Here, we focus on the use of such probabilistic numerical methods within a larger (statistical) computational pipeline, to demonstrate the strengths of this approach, argue for its use, and highlight open research questions. We also offer a minor theoretical observation in Section 3.5, to highlight a connection between Skilling's approach and the classic Runge-Kutta framework that has previously been overlooked.

Conceptually, the algorithm derived in the following applies to the general case of Eq. (31). For concreteness and simplicity of notation, though, we focus on the case of a second-order ODE. All experiments will be performed in the Riemannian context, where the object to be inferred is the geodesic curve $c : t \in \mathbb{R} \mapsto c(t) \in \mathbb{R}^D$ consistent with Eqs. (32) or (3): Distance calculations between two points are boundary value problems of the form $c(0) = a$, $c(1) = b$. Shifts of variables along geodesics are initial value problems of the form $c(0) = a$, $c'(0) = v$ where v is the length of the geodesic. The probabilistically uncertain versions of these are

$$p(c(0), c(1)) = \mathcal{N}(c(0); a, \Delta_a)\mathcal{N}(c(1); b, \Delta_b), \text{ or} \quad (8)$$

$$p(c(0), c'(0)) = \mathcal{N}(c(0); a, \Delta_a)\mathcal{N}(c'(0); v, \Delta_v). \quad (9)$$

with positive semidefinite covariance matrices $\Delta_a, \Delta_b, \Delta_v \in \mathbb{R}^{D \times D}$.

3.1 Prior

We assign a joint Gaussian process prior over the elements of the geodesic, with a covariance function

factorizing between its input and output dependence ($c_i(t)$ is the i-th element of $c(t) \in \mathbb{R}^D$):

$$p(c(t)) = \mathcal{GP}(c; \mu_c, V \otimes k), \qquad \text{that is,} \qquad (10)$$
$$\text{cov}(c_i(t_1), c_j(t_2)) = V_{ij}k(t_1, t_2),$$

with a positive semidefinite matrix V – the covariance between output dimensions, and a positive semidefinite kernel $k : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ defining the covariance between scalar locations along the curve. The geodesic problem only involves values $t \in [0, 1]$, but this restriction is not necessary in general. The mean functions $\mu_c : \mathbb{R} \rightarrow \mathbb{R}^D$ have only minor effects on the algorithm. For BVPs and IVPs, we choose linear functions $\mu(t) = a + (b-a)t$ and $\mu(t) = a + vt$, respectively. There is a wide space of covariance functions available for Gaussian process regression [Rasmussen and Williams, 2006, §4]. In principle, every smooth covariance function can be used for the following derivations (more precisely, see Eq. (35) below). For our implementation we use the popular square-exponential (aka. radial basis function, Gaussian) kernel $k(t_1, t_2) = \exp\left[-(t_1 - t_2)^2/(2\lambda^2)\right]$. This amounts to the prior assumption that geodesics are smooth curves varying on a length scale of λ along t, with output covariance V. Sec. 3.4 below explains how to choose these hyperparameters.

Gaussian processes are closed under linear transformations. For any linear operator A, if $p(x) = \mathcal{GP}(x; \mu, k)$ then $p(Ax) = \mathcal{GP}(Ax; A\mu, AkA^\top)$. In particular, a Gaussian process prior on the geodesics c thus also implies a Gaussian process belief over any derivative $\partial^n c(t)/\partial t^n$ (see Rasmussen and Williams [2006, §9.4]), with mean function $\partial^n \mu(t)/\partial t^n$ and covariance function

$$\text{cov}\left(\frac{\partial^n c_i(t_1)}{\partial t_1^n}, \frac{\partial^n c_j(t_2)}{\partial t_2^n}\right) = V_{ij}\frac{\partial^{2n} k(t_1, t_2)}{\partial t_1^n \partial t_2^n}, \quad (11)$$

provided the necessary derivatives of k exist, as they do for the square-exponential kernel $\forall n \in \mathbb{N}$. Explicit algebraic forms for the square-exponential kernel's case can be found in the supplements.

The boundary or initial values a, b or a, v, can be incorporated into the Gaussian process belief over c analytically via the factors defined in Eqs. (37) and (9). For the case of boundary value problems, this gives a Gaussian process conditional over c with mean and covariance functions (c.f. supplements)

$$\mu_{c|a,b}(t) = \mu_c(t) + \left[V \otimes \begin{pmatrix} k_{t0} & k_{t1} \end{pmatrix}\right] \qquad (12)$$
$$\cdot \underbrace{\left[V \otimes \begin{pmatrix} k_{00} & k_{01} \\ k_{10} & k_{11} \end{pmatrix} + \begin{pmatrix} \Delta_a & 0 \\ 0 & \Delta_b \end{pmatrix}\right]^{-1}}_{=:G^{-1}} \begin{pmatrix} a - \mu_c(0) \\ b - \mu_c(1) \end{pmatrix}$$

and $\text{cov}(c(t_1), c(t_2)|a, b) = V \otimes k_{t_1, t_2}$

$$- \left[V \otimes \begin{pmatrix} k_{t_1 0} & k_{t_1 1} \end{pmatrix}\right] G^{-1}\left[V \otimes \begin{pmatrix} k_{0 t_2} \\ k_{1 t_2} \end{pmatrix}\right].$$

For IVPs, $k_{t_1 1}, k_{1 t_2}$ have to be replaced with $k'_{t_1 0}, k'_{0 t_2}$, b with v and $\mu_c(1)$ with $\mu'_c(0)$.

3.2 Likelihood

At this point we pick up Skilling's idea: Select a sequence $t_1, \ldots, t_N \in [0, 1]$ of representer points. The algorithm now moves iteratively from t_1 to t_N. At point t_i, construct estimates $[\hat{c}_i, \hat{c}'_i]$ for the solution of the ODE and its derivative. The mean (and simultaneously the most probable) estimate under the Gaussian process belief on c is the mean function, so we choose $\hat{c}_i = \mu_{c|a,b}(t_i)$, $\hat{c}'_i = \partial_t \mu_{c|a,b}(t_i)$. This estimate can be used to construct an "observation" $y_i := f(\hat{c}_i, \hat{c}'_i) \approx c''(t_i)$ of the curvature of c along the solution. For our application we use the f defined in Eq. (32). This is homogeneous, so t_i does not feature as a direct input to f, but other f's, including inhomogeneous ones, can be treated in the exact same way. The idea of constructing such approximately correct evaluations of f along the solution is also at the heart of classic ODE solvers such as the Runge-Kutta family (see Section 3.5). Skilling's idea, however, is to probabilistically describe the error of this estimation by a Gaussian distribution: Assume that we have access to upper bounds $U, U' \in \mathbb{R}^D$ on the absolute values of the elements of the Jacobians of f with respect to c and c':

$$\left|\frac{\partial f_j(c(t), c'(t))}{\partial c_i(t)}\right| < U_{ij} \quad \text{and} \quad \left|\frac{\partial f_j(c(t), c'(t))}{\partial c'_i(t)}\right| < U'_{ij} \tag{13}$$

for all $t \in [0, 1]; i, j \in (1, \ldots, D)$. In the specific case of the Riemannian problem on learned metrics, these bounds arise from analysis of Eq. (7). The uncertainty on c, c' around $\hat{c}_i, \hat{c}'_i$ is Gaussian, with covariance (see also supplements)

$$\begin{pmatrix} \Sigma_i^{cc} & \Sigma_i^{cc'} \\ \Sigma_i^{c'c} & \Sigma_i^{c'c'} \end{pmatrix} := V \otimes \begin{pmatrix} k_{t_i t_i} & k'_{t_i, t_i} \\ k'_{t_i t_i} & k''_{t_i t_i} \end{pmatrix} \tag{14}$$

$$- \left[V \otimes \begin{pmatrix} k_{t_i 0} & k_{t_i 1} \\ k'_{t_i 0} & k'_{t_i 1} \end{pmatrix}\right] G^{-1} \left[V \otimes \begin{pmatrix} k_{0 t_i} & k'_{0 t_i} \\ k_{1 t_i} & k'_{1 t_i} \end{pmatrix}\right] \in \mathbb{R}^{2D \times 2D}.$$

Considering this prior uncertainty on c, c' to first order, the error caused by the estimation of c, c' used to construct the observation y_i is bounded by a zero-mean Gaussian, with covariance (in $\mathbb{R}^{D \times D}$)

$$\Lambda_i = U^\top \Sigma_i^{cc} U + \left|U'^\top \Sigma_i^{c'c} U\right| + \left|U^\top \Sigma_i^{cc'} U'\right| + U'^\top \Sigma_i^{c'c'} U'. \tag{15}$$

This uncertain observation y_i of $c''(t_i)$ is included into the belief over c as the likelihood factor

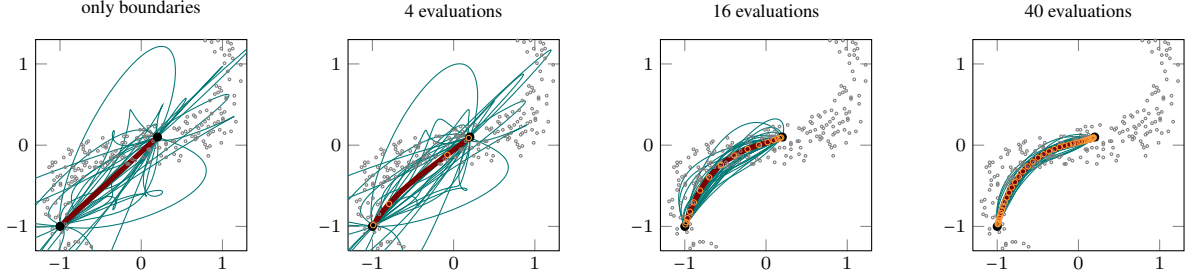$$p(y_i | c) = \mathcal{N}(y_i; c''(t_i), \Lambda_i). \tag{16}$$

Figure 1: Convergence of probabilistic ODE solver as number of evaluation points increases. Each panel shows a dataset (grey points) whose distribution determines a Riemannian manifold. The algorithm infers the geodesic connecting two points (black) on this manifold. Mean of inferred Gaussian process posterior in red. Evaluation points of the ODE in orange. Green lines are samples drawn from the posterior belief over geodesics. Left-most panel: no ODE evaluations (only boundary values). The supplements contain an animated version of this figure.

The algorithm now moves to t_{i+1} and repeats.

3.3   Posterior

After N steps, the algorithm has collected N evaluations $y_i = f(\hat{c}_i, \hat{c}'_i)$, and constructed equally many uncertainty estimates Λ_i. The marginal distribution over the y_i under the prior (34) is Gaussian with mean $[\partial^2 \mu_c(t_1)/\partial t^2, \ldots, \partial^2 \mu_c(t_N)/\partial t^2]^\top$ and covariance

$$\Gamma := V \otimes k_{TT}^{(4)} + \Lambda \qquad (17)$$

where $\quad k_{TT}^{(4)} \in \mathbb{R}^{N \times N} \quad$ with $\quad k_{TT,ij}^{(4)} = \dfrac{\partial^4 k(t_i, t_j)}{\partial t_i^2 \partial t_j^2}.$

The matrix $\Lambda \in \mathbb{R}^{DN \times DN}$ is a rearranged form of the block diagonal matrix whose N-th block diagonal element is the matrix $\Lambda_i \in \mathbb{R}^{D \times D}$. The posterior distribution over c, and the output of the algorithm, is thus a Gaussian process with mean function

$$\mu_{c|a,b,Y}(t) = \mu_c(t) + \left[V \otimes \left(k_{t0} \quad k_{t1} \quad k''_{tT} \right) \right] \cdot$$

$$\underbrace{\left[V \otimes \begin{pmatrix} k_{00} & k_{01} & k''_{0T} \\ k_{10} & k_{11} & k''_{1T} \\ k''_{T0} & k''_{T1} & k_{TT}^{(4)} \end{pmatrix} + \begin{pmatrix} \Delta_a & 0 & 0 \\ 0 & \Delta_b & 0 \\ 0 & 0 & \Lambda \end{pmatrix} \right]^{-1}}_{=: \Gamma^{-1}}$$

$$\cdot \left[a - \mu_c(0) \quad b - \mu_c(1) \quad Y - \mu''_c(T) \right]^\top \quad (18)$$

(where $k''_{0t_j} = \partial k(0, t_j)/\partial t_j^2$, etc.) and covariance function (see also supplements)

$$\mathrm{cov}(c(t_1), c(t_2) | a, b, Y) = V \otimes k_{t_1, t_2}$$

$$- \left[V \otimes \left(k_{t0} \quad k_{t1} \quad k''_{tT} \right) \right] \Gamma^{-1} \left[V \otimes \begin{pmatrix} k_{0t_2} \\ k_{1t_2} \\ k''_{Tt_2} \end{pmatrix} \right]. \quad (19)$$

Fig. 1 shows conceptual sketches. Crucially, this posterior is a joint distribution over the entire curve, not just about its value at a particular value along the curve.

This sets it apart from the error bounds returned by classic ODE solvers, and means it can be used directly for Riemannian statistics (Sec. 4).

3.3.1   Iterative Refinements

We have found empirically that the estimates of the method can be meaningfully (albeit not drastically) improved by a small number of iterations over the estimates $\hat{c}_i, \hat{c}'_i$ after the first round, without changing the Gram matrix Γ. In other words, we update the mean estimates used to construct $\hat{c}, \hat{c}'$ based on the posterior arising from later evaluations. This idea is related to the idea behind implicit Runge-Kutta methods [Hairer et al., 1987, §II.7]. It is important not to update Γ based on the posterior uncertainty in this process; this would lead to strong over-fitting, because it ignores the recursive role earlier evaluation play in the construction of later estimates.

3.4   Choice of Hyperparameters and Evaluation Points

The prior definitions of Eq. (34) rely on two hyperparameters V and λ. The output covariance V defines the overall scale on which values $c(t)$ are expected to vary. For curves within a dataset $x_{1:P} \in \mathbb{R}^D$, we assume that the curves are typically limited to the Euclidean domain covered by the data, and that geodesics connecting points close within the dataset typically have smaller absolute values than those geodesics connecting points far apart within the dataset. For curves between a, b, we thus set V in an empirical Bayesian fashion to $V = [(a-b)^\top S_x (a-b)] \cdot S_x \in \mathbb{R}^{D \times D}$, where S_x is the sample covariance of the dataset $x_{1:P}$. For IVPs, we replace $(a-b)$ in this expression by v.

The width λ of the covariance function controls the regularity of the inferred curve. Values $\lambda \ll 1$ give rise to rough estimates associated with high posterior uncertainty, while $\lambda \gg 1$ gives regular, almost-straight

curves with relatively low posterior uncertainty. Following a widely adapted approach in Gaussian regression [Rasmussen and Williams, 2006, §5.2], we choose λ to maximise its marginal likelihood under the observations Y. This marginal is Gaussian. Using the Gram matrix Γ from Eq. (25), its logarithm is

$$-2\log p(Y,a,b\,|\,\lambda) = \begin{pmatrix} a-\mu_c(0) \\ b-\mu_c(1) \\ Y-\mu_c''(T) \end{pmatrix}^\top \Gamma^{-1} \begin{pmatrix} a-\mu_c(0) \\ b-\mu_c(1) \\ Y-\mu_c''(T) \end{pmatrix} \\ + \log|\Gamma| + (2+N)\log 2\pi. \quad (20)$$

A final design choice is the location of the evaluation points t_i. The optimal choice of the grid points depends on f and is computationally prohibitive, so we use a pre-determined grid. For BVP's, we use sigmoid grids such as $t_i = 0.5(1 + \mathrm{erf}(-1.5 + 3i/N))$ as they emphasise boundary areas; for IVP's we use linear grids. But alternative choices give acceptably good results as well.
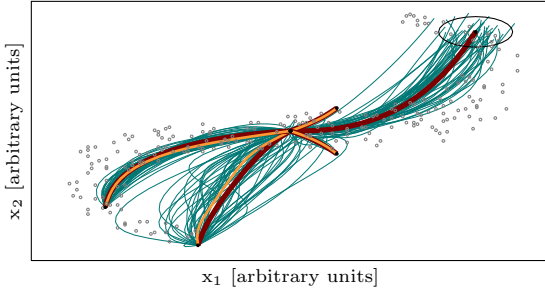

Figure 2: Solving five BVPs within a dataset (grey points). Orange: solutions found by Matlab's `bvp5c`. For the probabilistic solver, 30 posterior samples (thin green), and mean as thick red line. Simpler short curves lead to confident predictions, while the challenging curve to the bottom left has drastic uncertainty. Top right curve demonstrates the ability, not offered by classic methods, to deal with uncertain boundary values (the black ellipse represents two standard deviations).

Cost Because of the inversion of the $((N+2)D) \times ((N+2)D)$ matrix Γ in Eq. (25), the method described here has computational cost $\mathcal{O}(D^3N^3)$. This may sound costly. However, both N and D are usually small numbers for problems in Riemannian statistics. Modern ODE solvers rely on a number of additional algorithmic steps to improve performance, which lead to considerable computational overhead. Since point estimates require high precision, the computational cost of existing methods is quite high in practice. We will show below that, since probabilistic estimates can be marginalised within statistical calculations, our algorithm in fact improves computation time.

3.5 Relationship to Runge-Kutta methods

The idea of iteratively constructing estimates for the solution and using them to extrapolate may seem ad hoc. In fact, the step from the established Runge-Kutta (RK) methods to this probabilistic approach is shorter than one might think. RK methods proceed in extrapolation steps of length h. Each such step consists of N evaluations $y_i = f(\hat{c}_i, t_i)$. The estimate $\hat{c}_i$ is constructed by a *linear* combination of the previous evaluations and the initial values: $\hat{c}_i = w_0 c_0 + \sum_{j=1}^{i-1} w_j y_j$. This is exactly the structure of the mean prediction of Eq. (25). The difference lies in the way the weights w_j are constructed. For RK, they are constructed carefully such that the error between the true solution of the ODE and the estimated solution is $\mathcal{O}(h^{-N})$. Ensuring this is highly nontrivial, and often cited as a historical achievement of numerical mathematics [Hairer et al., 1987, §II.5]. In the Gaussian process framework, the w_i arise much more straightforwardly, from Eq. (25), the prior choice of kernel, and a (currently ad hoc) exploration strategy in t. The strength of this is that it naturally gives a probabilistic generative model for solutions, and thus various additional functionality. The downside is that, in this simplistic form, the probabilistic solver can not be expected to give the same local error bounds as RK methods. In fact one can readily construct examples where our specific form of the algorithm has low order. It remains a challenging research question whether the frameworks of nonparametric regression and Runge-Kutta methods can be unified under a consistent probabilistic concept.

4 Probabilistic Calculations on Riemannian Manifolds

We briefly describe how the results from Sec. 3 can be combined with the concepts from Sec. 2 to define tools for Riemannian statistics that marginalise numerical uncertainty of the solver.

Exponential maps Given an initial point $a \sim \mathcal{N}(\bar{a}, \Sigma_a)$ and initial direction $v \sim \mathcal{N}(\bar{v}, \Sigma_v)$, the exponential map can be computed by solving an IVP (3) as described in Sec. 3. This makes direct use of the ability of the solver to deal with uncertain boundary and initial values.

Logarithm maps The logarithm map (4) is slightly more complicated to compute as the length of a geodesic cannot be computed linearly. We can, however, sample from the logarithm map as follows. Let $a \sim \mathcal{N}(\bar{a}, \Sigma_a)$ and $b \sim \mathcal{N}(\bar{b}, \Sigma_b)$ and construct the geodesic Gaussian process posterior. Sample c_s and c_s' from this process and compute the corresponding logarithm map using Eq. (4). The length of c_s is evaluated using standard

quadrature. The mean and covariance of the logarithm map can then be estimated from the samples. Resorting to a sampling method might seem expensive, but in practice the runtime is dominated by the BVP solver.

Computing Means The mean of $x_{1:P}$ is computed using gradient descent (5), so we need to track the uncertainty through this optimisation. Let $\mu_k \sim \mathcal{N}(\bar{\mu}_k, \Sigma_{\mu_i})$ denote the estimate of the mean at iteration i of the gradient descent scheme. First, compute the mean and covariance of

$$v = \frac{1}{P} \sum_i \mathrm{Log}_{\mu_k}(x_i) \qquad (21)$$

by sampling as above. Assuming v follows a Gaussian distribution with the estimated mean and covariance, we then directly compute the distribution of

$$\mu_{k+1} = \mathrm{Exp}_{\mu_k}(\alpha v). \qquad (22)$$

Probabilistic optimisation algorithms [Hennig, 2013], which can directly return probabilistic estimates for the shape of the gradient (5), are a potential alternative.

Computing Principal Geodesics Given a mean $\mu \sim \mathcal{N}(\bar{\mu}, \Sigma_\mu)$, estimate the mean and covariance of $\mathrm{Log}_\mu(x_{1:P})$ and use these to estimate the mean and covariance of the principal direction v. The principal geodesic can then be computed as $\gamma = \mathrm{Exp}_\mu(v)$.

5 Experiments

This section reports on experiments with the suggested algorithms. It begins with an illustrative experiment on the well-known MNIST handwritten data set[2], followed by a study of the running time of the algorithms on the same data. Finally, we use the algorithms for visualising changes in human body shape.

5.1 Handwritten Digits

Fig. 3 (next page, for layout reasons), left and centre, show 1000 images of the digit 1 expressed in the first two principal components of the data. This data roughly follows a one-dimensional nonlinear structure, which we estimate using PGA. To find geodesics that follow the local structure of the data, we learn local metric tensors as local inverse covariance matrices, and combine them to form a smoothly changing metric using Eq. 7. The local metric tensors are learned using a standard EM algorithm for Gaussian mixtures [McLachlan and Krishnan, 1997].

For comparison, we consider a state-of-the-art BVP solver as implemented in Matlab in `bvp5c` (a four-stage implicit Runge-Kutta method [Hairer et al., 1987,
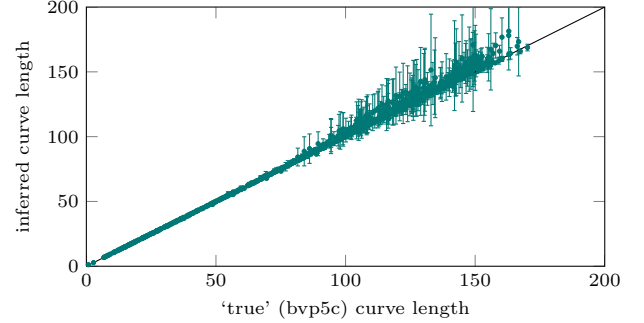
---

[2]`http://yann.lecun.com/exdb/mnist/`

Figure 4: Inferred curve lengths for the 1000 geodesics from Figure 3. The probabilistic estimates for the lengths (mean ± 2 standard deviations) are plotted against the (assumed reliable) point estimates of Matlab's `bvp5c`.
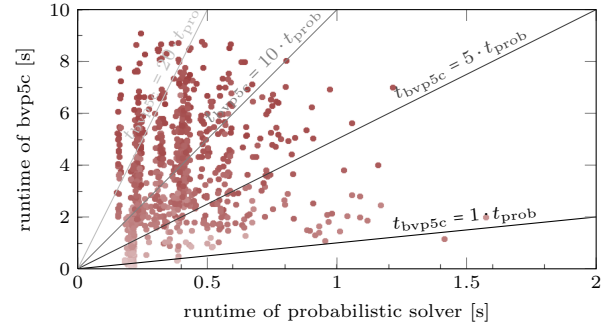

Figure 5: Runtime of `bvp5c` against that of the probabilistic solver for the same geodesics (note differing abscissa and ordinate scales). The color of points encodes the length of the corresponding curve, from 0 (white) to 200 (dark red), for comparison with Fig. 4.

§II.7]). This does not provide an uncertain output, so we resort to the deterministic Riemannian statistics algorithms (Sec. 2). Fig. 3, left, shows a few geodesics from the mean computed by our solver as well as `bvp5c`. The geodesics provided by the two solvers are comparable and tend to follow the structure of the data. Fig. 3, centre, shows the principal geodesics of the data computed using the different solvers; the results are comparable. This impression is supported by a more close comparison of the results: Figure 4 shows curve lengths achieved by the two solvers. While `bvp5c` tends to perform slightly better, in particular for long curves, the error estimates of the probabilistic solvers reflect the algorithm's imprecision quite well.

Even within this simple two-dimensional setting, the running times of the two different solvers can diverge considerably. Figure 5 plots the two runtimes against each other (along with the corresponding curve length, in color). The probabilistic solvers' slight decrease in precision is accompanied by a decrease in computational cost by about an order of magnitude. Due to
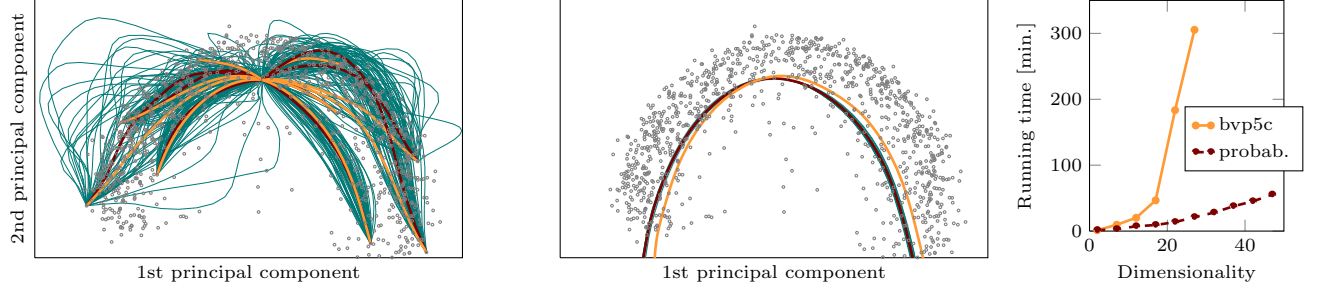
Figure 3: Results on 1000 images of the digit 1 from the MNIST handwritten digits data set, projected into its first two principal components. Left: Several geodesics from the mean computed using the probabilistic solver (red mean predictions, green samples) as well as Matlab's `bvp5c` (orange). Middle: The principal geodesics estimated using the same two solvers. Right: Running time of a mean value computation using different solvers as a function of the dimensionality.
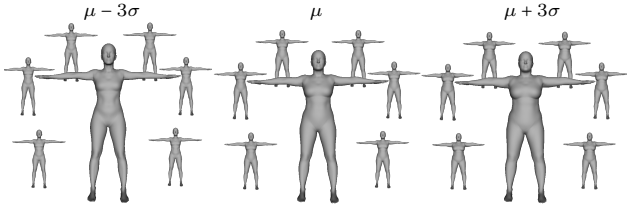

Figure 6: Samples from the posterior over the principal geodesic through the body shape dataset, estimated under a metric learned to emphasise belly circumference. Six samples each at the inferred manifold mean (centre) and three standard deviations of the dataset in either direction (left, right). *The variation is shown in the supplementary video.*

its design, the computational cost of the probabilistic solver is almost unaffected by the complexity of the problem (length of the curve). Instead, harder problems simply lead to larger uncertainty. As dimensionality increases, this advantage grows. Fig. 3, right, shows the running time for computations of the mean for an increasing number of dimensions. For this experiment we use 100 images and 5 iterations of the gradient descent scheme. As the plot shows, the probabilistic approach is substantially faster in this experiment.

5.2 Human Body Shape

We also experimented with human body shape data, using 1191 laser scans and anthropometric measurements of the belly circumference of female subjects [Robinette et al., 1999]. Body shape is represented using a 50-dimensional SCAPE model [Anguelov et al., 2005], such that each scan is a vector in $\mathbb{R}^{50}$. Following Hauberg et al. [2012a], we partition the subjects in 7 clusters by belly circumferences, learn a diagonal *Large Margin Nearest Neighbour (LMNN)* metric [Weinberger and Saul, 2009] for each cluster, and form a smoothly changing metric using Eq. 7. Under this metric vari-

ance is inflated locally along directions related to belly circumference. We visualise this increased variance using PGA. Based on the timing results above, we only consider the probabilistic algorithm here, which achieved a runtime of ~10 minutes.

Fig. 6 shows the first principal geodesic under the learned metric, where we show bodies at the mean as well as ± three standard deviations from the mean. Uncertainty is visualised by six samples from the principal geodesic at each point; this uncertainty is not described by other solvers (this is more easily seen in the supplementary video). In general, the principal geodesic shows an increasing belly circumference (again, see supplementary video).

6 Conclusion

We have studied a probabilistic solver for ordinary differential equations in the context of Riemannian statistics. The method can deal with uncertain boundary and initial values, and returns a joint posterior over the solution. The theoretical performance of such methods is currently not as tightly bounded as that of Runge-Kutta methods, but their structured error estimates allow a closed probabilistic formulation of computations on Riemannian manifolds. This includes marginalisation over the solution space, which weakens the effect of numerical errors. This, in turn, means computational precision, and thus cost, can be lowered. The resulting improvement in computation time and functionality is an example for the conceptual and applied value of probabilistically designed numerical algorithms for computational statistics.

Acknowledgements

References

D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: shape completion and animation of people. *ACM Transactions on Graphics*, 24(3):408–416, 2005.

M. Berger. *A Panoramic View of Riemannian Geometry.* Springer, 2007.

O. Chkrebtii, D.A. Campbell, M.A. Girolami, and B. Calderhead. Bayesian uncertainty quantification for differential equations. *arXiv*, stat.ME 1306.2365, June 2013.

P. Diaconis. Bayesian numerical analysis. *Statistical decision theory and related topics*, IV(1):163–175, 1988.

P.T. Fletcher, C. Lu, S.M. Pizer, and S. Joshi. Principal Geodesic Analysis for the study of Nonlinear Statistics of Shape. *IEEE Transactions on Medical Imaging*, 23(8):995–1005, 2004.

O. Freifeld and M.J. Black. Lie bodies: A manifold representation of 3D human shape. In A. Fitzgibbon et al., editor, *European Conf. on Computer Vision (ECCV)*, Part I, LNCS 7572, pages 1–14. Springer-Verlag, oct 2012.

E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I – Nonstiff Problems.* Springer, 1987.

S. Hauberg, O. Freifeld, and M.J. Black. A geometric take on metric learning. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS) 25*, pages 2033–2041. MIT Press, 2012a.

S. Hauberg, S. Sommer, and K.S. Pedersen. Natural metrics and least-committed priors for articulated tracking. *Image and Vision Computing*, 30(6-7):453–461, 2012b.

P. Hennig. Fast probabilistic optimization from noisy gradients. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

W. Kutta. Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. *Zeitschrift für Mathematik und Physik*, 46:435–453, 1901.

G.J. McLachlan and T. Krishnan. *The EM Algorithm and its Extensions.* Wiley, 1997.

A. O'Hagan. Some Bayesian numerical analysis. *Bayesian Statistics*, 4:345–363, 1992.

X. Pennec. Probabilities and statistics on Riemannian manifolds: Basic tools for geometric measurements. In *Proceedings of Nonlinear Signal and Image Processing*, pages 194–198, 1999.

F. Porikli, O. Tuzel, and P. Meer. Covariance tracking using model update based on Lie algebra. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 728–735. IEEE, 2006.

C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

K.M. Robinette, H. Daanen, and E. Paquet. The CAESAR project: a 3-D surface anthropometry survey. In *3-D Digital Imaging and Modeling*, pages 380–386, 1999.

C. Runge. Über die numerische Auflösung von Differentialgleichungen. *Mathematische Annalen*, 46:167–178, 1895.

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

M. Tenenbaum and H. Pollard. *Ordinary Differential Equations.* Dover Publications, 1963.

C.F. van Loan. The ubiquitous Kronecker product. *J of Computational and Applied Mathematics*, 123:85–100, 2000.

K.Q. Weinberger and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009.

— Supplementary Material —

A   Gaussian Process Posteriors

Equations (10), (12), (18) and (19) in the main paper are Gaussian process posterior distributions over the curve c arising from observations of various combinations of derivatives of c. These forms arise from the following general result.[3] Consider a Gaussian process prior distribution

$$p(c) = \mathcal{GP}(c; \mu, k) \tag{23}$$

over the function c, and observations y with the likelihood

$$p(y\,|\,c, A) = \mathcal{N}(y; Ac, \Lambda), \tag{24}$$

with a linear operator A. This includes the special cases of the selection operator $A = \delta(x - x_i)$ which selects function values $Ac = \int \delta(x - x_i)c(x)\mathrm{d}x = c(x_i)$, and the special case of derivative operators $\partial_x^n \delta(x - x_i)$ which give $Ac = \int \partial_x \delta(x - x_i)c(x)\mathrm{d}x = c^{(n)}(x_i)$. Then the posterior over any linear map Bc of the curve c (including $B = \delta(x - x_j)$, giving $Bc = c(x_j)$) is

$$p(Bc\,|\,y, A) = \mathcal{GP}(Bc; B\mu + BkA^\top(AkA^\top + \Lambda)^{-1}(y - A\mu), BkB - BkA^\top(AkA^\top + \Lambda)^{-1}AkB^\top). \tag{25}$$

And the marginal probability for y is

$$p(y\,|\,A) = \int p(y\,|\,c, A)p(c)\,dc = \mathcal{N}(y; A\mu, AkA^\top + \Lambda) \tag{26}$$

The classic example is that of the marginal posterior at $c(x_*)$ arising from noisy observations at $[c(x_1), \ldots, c(x_N)]^\top$. This is the case of $B = \delta(x - x_*)$ and $A = [\delta(x - x_1), \ldots, \delta(x - x_N)]^\top$, which gives

$$B\mu = \mu(x_*) \tag{27}$$

$$A\mu = [\mu(x_1), \ldots \mu(x_N)]^\top \tag{28}$$

$$BkA^\top = \left[\iint \delta(a - x_*)k(a, b)\delta(b - x_i)\,da\,db \right]_{i=1,\ldots,N} = [k(x_*, x_1), \ldots k(x_*, x_N)] \tag{29}$$

$$AkA^\top = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{pmatrix} \tag{30}$$

and so on. All the Gaussian forms in the paper are special cases with various combinations of A and B.

B   Covariance Functions

The models in the paper assume a squared-exponential (aka. radial basis function, Gaussian) covariance function between values of the function $\boldsymbol{f} : \mathbb{R} \twoheadrightarrow \mathbb{R}^N$, of the form

$$\mathrm{cov}\big(f_i(t), f_j(t')\big) = V_{ij} \exp\left(-\frac{(t - t')^2}{2\lambda^2} \right) =: V_{ij}k_{tt'} \tag{31}$$

---

[3]Equation A.6 in C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006

The calculations require the covariance between various combinations of derivatives of the function. For clear notation, we'll use the operator $\partial := \partial/\partial t$, and the abbreviation $\delta_{tt'} := (t-t')/\lambda^2$

$$\operatorname{cov}(f_i(t),\dot{f}_j(t')) = V_{ij}k_{tt'}\partial^\top = V_{ij}\frac{t-t'}{\lambda^2}k_{tt'} = V_{ij}\delta_{tt'}k_{tt'} = -\operatorname{cov}(\dot{f}_i(t),f_j(t')) \tag{32}$$

$$\operatorname{cov}(\dot{f}_i(t),\dot{f}_j(t')) = V_{ij}\partial k_{tt'}\partial^\top = V_{ij}\left(\frac{1}{\lambda^2}-\left(\frac{t-t'}{\lambda^2}\right)^2\right)k_{tt'} = V_{ij}\left(\frac{1}{\lambda^2}-\delta_{tt'}^2\right)k_{tt'} \tag{33}$$

$$\operatorname{cov}(f_i(t),\ddot{f}_j(t')) = V_{ij}k_{tt'}\partial^\top\partial^\top = V_{ij}\left(\left(\frac{t-t'}{\lambda^2}\right)^2-\frac{1}{\lambda^2}\right)k_{tt'} = V_{ij}\left(\delta_{tt'}^2-\frac{1}{\lambda^2}\right)k_{tt'} = -\operatorname{cov}(\dot{f}_i(t),\dot{f}_j(t')) \tag{34}$$

$$\operatorname{cov}(\dot{f}_i(t),\ddot{f}_j(t')) = V_{ij}\partial k_{tt'}\partial^\top\partial^\top = V_{ij}\left(\frac{2}{\lambda^2}\frac{t-t'}{\lambda^2}-\frac{t-t'}{\lambda^2}\left(\left(\frac{t-t'}{\lambda^2}\right)^2-\frac{1}{\lambda^2}\right)\right)k_{tt'} = V_{ij}\left(-\delta_{tt'}^3+\frac{3}{\lambda^2}\delta_{tt'}\right)k_{tt'} \tag{35}$$

$$\operatorname{cov}(\ddot{f}_i(t),\ddot{f}_j(t')) = V_{ij}\partial\partial k_{tt'}\partial^\top\partial^\top = V_{ij}\left(\delta_{tt'}^4-\frac{6}{\lambda^2}\delta_{tt'}^2+\frac{3}{\lambda^4}\right)k_{tt'} \tag{36}$$

Of course, all those derivatives retain the Kronecker structure of the original kernel, because $\partial(V\otimes k) = V\otimes\partial k$.

C Inferring Hyperparameters

Perhaps the most widely used way to learn hyperparameters for Gaussian process models it type-II maximum likelihood estimation, also known as evidence maximisation: The marginal probability for the observations y is $p(y\,|\,\lambda) = \int p(y\,|\,c)p(c\,|\,\lambda)\,dc = \mathcal{N}(y;\ddot{\mu}_T,\partial\partial k_{TT}(\lambda)\partial\partial+\Lambda)$. Using the shorthand $G := (\partial\partial k_{TT}(\lambda)\partial\partial+\Lambda)$, its logarithm is

$$-2\log p(y\,|\,\lambda) = (y-\ddot{\mu}_T)^\top G^{-1}(y-\ddot{\mu}_T) + \log|G| + N\log 2\pi \tag{37}$$

To optimise this expression with respect to the length scale λ, we use

$$-2\frac{\partial\log p(y\,|\,\lambda)}{\partial\lambda^2} = -(y-\ddot{\mu}_T)^\top G^{-1}\frac{\partial G}{\partial\lambda^2}G^{-1}(y-\ddot{\mu}_T) + \operatorname{tr}(G^{-1}\frac{\partial G}{\partial\lambda^2}). \tag{38}$$

From Equation (36), and using

$$\frac{\partial\delta_{tt'}}{\partial\lambda^2} = -\frac{\delta_{tt'}}{\lambda^2} \qquad\qquad \frac{\partial k_{tt'}}{\partial\lambda^2} = k_{tt'}\frac{\delta_{tt'}^2}{2} \tag{39}$$

we find

$$\frac{\partial G_{tt'}^{ij}}{\partial\lambda^2} = V_{ij}\left[\left(-\frac{4}{\lambda^2}\delta_{tt'}^4+\frac{18}{\lambda^4}\delta_{tt'}^2+\frac{6}{\lambda^6}\right)k_{tt'}+\partial\partial k_{tt'}\partial^\top\partial^\top\frac{\delta_{tt'}^2}{2}\right] \tag{40}$$

$$= V_{ij}\left(\frac{\delta^6}{2}-\frac{7}{\lambda^2}\delta_{tt'}^4+\frac{39}{2\lambda^4}\delta_{tt'}^2+\frac{6}{\lambda^6}\right)k_{tt'} \tag{41}$$

It is also easy to evaluate the second derivative, giving

$$-2\frac{\partial^2\log p(y\,|\,\lambda)}{(\partial\lambda^2)^2} = 2(y-\ddot{\mu}_T)^\top G^{-1}\frac{\partial G}{\partial\lambda^2}G^{-1}\frac{\partial G}{\partial\lambda^2}G^{-1}(y-\ddot{\mu}_T) - \operatorname{tr}\left[\frac{\partial G}{\partial\lambda^2}G^{-1}\frac{\partial G}{\partial\lambda^2}G^{-1}\right] \tag{42}$$

$$- (y-\ddot{\mu}_T)^\top G^{-1}\frac{\partial^2 G}{(\partial\lambda^2)^2}G^{-1}(y-\ddot{\mu}_T) + \operatorname{tr}\left[G^{-1}\frac{\partial^2 G}{(\partial\lambda^2)^2}\right] \tag{43}$$

$$\text{where} \qquad \frac{\partial^2 G_{tt'}^{ij}}{(\partial\lambda^2)^2} = V_{ij}\left(-\frac{3}{\lambda^2}\delta_{tt'}^6+\frac{35}{\lambda^4}\delta_{tt'}^4-\frac{78}{\lambda^6}\delta_{tt'}^2+\frac{18}{\lambda^8}\right)k_{tt'}+\frac{\delta^2}{2}\frac{\partial G_{tt'}^{ij}}{\partial\lambda^2} \tag{44}$$

This allows constructing a Newton-Raphson optimisation scheme for the length scale of the algorithm.