Journal of Machine Learning Research 17 (2016) 1-87 Submitted 7/15; Revised 6/16; Published 8/16

String and Membrane Gaussian Processes

Yves-Laurent Kom Samo YLKS@ROBOTS.OX.AC.UK

Stephen J. Roberts SJROB@ROBOTS.OX.AC.UK

Department of Engineering Science and Oxford-Man Institute
University of Oxford
Eagle House, Walton Well Road,
OX2 6ED, Oxford, United Kingdom

Editor: Neil Lawrence

Abstract

In this paper we introduce a novel framework for making exact nonparametric Bayesian inference on latent functions that is particularly suitable for *Big Data* tasks. Firstly, we introduce a class of stochastic processes we refer to as *string Gaussian processes* (*string GPs* which are not to be mistaken for Gaussian processes operating on text). We construct *string GPs* so that their finite-dimensional marginals exhibit suitable *local* conditional independence structures, which allow for *scalable*, *distributed*, and *flexible* nonparametric Bayesian inference, without resorting to approximations, and while ensuring some mild global regularity constraints. Furthermore, *string GP* priors naturally cope with heterogeneous input data, and the gradient of the learned latent function is readily available for explanatory analysis. Secondly, we provide some theoretical results relating our approach to the *standard GP paradigm*. In particular, we prove that some *string GPs* are Gaussian processes, which provides a complementary *global* perspective on our framework. Finally, we derive a scalable and distributed MCMC scheme for supervised learning tasks under *string GP* priors. The proposed MCMC scheme has computational time complexity $\mathcal{O}(N)$ and memory requirement $\mathcal{O}(dN)$, where N is the data size and d the dimension of the input space. We illustrate the efficacy of the proposed approach on several synthetic and real-world data sets, including a data set with 6 millions input points and 8 attributes.

Keywords: String Gaussian processes, scalable Bayesian nonparametrics, Gaussian processes, nonstationary kernels, reversible-jump MCMC, point process priors

1. Introduction

Many problems in statistics and machine learning involve inferring a latent function from training data (for instance regression, classification, inverse reinforcement learning, inference on point processes to name but a few). Real-valued stochastic processes, among which Gaussian processes (GPs), are often used as functional priors for such problems, thereby allowing for a full Bayesian nonparametric treatment. In the machine learning community, interest in GPs grew out of the observation that some Bayesian neural networks converge to GPs as the number of hidden units approaches infinity (Neal (1996)). Since then, other similarities have been established between GPs and popular models such as Bayesian linear regression, Bayesian basis function regression, spline models and support vector machines (Rasmussen and Williams (2006)). However, they often perform poorly on *Big Data* tasks primarily for two reasons. Firstly, large data sets are likely to exhibit multiple types of local patterns that should appropriately be accounted for by flexible and possibly

nonstationary covariance functions, the development of which is still an active subject of research. Secondly, inference under GP priors often consists of looking at the values of the GP at all input points as a jointly Gaussian vector with fully dependent coordinates, which induces a memory requirement and time complexity respectively squared and cubic in the training data size, and thus is intractable for large data sets. We refer to this approach as the *standard GP paradigm*. The framework we introduce in this paper addresses both of the above limitations.

Our work is rooted in the observation that, from a Bayesian nonparametric perspective, it is inefficient to define a stochastic process through *fully-dependent* marginals, as it is the case for Gaussian processes. Indeed, if a stochastic process $(f(x))_{x\in\mathbb{R}^d}$ has fully dependent marginals and exhibits no additional conditional independence structure then, when f is used as functional prior and some observations related to $(f(x_1),\dots,f(x_n))$ are gathered, namely $(y_1,\dots,y_n)$, the *additional* memory required to take into account an additional piece of information (y_{n+1},x_{n+1}) grows in $\mathcal{O}(n)$, as one has to keep track of the extent to which y_{n+1} informs us about $f(x_i)$ for every $i\leq n$, typically through a covariance matrix whose size will increase by $2n+1$ terms. Clearly, this is inefficient, as y_{n+1} is unlikely to be informative about $f(x_i)$, unless x_i is sufficiently close to x_{n+1}. More generally, the larger n, the less information a single additional pair (y_{n+1},x_{n+1}) will add to existing data, and yet the increase in memory requirement will be much higher than that required while processing earlier and more informative data. This inefficiency in resource requirements extends to computational time, as the *increase* in computational time resulting from adding (y_{n+1},x_{n+1}) typically grows in $\mathcal{O}(n^2)$, which is the difference between the numbers of operations required to invert a $n\times n$ matrix and to invert a $(n+1)\times(n+1)$ matrix. A solution for addressing this inefficiency is to appropriately limit the extent to which values $f(x_1),\dots,f(x_n)$ are related to each other. Existing approaches such as sparse Gaussian processes (see Quinonero-Candela and Rasmussen (2005) for a review), resort to an *ex-post* approximation of fully-dependent Gaussian marginals with multivariate Gaussians exhibiting conditional independence structures. Unfortunately, these approximations trade-off accuracy for scalability through a control variable, namely the number of inducing points, whose choice is often left to the user. The approach we adopt in this paper consists of going back to stochastic analysis basics, and constructing stochastic processes whose finite-dimensional marginals exhibit suitable conditional independence structures so that we need not resorting to *ex-post* approximations. Incidentally, unlike sparse GP techniques, the conditional independence structures we introduce also allow for flexible and principled learning of local patterns, and this increased flexibility does not come at the expense of scalability.

The contributions of this paper are as follows. We introduce a novel class of stochastic processes, string Gaussian processes (*string GPs*), that may be used as priors over latent functions within a Bayesian nonparametric framework, especially for large scale problems and in the presence of possibly multiple types of local patterns. We propose a framework for analysing the flexibility of random functions and surfaces, and prove that our approach yields more flexible stochastic processes than isotropic Gaussian processes. We demonstrate that exact inference under a *string GP* prior scales considerably better than in the *standard GP paradigm*, and is amenable to distributed computing. We illustrate that popular stationary kernels can be well approximated within our framework, making *string GPs* a scalable alternative to commonly used GP models. We derive the joint law of a *string GP* and its gradient, thereby allowing for explanatory analysis on the learned latent function. We propose a reversible-jump Markov Chain Monte Carlo sampler for automatic learning of model complexity and local patterns from data.

The rest of the paper is structured as follows. In Section 2 we review recent advances on Gaussian processes in relation to inference on large data sets. In Section 3 we formally construct *string GPs* and derive some important results. In Section 4 we provide detailed illustrative and theoretical comparisons between *string GPs* and the *standard GP paradigm*. In Section 5 we propose methods for inferring latent functions under *string GP* priors with time complexity and memory requirement that are linear in the size of the data set. The efficacy of our approach compared to competing alternatives is illustrated in Section 6. Finally, we finish with a discussion in Section 7.

2. Related Work

The two primary drawbacks of the *standard GP paradigm* on large scale problems are the lack of scalability resulting from postulating a full multivariate Gaussian prior on function values at *all* training inputs, and the difficulty postulating *a priori* a class of covariance functions capable of capturing intricate and often local patterns likely to occur in large data sets. A tremendous amount of work has been published that attempt to address either of the aforementioned limitations. However, scalability is often achieved either through approximations or for specific applications, and nonstationarity is usually introduced at the expense of scalability, again for specific applications.

2.1 Scalability Through Structured Approximations

As far as scalability is concerned, sparse GP methods have been developed that approximate the multivariate Gaussian probability density function (pdf) over training data with the marginal over a smaller set of inducing points multiplied by an approximate conditional pdf (Smola and Bartlett (2001); Lawrence et al. (2003); Seeger (2003b,a); Snelson and Ghahramani (2006)). This approximation yields a time complexity linear—rather than cubic—in the data size and squared in the number of inducing points. We refer to Quinonero-Candela and Rasmussen (2005) for a review of sparse GP approximations. More recently, Hensman et al. (2013, 2015) combined sparse GP methods with Stochastic Variational Inference (Hoffman et al. (2013)) for GP regression and GP classification. However, none of these sparse GP methods addresses the selection of the number of inducing points (and the size of the minibatch in the case of Hensman et al. (2013, 2015)), although this may greatly affect scalability. More importantly, although these methods do not impose strong restrictions on the covariance function of the GP model to approximate, they do not address the need for flexible covariance functions inherent to large scale problems, which are more likely to exhibit intricate and local patterns, and applications considered by the authors typically use the vanilla squared exponential kernel.

Lazaro-Gredilla et al. (2010) proposed approximating stationary kernels with truncated Fourier series in Gaussian process regression. An interpretation of the resulting sparse spectrum Gaussian process model as Bayesian basis function regression with a finite number K of trigonometric basis functions allows making inference in time complexity and memory requirement that are both linear in the size of the training sample. However, this model has two major drawbacks. Firstly, it is prone to over-fitting. In effect, the learning machine will aim at inferring the K major spectral frequencies evidenced in the training data. This will only lead to appropriate prediction out-of-sample when the underlying latent phenomenon can be appropriately characterised by a finite discrete spectral decomposition that is expected to be the same everywhere on the domain. Secondly, this model implicitly postulates that the covariance between the values of the GP at two points does not vanish as the distance between the points becomes arbitrarily large. This imposes *a priori* the

view that the underlying function is highly structured, which might be unrealistic in many real-life non-periodic applications. This approach is generalised by the so-called *random Fourier features* methods (Rahimi and Recht (2007); Le et al. (2013); Yang et al. (2015)). Unfortunately all existing random Fourier features methods give rise to stationary covariance functions, which might not be appropriate for data sets exhibiting local patterns.

The bottleneck of inference in the *standard GP paradigm* remains inverting and computing the determinant of a covariance matrix, normally achieved through the Cholesky decomposition or Singular Value Decomposition. Methods have been developed that speed-up these decompositions through low rank approximations (Williams and Seeger (2001)) or by exploiting specific structures in the covariance function and in the input data (Saatchi (2011); Wilson et al. (2014)), which typically give rise to Kronecker or Toeplitz covariance matrices. While the Kronecker method used by Saatchi (2011) and Wilson et al. (2014) is restricted to inputs that form a Cartesian grid and to separable kernels,[1] low rank approximations such as the Nyström method used by Williams and Seeger (2001) modify the covariance function and hence the functional prior in a non-trivial way. Methods have also been proposed to interpolate the covariance matrix on a uniform or Cartesian grid in order to benefit from some of the computational gains of Toeplitz and Kronecker techniques even when the input space is not structured (Wilson and Nickisch (2015)). However, none of these solutions is general as they require that either the covariance function be separable (Kronecker techniques), or the covariance function be stationary and the input space be one-dimensional (Toeplitz techniques).

2.2 Scalability Through Data Distribution

A family of methods have been proposed to scale-up inference in GP models that are based on the observation that it is more computationally efficient to compute the pdf of K independent small Gaussian vectors with size n than to compute the pdf of a single bigger Gaussian vector of size nK. For instance, Kim et al. (2005) and Gramacy and Lee (2008) partitioned the input space, and put independent stationary GP priors on the restrictions of the latent function to the subdomains forming the partition, which can be regarded as independent *local GP experts*. Kim et al. (2005) partitioned the domain using Voronoi tessellations, while Gramacy and Lee (2008) used tree based partitioning. These two approaches are provably equivalent to postulating a (nonstationary) GP prior on the whole domain that is discontinuous along the boundaries of the partition, which might not be desirable if the latent function we would like to infer is continuous, and might affect predictive accuracy. The more local experts there are, the more scalable the model will be, but the more discontinuities the latent function will have, and subsequently the less accurate the approach will be.

Mixtures of Gaussian process experts models (MoE) (Tresp (2001); Rasmussen and Ghahramani (2001); Meeds and Osindero (2006); Ross and Dy (2013)) provide another implementation of this idea. MoE models assume that there are multiple latent functions to be inferred from the data, on which it is placed independent GP priors, and each training input is associated to one latent function. The number of latent functions and the repartition of data between latent functions can then be performed in a full Bayesian nonparametric fashion (Rasmussen and Ghahramani (2001); Ross and Dy (2013)). When there is a single continuous latent function to be inferred, as it is the case for most regression models, the foregoing Bayesian nonparametric approach will learn a single latent function, thereby leading to a time complexity and a memory requirement that are the same as in the *standard GP paradigm*, which defies the scalability argument.

1. That is multivariate kernel that can be written as product of univariate kernels.

The last implementation of the idea in this section consists of distributing the training data over multiple independent but identical GP models. In regression problems, examples include the *Bayesian Committee Machines* (BCM) of Tresp (2000), the *generalized product of experts* (gPoE) model of Cao and Fleet (2014), and the *robust Bayesian Committee Machines* (rBCM) of Deisenroth and Ng (2015). These models propose splitting the training data in small subsets, each subset being assigned to a different GP regression model—referred to as an expert—that has the same hyper-parameters as the other experts, although experts are assumed to be mutually independent. Training is performed by maximum marginal likelihood, with time complexity (resp. memory requirement) linear in the number of experts and cubic (resp. squared) in the size of the largest data set processed by an expert. Predictions are then obtained by aggregating the predictions of all GP experts in a manner that is specific to the method used (that is the BCM, the gPoE or the rBCM). However, these methods present major drawbacks in the training and testing procedures. In effect, the assumption that experts have identical hyper-parameters is inappropriate for data sets exhibiting local patterns. Even if one would allow GP experts to be driven by different hyper-parameters as in Nguyen and Bonilla (2014) for instance, learned hyper-parameters would lead to overly simplistic GP experts and poor aggregated predictions when the number of training inputs assigned to each expert is small—this is a direct consequence of the (desirable) fact that maximum marginal likelihood GP regression abides by Occam's razor. Another critical pitfall of BCM, gPoE and rBCM is that their methods for aggregating expert predictions are Kolmogorov *inconsistent*. For instance, denoting $\hat{p}$ the predictive distribution in the BCM, it can be easily seen from Equations (2.4) and (2.5) in Tresp (2000) that the predictive distribution $\hat{p}(f(x_1^*)|\mathcal{D})$ (resp. $\hat{p}(f(x_2^*)|\mathcal{D})$)[2] provided by the aggregation procedure of the BCM is *not* the marginal over $f(x_2^*)$ (resp. over $f(x_1^*)$) of the multivariate predictive distribution $\hat{p}(f(x_1^*), f(x_2^*)|\mathcal{D})$ obtained from experts multivariate predictions $p_k(f(x_1^*), f(x_2^*)|\mathcal{D})$ using the same aggregation procedure: $\hat{p}(f(x_1^*)|\mathcal{D}) \neq \int \hat{p}(f(x_1^*), f(x_2^*)|\mathcal{D}) df(x_2^*)$. Without Kolmogorov consistency, it is impossible to make principled Bayesian inference of latent function values. A principled Bayesian nonparametric model should not provide predictions about $f(x_1^*)$ that differ depending on whether or not one is also interested in predicting other values $f(x_i^*)$ simultaneously. This pitfall might be the reason why Cao and Fleet (2014) and Deisenroth and Ng (2015) restricted their expositions to predictive distributions about a single function value at a time $\hat{p}(f(x^*)|\mathcal{D})$, although their procedures (Equation 4 in Cao and Fleet (2014) and Equation 20 in Deisenroth and Ng (2015)) are easily extended to posterior distributions over multiple function values. These extensions would also be Kolmogorov *inconsistent*, and restricting the predictions to be of exactly one function value is unsatisfactory as it does not allow determining the posterior covariance between function values at two test inputs.

2.3 Expressive Stationary Kernels

In regards to flexibly handling complex patterns likely to occur in large data sets, Wilson and Adams (2013) introduced a class of expressive stationary kernels obtained by summing up convolutions of Gaussian basis functions with Dirac delta functions in the spectral domain. The sparse spectrum kernel can be thought of as the special case where the convolving Gaussian is degenerate. Although such kernels perform particularly well in the presence of globally repeated patterns in the data, their stationarity limits their utility on data sets with local patterns. Moreover the proposed covariance

2. Here f is the latent function to be inferred, x_1^*, x_2^* are test points and $\mathcal{D}$ denotes training data.

functions generate infinitely differentiable random functions, which might be too restrictive in some applications.

2.4 Application-Specific Nonstationary Kernels

As for nonstationary kernels, Paciorek and Schervish (2004) proposed a method for constructing nonstationary covariance functions from any stationary one that involves introducing n input dependent $d \times d$ covariance matrices that will be inferred from the data. Plagemann et al. (2008) proposed a faster approximation to the model of Paciorek and Schervish (2004). However, both approaches scale poorly with the input dimension and the data size as they have time complexity $\mathcal{O}\left(\max(nd^3, n^3)\right)$. MacKay (1998), Schmidt and O'Hagan (2003), and Calandra et al. (2014) proposed kernels that can be regarded as stationary after a non-linear transformation d on the input space: $k(x, x') = h\left(\|d(x) - d(x')\|\right)$, where h is positive semi-definite. Although for a given deterministic function d the kernel k is nonstationary, Schmidt and O'Hagan (2003) put a GP prior on d with mean function $m(x) = x$ and covariance function invariant under translation, which unfortunately leads to a kernel that is (unconditionally) stationary, albeit more flexible than $h\left(\|x - x'\|\right)$. To model nonstationarity, Adams and Stegle (2008) introduced a functional prior of the form $y(x) = f(x) \exp g(x)$ where f is a stationary GP and g is some scaling function on the domain. For a given non-constant function g such a prior indeed yields a nonstationary Gaussian process. However, when a stationary GP prior is placed on the function g as Adams and Stegle (2008) did, the resulting functional prior $y(x) = f(x) \exp g(x)$ becomes stationary. The piecewise GP (Kim et al. (2005)) and treed GP (Gramacy and Lee (2008)) models previously discussed also introduce nonstationarity. The authors' premise is that heterogeneous patterns might be locally homogeneous. However, as previously discussed such models are inappropriate for modelling continuous latent functions.

2.5 Our Approach

The approach we propose in this paper for inferring latent functions in large scale problems, possibly exhibiting locally homogeneous patterns, consists of constructing a novel class of *smooth*, *nonstationary* and *flexible* stochastic processes we refer to as *string Gaussian processes* (*string GPs*), whose finite dimensional marginals are structured enough so that full Bayesian nonparametric inference scales linearly with the sample size, without resorting to approximations. Our approach is analogous to MoE models in that, when the input space is one-dimensional, a *string GP* can be regarded as a *collaboration of local GP experts* on non-overlapping supports, that implicitly exchange messages with one another, and that are independent conditional on the aforementioned messages. Each local GP expert only shares just enough information with adjacent local GP experts for the whole stochastic process to be sufficiently smooth (for instance continuously differentiable), which is an important improvement over MoE models as the latter generate discontinuous latent functions. These messages will take the form of boundary conditions, conditional on which each local GP expert will be independent from any other local GP expert. Crucially, unlike the BCM, the gPoE and the rBCM, we do not assume that local GP experts share the same prior structure (that is mean function, covariance function, or hyper-parameters). This allows each local GP expert to flexibly learn local patterns from the data if there are any, while preserving global smoothness, which will result in improved accuracy. Similarly to MoEs, the computational gain in our approach stems from the fact that the conditional independence of the local GP experts conditional on shared boundary

conditions will enable us to write the joint distribution over function and derivative values at a large number of inputs as the product of pdfs of much smaller Gaussian vectors. The resulting effect on time complexity is a decrease from $\mathcal{O}(N^3)$ to $\mathcal{O}(\max_k n_k^3)$, where $N = \sum_k n_k,\ n_k \ll N$. In fact, in Section 5 we will propose Reversible-Jump Monte Carlo Markov Chain (RJ-MCMC) inference methods that achieve memory requirement and time complexity $\mathcal{O}(N)$, without any loss of flexibility. All these results are preserved by our extension of *string GPs* to multivariate input spaces, which we will occasionally refer to as *membrane Gaussian processes* (or membrane GPs). Unlike the BCM, the gPoE and the rBCM, the approach we propose in this paper, which we will refer to as the *string GP paradigm*, is Kolmogorov consistent, and enables principled inference of the posterior distribution over the values of the latent function at multiple test inputs.

3. Construction of String and Membrane Gaussian Processes

In this section we formally construct *string* Gaussian processes, and we provide some important theoretical results including smoothness, and the joint law of *string GPs* and their gradients. We construct *string GPs* indexed on $\mathbb{R}$, before generalising to *string GPs* indexed on $\mathbb{R}^d$, which we will occasionally refer to as *membrane GPs* to stress that the input space is multivariate. We start by considering the joint law of a differentiable GP on an interval and its derivative, and introducing some related notions that we will use in the construction of *string GPs*.

Proposition 1 ***(Derivative Gaussian processes)***
Let I be an interval, $k : I \times I \to \mathbb{R}$ a $\mathcal{C}^2$ symmetric positive semi-definite function,[3] *$m : I \to \mathbb{R}$ a $\mathcal{C}^1$ function.*

(A) There exists a $\mathbb{R}^2$-valued stochastic process $(D_t)_{t\in I}$, $D_t = (z_t, z'_t)$, such that for all $t_1, \dots, t_n \in I$, $(z_{t_1}, \dots, z_{t_n}, z'_{t_1}, \dots, z'_{t_n})$ is a Gaussian vector with mean $\left(m(t_1), \dots, m(t_n), \frac{dm}{dt}(t_1), \dots, \frac{dm}{dt}(t_n)\right)$ and covariance matrix such that

$$cov(z_{t_i}, z_{t_j}) = k(t_i, t_j), \quad cov(z_{t_i}, z'_{t_j}) = \frac{\partial k}{\partial y}(t_i, t_j), \quad \text{and} \quad cov(z'_{t_i}, z'_{t_j}) = \frac{\partial^2 k}{\partial x \partial y}(t_i, t_j),$$

where $\frac{\partial}{\partial x}$ (resp. $\frac{\partial}{\partial y}$) refers to the partial derivative with respect to the first (resp. second) variable of k. We herein refer to $(D_t)_{t\in I}$ as a ***derivative Gaussian process****.*

(B) $(z_t)_{t\in I}$ is a Gaussian process with mean function m, covariance function k and that is $\mathcal{C}^1$ in the L^2 (mean square) sense.

(C) $(z'_t)_{t\in I}$ is a Gaussian process with mean function $\frac{dm}{dt}$ and covariance function $\frac{\partial^2 k}{\partial x \partial y}$. Moreover, $(z'_t)_{t\in I}$ is the L^2 derivative of the process $(z_t)_{t\in I}$.

Proof Although this result is known in the Gaussian process community, we provide a proof for the curious reader in Appendix B. ■

We will say of a kernel k that it is **degenerate at** a when a *derivative Gaussian process* $(z_t, z'_t)_{t\in I}$

3. $\mathcal{C}^1$ (resp. $\mathcal{C}^2$) functions denote functions that are once (resp. twice) continuously differentiable on their domains.

with kernel k is such that z_a and z'_a are perfectly correlated,[4] that is

$$|\text{corr}(z_a, z'_a)| = 1.$$

As an example, the linear kernel $k(u,v) = \sigma^2(u-c)(v-c)$ is degenerate at 0. Moreover, we will say of a kernel k that it is **degenerate at** b **given** a when it is not degenerate at a and when the *derivative Gaussian process* $(z_t, z'_t)_{t\in I}$ with kernel k is such that the variances of z_b and z'_b conditional on (z_a, z'_a) are both zero.[5] For instance, the periodic kernel proposed by MacKay (1998) with period T is degenerate at $u+T$ given u.

An important subclass of *derivative Gaussian processes* in our construction are the processes resulting from conditioning paths of a *derivative Gaussian process* to take specific values at certain times $(t_1, \ldots, t_c)$. We herein refer to those processes as ***conditional derivative Gaussian process***. As an illustration, when k is $\mathcal{C}^3$ on $I \times I$ with $I = [a,b]$, and neither degenerate at a nor degenerate at b given a, the *conditional derivative Gaussian process* on $I = [a,b]$ with unconditional mean function m and unconditional covariance function k that is conditioned to start at $(\tilde{z}_a, \tilde{z}'_a)$ is the *derivative Gaussian process* with mean function

$$\forall\, t \in I, \quad m_c^a(t; \tilde{z}_a, \tilde{z}'_a) = m(t) + \tilde{\mathbf{K}}_{t;a}\mathbf{K}_{a;a}^{-1}\begin{bmatrix} \tilde{z}_a - m(a) \\ \tilde{z}'_a - \frac{dm}{dt}(a) \end{bmatrix}, \tag{1}$$

and covariance function k_c^a that reads

$$\forall\, t, s \in I, \quad k_c^a(t,s) = k(t,s) - \tilde{\mathbf{K}}_{t;a}\mathbf{K}_{a;a}^{-1}\tilde{\mathbf{K}}_{s;a}^T \tag{2}$$

where $\mathbf{K}_{u;v} = \begin{bmatrix} k(u,v) & \frac{\partial k}{\partial y}(u,v) \\ \frac{\partial k}{\partial x}(u,v) & \frac{\partial^2 k}{\partial x \partial y}(u,v) \end{bmatrix}$, and $\tilde{\mathbf{K}}_{t;a} = \begin{bmatrix} k(t,a) & \frac{\partial k}{\partial y}(t,a) \end{bmatrix}$. Similarly, when the process is conditioned to start at $(\tilde{z}_a, \tilde{z}'_a)$ and to end at $(\tilde{z}_b, \tilde{z}'_b)$, the mean function reads

$$\forall\, t \in I, \quad m_c^{a,b}(t; \tilde{z}_a, \tilde{z}'_a, \tilde{z}_b, \tilde{z}'_b) = m(t) + \tilde{\mathbf{K}}_{t;(a,b)}\mathbf{K}_{(a,b);(a,b)}^{-1}\begin{bmatrix} \tilde{z}_a - m(a) \\ \tilde{z}'_a - \frac{dm}{dt}(a) \\ \tilde{z}_b - m(b) \\ \tilde{z}'_b - \frac{dm}{dt}(b) \end{bmatrix}, \tag{3}$$

and the covariance function $k_c^{a,b}$ reads

$$\forall\, t, s \in I, \quad k_c^{a,b}(t,s) = k(t,s) - \tilde{\mathbf{K}}_{t;(a,b)}\mathbf{K}_{(a,b);(a,b)}^{-1}\tilde{\mathbf{K}}_{s;(a,b)}^T, \tag{4}$$

where $\mathbf{K}_{(a,b);(a,b)} = \begin{bmatrix} \mathbf{K}_{a;a} & \mathbf{K}_{a;b} \\ \mathbf{K}_{b;a} & \mathbf{K}_{b;b} \end{bmatrix}$, and $\tilde{\mathbf{K}}_{t;(a,b)} = \begin{bmatrix} \tilde{\mathbf{K}}_{t;a} & \tilde{\mathbf{K}}_{t;b} \end{bmatrix}$. It is important to note that both $\mathbf{K}_{a;a}$ and $\mathbf{K}_{(a,b);(a,b)}$ are indeed invertible because the kernel is assumed to be neither degenerate at a nor degenerate at b given a. Hence, the support of (z_a, z'_a, z_b, z'_b) is $\mathbb{R}^4$, and any function and derivative values can be used for conditioning. Figure 1 illustrates example independent draws from a *conditional derivative Gaussian process*.

4. Or equivalently when the Gaussian vector (z_a, z'_a) is degenerate.
5. Or equivalently when the Gaussian vector (z_a, z'_a) is not degenerate but (z_a, z'_a, z_b, z'_b) is.

3.1 String Gaussian Processes on $\mathbb{R}$

The intuition behind string Gaussian processes on an interval comes from the analogy of collaborative local GP experts we refer to as *strings* that are connected but independent of each other conditional on some regularity boundary conditions. While each string is tasked with representing local patterns in the data, a string only shares the *states* of its extremities (value and derivative) with adjacent strings. Our aim is to preserve global smoothness and limit the amount of information shared between strings, thus reducing computational complexity. Furthermore, the conditional independence between strings will allow for distributed inference, greater flexibility and principled nonstationarity construction.

The following theorem at the core of our framework establishes that it is possible to connect together GPs on a partition of an interval I, in a manner consistent enough that the newly constructed stochastic object will be a stochastic process on I and in a manner restrictive enough that any two connected GPs will share just enough information to ensure that the constructed stochastic process is continuously differentiable ($\mathcal{C}^1$) on I in the L^2 sense.

Theorem 2 ***(String Gaussian process)***
Let $a_0 < \cdots < a_k < \cdots < a_K$, $I = [a_0, a_K]$ and let $p_{\mathcal{N}}(x; \mu, \Sigma)$ be the multivariate Gaussian density with mean vector μ and covariance matrix Σ. Furthermore, let $(m_k : [a_{k-1}, a_k] \to \mathbb{R})_{k\in[1..K]}$ be $\mathcal{C}^1$ functions, and $(k_k : [a_{k-1}, a_k] \times [a_{k-1}, a_k] \to \mathbb{R})_{k\in[1..K]}$ be $\mathcal{C}^3$ symmetric positive semi-definite functions, neither degenerate at a_{k-1}, nor degenerate at a_k given a_{k-1}.

(A) There exists an $\mathbb{R}^2$-valued stochastic process $(SD_t)_{t\in I}$, $SD_t = (z_t, z'_t)$ satisfying the following conditions:
1) The probability density of $(SD_{a_0}, \ldots, SD_{a_K})$ reads:

$$p_b(x_0, \ldots, x_K) := \prod_{k=0}^{K} p_{\mathcal{N}}\left(x_k; \mu_k^b, \Sigma_k^b\right) \tag{5}$$

$$\textit{where:} \quad \Sigma_0^b = {}_1\boldsymbol{K}_{a_0;a_0}, \ \ \forall\, k > 0 \quad \Sigma_k^b = {}_k\boldsymbol{K}_{a_k;a_k} - {}_k\boldsymbol{K}_{a_k;a_{k-1}}\, {}_k\boldsymbol{K}^{-1}_{a_{k-1};a_{k-1}}\, {}_k\boldsymbol{K}^{T}_{a_k;a_{k-1}}, \tag{6}$$

$$\mu_0^b = {}_1\boldsymbol{M}_{a_0}, \ \ \forall\, k > 0 \quad \mu_k^b = {}_k\boldsymbol{M}_{a_k} + {}_k\boldsymbol{K}_{a_k;a_{k-1}}\, {}_k\boldsymbol{K}^{-1}_{a_{k-1};a_{k-1}}(x_{k-1} - {}_k\boldsymbol{M}_{a_{k-1}}), \tag{7}$$

$$\textit{with} \quad {}_k\boldsymbol{K}_{u;v} = \begin{bmatrix} k_k(u,v) & \frac{\partial k_k}{\partial y}(u,v) \\ \frac{\partial k_k}{\partial x}(u,v) & \frac{\partial^2 k_k}{\partial x \partial y}(u,v) \end{bmatrix}, \quad \textit{and} \quad {}_k\boldsymbol{M}_u = \begin{bmatrix} m_k(u) \\ \frac{dm_k}{dt}(u) \end{bmatrix}.$$

*2) Conditional on $(SD_{a_k} = x_k)_{k\in[0..K]}$, the restrictions $(SD_t)_{t\in]a_{k-1},a_k[}$, $k \in [1..K]$ are **independent conditional derivative Gaussian processes**, respectively with unconditional mean function m_k and unconditional covariance function k_k and that are conditioned to take values x_{k-1} and x_k at a_{k-1} and a_k respectively. We refer to $(SD_t)_{t\in I}$ as a **string derivative Gaussian process**, and to its first coordinate $(z_t)_{t\in I}$ as a **string Gaussian process** namely,*

$$(z_t)_{t\in I} \sim \mathcal{SGP}(\{a_k\}, \{m_k\}, \{k_k\}).$$

*(B) The **string Gaussian process** $(z_t)_{t\in I}$ defined in (A) is $\mathcal{C}^1$ in the L^2 sense and its L^2 derivative is the process $(z'_t)_{t\in I}$ defined in (A).*

Proof See Appendix C. ■

In our *collaborative local GP experts* analogy, Theorem 2 stipulates that each local expert takes as message from the previous expert its left hand side boundary conditions, conditional on which it generates its right hand side boundary conditions, which it then passes on to the next expert. Conditional on their boundary conditions local experts are independent of each other, and resemble vibrating pieces of string on fixed extremities, hence the name *string Gaussian process*.

3.2 Pathwise Regularity

Thus far we have dealt with regularity only in the L^2 sense. However, we note that a sufficient condition for the process $(z'_t)_{t\in I}$ in Theorem 2 to be almost surely continuous (i.e. sample paths are continuous with probability 1) and to be the almost sure derivative of the string Gaussian process $(z_t)_{t\in I}$, is that the Gaussian processes on $I_k = [a_{k-1}, a_k]$ with mean and covariance functions $m_{ck}^{a_{k-1},a_k}$ and $k_{ck}^{a_{k-1},a_k}$ (as per Equations 3 and 4 with $m := m_k$ and $k := k_k$) are themselves almost surely $\mathcal{C}^1$ for every boundary condition.[6] We refer to (Adler and Taylor, 2011, Theorem 2.5.2) for a sufficient condition under which a $\mathcal{C}^1$ in L^2 Gaussian process is also almost surely $\mathcal{C}^1$. As the above question is provably equivalent to that of the almost sure continuity of a Gaussian process (see Adler and Taylor, 2011, p. 30), *Kolmogorov's continuity theorem* (see Øksendal, 2003, Theorem 2.2.3) provides a more intuitive, albeit stronger, sufficient condition than that of (Adler and Taylor, 2011, Theorem 2.5.2).

3.3 Illustration

Algorithm 1 illustrates sampling jointly from a string Gaussian process and its derivative on an interval $I = [a_0, a_K]$. We start off by sampling the string boundary conditions (z_{a_k}, z'_{a_k}) sequentially, conditional on which we sample the values of the stochastic process on each string. This we may do in parallel as the strings are independent of each other conditional on boundary conditions. The resulting time complexity is the sum of $\mathcal{O}(\max\ n_k^3)$ for sampling values within strings, and $\mathcal{O}(n)$ for sampling boundary conditions, where the sample size is $n = \sum_k n_k$. The memory requirement grows as the sum of $\mathcal{O}(\sum_k n_k^2)$, required to store conditional covariance matrices of the values within strings, and $\mathcal{O}(K)$ corresponding to the storage of covariance matrices of boundary conditions. In the special case where strings are all empty, that is inputs and boundary times are the same, the resulting time complexity and memory requirement are $\mathcal{O}(n)$. Figure 2 illustrates a sample from a string Gaussian process, drawn using this approach.

3.4 String Gaussian Processes on $\mathbb{R}^d$

So far the input space has been assumed to be an interval. We generalise *string GPs* to hyper-rectangles in $\mathbb{R}^d$ as stochastic processes of the form:

$$f(t_1, \ldots, t_d) = \phi\left(z_{t_1}^1, \ldots, z_{t_d}^d\right), \tag{10}$$

where the *link function* $\phi : \mathbb{R}^d \to \mathbb{R}$ is a $\mathcal{C}^1$ function and (z_t^j) are d independent ($\perp$) latent string Gaussian processes on intervals. We will occasionally refer to *string GPs* indexed on $\mathbb{R}^d$ with $d > 1$ as *membrane GPs* to avoid any ambiguity. We note that when $d = 1$ and when the link function

6. The proof is provided in Appendix D.

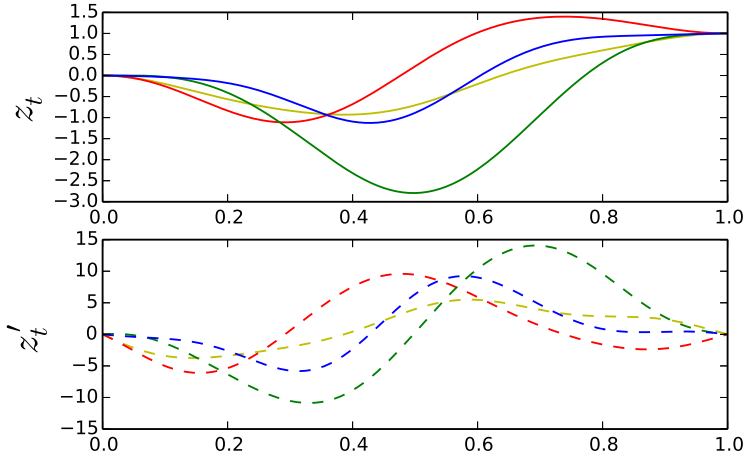

Figure 1: Draws from a conditional derivative GP conditioned to start at 0 with derivative 0 and to finish at 1.0 with derivative 0.0. The unconditional kernel is the squared exponential kernel with variance 1.0 and input scale 0.2.

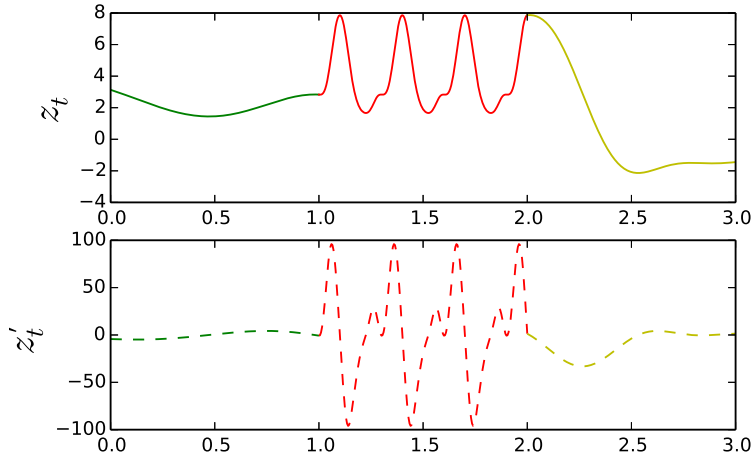

Figure 2: Draw from a *string GP* (z_t) with 3 strings and its derivative (z'_t), under squared exponential kernels (green and yellow strings), and the periodic kernel of MacKay (1998) (red string).

Algorithm 1 Simulation of a string derivative Gaussian process

Inputs: boundary times $a_0 < \cdots < a_K$, string times $\{t_j^k \in]a_{k-1}, a_k[\}_{j\in[1..n_k],k\in[1..K]}$, unconditional mean (resp. covariance) functions m_k (resp. k_k)
Output: $\{\ldots, z_{a_k}, z'_{a_k}, \ldots, z_{t_j^k}, z'_{t_j^k}, \ldots\}$.

Step 1: sample the boundary conditions sequentially.
for $k = 0$ **to** K **do**
Sample $(z_{a_k}, z'_{a_k}) \sim \mathcal{N}\left(\mu_k^b, \Sigma_k^b\right)$, with μ_k^b and Σ_k^b as per Equations (7) and (6).
end for
Step 2: sample the values on each string conditional on the boundary conditions in *parallel*.
parfor $k = 1$ **to** K **do**
Let ${}_k\mathbf{M}_u$ and ${}_k\mathbf{K}_{u;v}$ be as in Theorem 2,

$$ {}_k\Lambda = \begin{bmatrix} {}_k\mathbf{K}_{t_1^k;a_{k-1}} & {}_k\mathbf{K}_{t_1^k;a_k} \\ \cdots & \cdots \\ {}_k\mathbf{K}_{t_{n_k}^k;a_{k-1}} & {}_k\mathbf{K}_{t_{n_k}^k;a_k} \end{bmatrix} \begin{bmatrix} {}_k\mathbf{K}_{a_{k-1};a_{k-1}} & {}_k\mathbf{K}_{a_{k-1};a_k} \\ {}_k\mathbf{K}_{a_k;a_{k-1}} & {}_k\mathbf{K}_{a_k;a_k} \end{bmatrix}^{-1}, $$

$$ \mu_k^s = \begin{bmatrix} {}_k\mathbf{M}_{t_1^k} \\ \cdots \\ {}_k\mathbf{M}_{t_{n_k}^k} \end{bmatrix} + {}_k\Lambda \begin{bmatrix} z_{a_{k-1}} - m_k(a_{k-1}) \\ z'_{a_{k-1}} - \frac{dm_k}{dt}(a_{k-1}) \\ z_{a_k} - m_k(a_k) \\ z'_{a_k} - \frac{dm_k}{dt}(a_k) \end{bmatrix}, \tag{8} $$

$$ \Sigma_k^s = \begin{bmatrix} {}_k\mathbf{K}_{t_1^k;t_1^k} & \cdots & {}_k\mathbf{K}_{t_1^k;t_{n_k}^k} \\ \cdots & \cdots & \cdots \\ {}_k\mathbf{K}_{t_{n_k}^k;t_1^k} & \cdots & {}_k\mathbf{K}_{t_{n_k}^k;t_{n_k}^k} \end{bmatrix} - {}_k\Lambda \begin{bmatrix} {}_k\mathbf{K}_{t_1^k;a_{k-1}} & {}_k\mathbf{K}_{t_1^k;a_k} \\ \cdots & \cdots \\ {}_k\mathbf{K}_{t_{n_k}^k;a_{k-1}} & {}_k\mathbf{K}_{t_{n_k}^k;a_k} \end{bmatrix}^T. \tag{9} $$

Sample $\left(z_{t_1^k}, z'_{t_1^k} \ldots, z_{t_{n_k}^k}, z'_{t_{n_k}^k}\right) \sim \mathcal{N}\left(\mu_k^s, \Sigma_k^s\right)$.
end parfor

is $\phi(x) = x$, we recover *string GPs* indexed on an interval as previously defined. When the *string GPs* (z_t^j) are a.s. $\mathcal{C}^1$, the *membrane GP* f in Equation (10) is also a.s. $\mathcal{C}^1$, and the partial derivative with respect to the j-th coordinate reads:

$$\frac{\partial f}{\partial t_j}(t_1, \ldots, t_d) = z_{t_j}^{j\prime} \frac{\partial \phi}{\partial t_j}\left(z_{t_1}^1, \ldots, z_{t_d}^d\right). \tag{11}$$

Thus in high dimensions, *string GPs* easily allow an explanation of the sensitivity of the learned latent function to inputs.

3.5 Choice of Link Function

Our extension of *string GPs* to $\mathbb{R}^d$ departs from the *standard GP paradigm* in that we did not postulate a covariance function on $\mathbb{R}^d \times \mathbb{R}^d$ directly. Doing so usually requires using a metric on $\mathbb{R}^d$, which is often problematic for heterogeneous input dimensions, as it introduces an arbitrary comparison between distances in each input dimension. This problem has been partially addressed by approaches such as Automatic Relevance Determination (ARD) kernels, that allow for a linear rescaling of input dimensions to be learned jointly with kernel hyper-parameters. However, inference under a *string GP* prior can be thought of as learning a coordinate system in which the latent function f resembles the link function ϕ through non-linear rescaling of input dimensions. In particular, when ϕ is symmetric, the learned univariate *string GPs* (being interchangeable in ϕ) implicitly aim at normalizing input data across dimensions, making *string GPs* naturally cope with heterogeneous data sets.

An important question arising from our extension is whether or not the link function ϕ needs to be learned to achieve a flexible functional prior. The flexibility of a *string GP* as a functional prior depends on both the *link function* and the covariance structures of the underlying *string GP* building blocks (z_t^j). To address the impact of the choice of ϕ on flexibility, we constrain the *string GP* building blocks by restricting them to be independent identically distributed *string GPs* with one string each (i.e. (z_t^j) are i.i.d Gaussian processes). Furthermore, we restrict ourselves to isotropic kernels as they provide a consistent basis for putting the same covariance structure in $\mathbb{R}$ and $\mathbb{R}^d$. One question we might then ask, for a given *link function* ϕ_0, is whether or not an isotropic GP indexed on $\mathbb{R}^d$ with covariance function k yields more flexible random surfaces than the stationary *string GP* $f(t_1, \ldots, t_d) = \phi_0(z_{t_1}^1, \ldots, z_{t_d}^d)$, where $(z_{t_j}^j)$ are stationary GPs indexed on $\mathbb{R}$ with the same covariance function k. If we find a *link function* ϕ_0 generating more flexible random surfaces than isotropic GP counterparts it would suggest ϕ need not to be inferred in dimension $d > 1$ to be more flexible than any GP using one of the large number of commonly used isotropic kernels, among which squared exponential kernels, rational quadratic kernels, and Matérn kernels to name but a few.

Before discussing whether such a ϕ_0 exists, we need to introduce a rigorous meaning to 'flexibility'. An intuitive qualitative definition of the flexibility of a stochastic process indexed on $\mathbb{R}^d$ is the ease with which it can generate surfaces with varying shapes from one random sample to another independent one. We recall that the tangent hyperplane to a $\mathcal{C}^1$ surface $y - f(x) = 0, x \in \mathbb{R}^d$ at some point $x_0 = (t_1^0, \ldots, t_d^0)$ has equation $\nabla f(x_0)^T(x - x_0) - (y - f(x_0)) = 0$ and admits as normal vector $(\frac{\partial f}{\partial t_1}(t_1^0), \ldots, \frac{\partial f}{\partial t_d}(t_d^0), -1)$. As tangent hyperplanes approximate a surface locally, a first criterion of flexibility for a random surface $y - f(x) = 0,\ x \in \mathbb{R}^d$ is the proclivity of the (random) direction of its tangent hyperplane at any point x—and hence the proclivity of $\nabla f(x)$—to vary.

This criterion alone, however, does not capture the difference between the local shapes of the random surface at two distinct points. A complementary second criterion of flexibility is the proclivity of the (random) directions of the tangent hyperplanes at any two distinct points $x_0, x_1 \in \mathbb{R}^d$—and hence the proclivity of $\nabla f(x_0)$ and $\nabla f(x_1)$—to be independent. The first criterion can be measured using the entropy of the gradient at a point, while the second criterion can be measured through the mutual information between the two gradients. The more flexible a stochastic process, the higher the entropy of its gradient at any point, and the lower the mutual information between its gradients at any two distinct points. This is formalised in the definition below.

Definition 3 ***(Flexibility of stochastic processes)***
Let f and g be two real valued, almost surely $\mathcal{C}^1$ stochastic processes indexed on $\mathbb{R}^d$, and whose gradients have a finite entropy everywhere (i.e. $\forall\ x,\ H(\nabla f(x)), H(\nabla g(x)) < \infty$). We say that f is more flexible than g if the following conditions are met:
1) $\forall\ x,\ H(\nabla f(x)) \geq H(\nabla g(x))$,
2) $\forall\ x \neq y,\ I(\nabla f(x); \nabla f(y)) \leq I(\nabla g(x); \nabla g(y))$,
where H is the entropy operator, and $I(X;Y) = H(X) + H(Y) - H(X,Y)$ stands for the mutual information between X and Y.

The following proposition establishes that the *link function* $\phi_s\left(x_1, \ldots, x_d\right) = \sum_{i=j}^{d} x_j$ yields more flexible stationary *string GPs* than their isotropic GP counterparts, thereby providing a theoretical underpinning for not inferring ϕ.

Proposition 4 ***(Additively separable* string GPs *are flexible)***
Let $k(x,y) := \rho\left(||x-y||_{L^2}^2\right)$ be a stationary covariance function generating a.s. $\mathcal{C}^1$ GP paths indexed on $\mathbb{R}^d$, $d > 0$, and ρ a function that is $\mathcal{C}^2$ on $]0, +\infty[$ and continuous at 0. Let $\phi_s(x_1, \ldots, x_d) = \sum_{j=1}^{d} x_j$, let $(z_t^j)_{t \in I^j,\ j \in [1..d]}$ be independent stationary Gaussian processes with mean 0 and covariance function k (where the L^2 norm is on $\mathbb{R}$), and let $f(t_1, \ldots, t_d) = \phi_s(z_{t_1}^1, \ldots, z_{t_d}^d)$ be the corresponding stationary string GP. Finally, let g be an isotropic Gaussian process indexed on $I^1 \times \cdots \times I^d$ with mean 0 and covariance function k (where the L^2 norm is on $\mathbb{R}^d$). Then:

1) $\forall\ x \in I^1 \times \cdots \times I^d,\ H(\nabla f(x)) = H(\nabla g(x))$,
2) $\forall\ x \neq y \in I^1 \times \cdots \times I^d,\ I(\nabla f(x); \nabla f(y)) \leq I(\nabla g(x); \nabla g(y))$.

Proof See Appendix E. ■

Although the link function need not be inferred in a full nonparametric fashion to yield comparable if not better results than most isotropic kernels used in the *standard GP paradigm*, for some problems certain link functions might outperform others. In Section 4.2 we analyse a broad family of link functions, and argue that they extend successful anisotropic approaches such as the Automatic Relevance Determination (MacKay (1998)) and the additive kernels of Duvenaud et al. (2011). Moreover, in Section 5 we propose a scalable inference scheme applicable to any link function.

4. Comparison with the Standard GP Paradigm

We have already established that sampling *string GPs* scales better than sampling GPs under the *standard GP paradigm* and is amenable to distributed computing. We have also established that

stationary additively separable *string GPs* are more flexible than their isotropic counterparts in the *standard GP paradigm*. In this section, we provide further theoretical results relating the *string GP paradigm* to the *standard GP paradigm*. Firstly we establish that *string GPs* with link function $\phi_s(x_1, \ldots, x_d) = \sum_{i=j}^{d} x_j$ are GPs. Secondly, we derive the global mean and covariance functions induced by the *string GP* construction for a variety of link functions. Thirdly, we provide a sense in which the *string GP paradigm* can be thought of as extending the *standard GP paradigm*. And finally, we show that the *string GP paradigm* may serve as a scalable approximation of commonly used stationary kernels.

4.1 Some String GPs are GPs

On one hand we note from Theorem 2 that the restriction of a *string GP* defined on an interval to the support of the first string—in other words the first local GP expert—is a Gaussian process. On the other hand, the messages passed on from one local GP expert to the next are not necessarily consistent with the unconditional law of the receiving local expert, so that overall a *string GP* defined on an interval, that is when looked at globally and unconditionally, might not be a Gaussian process. However, the following proposition establishes that some *string GPs* are indeed Gaussian processes.

Proposition 5 ***(Additively separable* string GPs *are GPs)***
String Gaussian processes on $\mathbb{R}$ are Gaussian processes. Moreover, string Gaussian processes on $\mathbb{R}^d$ with link function $\phi_s(x_1, \ldots, x_d) = \sum_{j=1}^{d} x_j$ are also Gaussian processes.

Proof The intuition behind this proof lies in the fact that if X is a multivariate Gaussian, and if conditional on X, Y is a multivariate Gaussian, providing that the conditional mean of Y depends linearly on X and the conditional covariance matrix of Y does not depend on X, the vector (X, Y) is jointly Gaussian. This will indeed be the case for our collaboration of local GP experts as the boundary conditions picked up by an expert from the previous will not influence the conditional covariance structure of the expert (the conditional covariance strucuture depends only on the partition of the domain, not the values of the boundary conditions) and will affect the mean linearly. See Appendix H for the full proof. ■

The above result guarantees that commonly used closed form predictive equations under GP priors are still applicable under some *string GP* priors, providing the global mean and covariance functions, which we derive in the following section, are available. Proposition 5 also guarantees stability of the corresponding *string GPs* in the GP family under addition of independent Gaussian noise terms as in regression settings. Moreover, it follows from Proposition 5 that inference techniques developed for Gaussian processes can be readily used under *string GP* priors. In Section 5 we provide an additional MCMC scheme that exploits the conditional independence between strings to yield greater scalability and distributed inference.

4.2 String GP Kernels and String GP Mean Functions

The approach we have adopted in the construction of *string GPs* and *membrane GPs* did not require explicitly postulating a global mean function or covariance function. In Appendix I we derive the global mean and covariance functions that result from our construction. The global covariance function could be used for instance as a stand-alone kernel in any kernel method, for instance GP

models under the *standard GP paradigm*, which would provide a flexible and nonstationary alternative to commonly used kernels that may be used to learn local patterns in data sets—some successful example applications are provided in Section 5. That being said, adopting such a global approach should be limited to small scale problems as the conditional independence structure of *string GPs* does not easily translate into structures in covariance matrices over *string GP* values (without derivative information) that can be exploited to speed-up SVD or Cholesky decomposition. Crucially, marginalising out all derivative information in the distribution of *derivative string GP* values at some inputs would destroy any conditional independence structure, thereby limiting opportunities for scalable inference. In Section 5 we will provide a RJ-MCMC inference scheme that fully exploits the conditional independence structure in *string GPs* and scales to *very large* data sets.

4.3 Connection Between Multivariate String GP Kernels and Existing Approaches

We recall that for $n \leq d$, the n-th order *elementary symmetric polynomial* (Macdonald (1995)) is given by

$$e_0(x_1,\ldots,x_d) := 1, \quad \forall 1 \leq n \leq d \;\; e_n(x_1,\ldots,x_d) = \sum_{1\leq j_1<j_2<\cdots<j_n\leq d} \prod_{k=1}^{n} x_{j_k}. \tag{12}$$

As an illustration,

$$e_1(x_1,\ldots,x_d) = \sum_{j=1}^{d} x_j = \phi_s(x_1,\ldots,x_d),$$

$$e_2(x_1,\ldots,x_d) = x_1x_2 + x_1x_3 + \cdots + x_1x_d + \cdots + x_{d-1}x_d,$$

$$\ldots$$

$$e_d(x_1,\ldots,x_d) = \prod_{j=1}^{d} x_j = \phi_p(x_1,\ldots,x_d).$$

Covariance kernels of *string GPs*, using as link functions *elementary symmetric polynomials* e_n, extend most popular approaches that combine unidimensional kernels over features for greater flexibility or cheaper design experiments.

The first-order polynomial e_1 gives rise to additively separable Gaussian processes, that can be regarded as Bayesian nonparametric *generalised additive models* (GAM), particularly popular for their interpretability. Moreover, as noted by Durrande et al. (2012), additively separable Gaussian processes are considerably cheaper than alternate transformations in design experiments with high-dimensional input spaces. In addition to the above, additively separable *string GPs* also allow postulating the existence of local properties in the experimental design process at no extra cost.

The d-th order polynomial e_d corresponds to a product of unidimensional kernels, also known as separable kernels. For instance, the popular squared exponential kernel is separable. Separable kernels have been successfully used on large scale inference problems where the inputs form a grid (Saatchi, 2011; Wilson et al., 2014), as they yield covariance matrices that are Kronecker products, leading to maximum likelihood inference in linear time complexity and with linear memory requirement. Separable kernels are often used in conjunction with the *automatic relevance determination* (ARD) model, to learn the relevance of features through global linear rescaling. However, ARD

kernels might be limited in that we might want the relevance of a feature to depend on its value. As an illustration, the market value of a watch can be expected to be a stronger indicator of its owner's wealth when it is in the top 1 percentile, than when it is in the bottom 1 percentile; the rationale being that possessing a luxurious watch is an indication that one can afford it, whereas possessing a cheap watch might be either an indication of lifestyle or an indication that one cannot afford a more expensive one. Separable *string GP* kernels extend ARD kernels, in that strings between input dimensions and within an input dimension may have unconditional kernels with different hyper-parameters, and possibly different functional forms, thereby allowing for *automatic local relevance determination* (ALRD).

More generally, using as link function the n-th order elementary symmetric polynomial e_n corresponds to the n-th order interaction of the *additive kernels* of Duvenaud et al. (2011). We also note that the class of link functions $\phi(x_1, \ldots, x_d) = \sum_{i=1}^{d} \sigma_i e_i(x_1, \ldots, x_d)$ yield full *additive kernels*. Duvenaud et al. (2011) noted that such kernels are 'exceptionally well-suited' to learn non-local structures in data. *String GPs* complement *additive kernels* by allowing them to learn local structures as well.

4.4 String GPs as Extension of the Standard GP Paradigm

The following proposition provides a perspective from which *string GPs* may be considered as extending Gaussian processes on an interval.

Proposition 6 ***(Extension of the* standard GP paradigm*)***
Let $K \in \mathbb{N}^*$, let $I = [a_0, a_K]$ and $I_k = [a_{k-1}, a_k]$ be intervals with $a_0 < \cdots < a_K$. Furthermore, let $m : I \to \mathbb{R}$ be a $\mathcal{C}^1$ function, m_k the restriction of m to I_k, $h : I \times I \to \mathbb{R}$ a $\mathcal{C}^3$ symmetric positive semi-definite function, and h_k the restriction of h to $I_k \times I_k$. If

$$(z_t)_{t \in I} \sim \mathcal{SGP}(\{a_k\}, \{m_k\}, \{h_k\}),$$

then

$$\forall\, k \in [1..K],\ (z_t)_{t \in I_k} \sim \mathcal{GP}(m, h).$$

Proof See Appendix F. ■

We refer to the case where unconditional string mean and kernel functions are restrictions of the same functions as in Proposition 6 as *uniform string GPs*. Although uniform *string GPs* are not guaranteed to be as much regular at boundary times as their counterparts in the *standard GP paradigm*, we would like to stress that they may well generate paths that are. In other words, the functional space induced by a uniform *string GP* on an interval extends the functional space of the GP with the same mean and covariance functions m and h taken globally and unconditionally on the whole interval as in the *standard GP paradigm*. This allows for (but does not enforce) less regularity at the boundary times. When *string GPs* are used as functional prior, the posterior mean can in fact have more regularity at the boundary times than the continuous differentiability enforced in the *string GP paradigm*, providing such regularity is evidenced in the data.

We note from Proposition 6 that when m is constant and h is stationary, the restriction of the uniform *string GP* $(z_t)_{t \in I}$ to any interval whose interior does not contain a boundary time, the largest of which being the intervals $[a_{k-1}, a_k]$, is a stationary GP. We refer to such cases as *partition stationary string GPs*.

4.5 Commonly Used Covariance Functions and their String GP Counterparts

Considering the superior scalability of the *string GP paradigm*, which we may anticipate from the scalability of sampling *string GPs*, and which we will confirm empirically in Section 5, a natural question that comes to mind is whether or not kernels commonly used in the *standard GP paradigm* can be well approximated by *string GP* kernels, so as to take advantage of the improved scalability of the *string GP paradigm*. We examine the distortions to commonly used covariance structures resulting from restricting strings to share only $\mathcal{C}^1$ boundary conditions, and from increasing the number of strings.

Figure 3 compares some popular stationary kernels on $[0,1] \times [0,1]$ (first column) to their uniform *string GP* kernel counterparts with 2, 4, 8 and 16 strings of equal length. The popular kernels considered are the squared exponential kernel (SE), the rational quadratic kernel $k_{RQ}(u,v) = \left(1 + \frac{2(u-v)^2}{\alpha}\right)^{-\alpha}$ with $\alpha = 1$ (RQ 1) and $\alpha = 5$ (RQ 5), the Matérn 3/2 kernel (MA 3/2), and the Matérn 5/2 kernel (MA 5/2), each with output scale (variance) 1 and input scale 0.5. Firstly, we observe that each of the popular kernels considered coincides with its uniform *string GP* counterparts regardless of the number of strings, so long as the arguments of the covariance function are less than an input scale apart. Except for the Matérn 3/2, the loss of information induced by restricting strings to share only $\mathcal{C}^1$ boundary conditions becomes noticeable when the arguments of the covariance function are more than 1.5 input scales apart, and the effect is amplified as the number of strings increases. As for the Matérn 3/2, no loss of information can been noticed, as further attests Table 1. In fact, this comes as no surprise given that stationary Matérn 3/2 GP are 1-Markov, that is the corresponding derivative Gaussian process is a Markov process so that the vector (z_t, z_t') contains as much information as all *string GP* or derivative values prior to t (see Doob (1944)). Table 1 provides some statistics on the absolute errors between each of the popular kernels considered and uniform *string GP* counterparts.

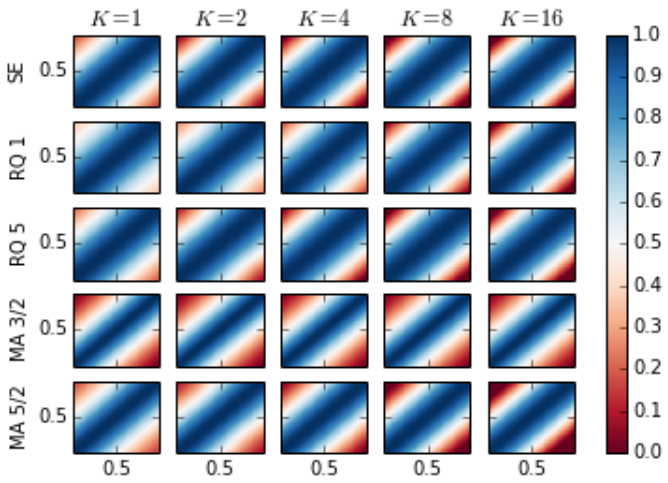

Figure 3: Commonly used covariance functions on $[0, 1] \times [0, 1]$ with the same input and output scales (first column) and their uniform *string GP* counterparts with $K > 1$ strings of equal length.

	$K = 2$			$K = 4$			$K = 8$			$K = 16$		
	min	avg	max	min	avg	max	min	avg	max	min	avg	max
SE	0	0.01	0.13	0	0.02	0.25	0	0.03	0.37	0	0.04	0.44
RQ 1	0	0.01	0.09	0	0.03	0.20	0	0.05	0.37	0	0.07	0.52
RQ 5	0	0.01	0.12	0	0.02	0.24	0	0.04	0.37	0	0.05	0.47
MA 3/2	0	0	0	0	0	0	0	0	0	0	0	0
MA 5/2	0	0.01	0.07	0	0.03	0.15	0	0.05	0.29	0	0.08	0.48

Table 1: Minimum, average, and maximum absolute errors between some commonly used stationary covariance functions on $[0, 1] \times [0, 1]$ (with unit variance and input scale 0.5) and their uniform *string GP* counterparts with $K > 1$ strings of equal length.

5. Inference under String and Membrane GP Priors

In this section we move on to developing inference techniques for Bayesian nonparametric inference of latent functions under *string GP* priors. We begin with marginal likelihood inference in regression problems. We then propose a novel reversible-jump MCMC sampler that enables automatic learning of model complexity (that is the number of different unconditional kernel configurations) from the data, with a time complexity and memory requirement both linear in the number of training inputs.

5.1 Maximum Marginal Likelihood for Small Scale Regression Problems

Firstly, we leverage the fact that additively separable *string GPs* are Gaussian processes to perform Bayesian nonparametric regressions in the presence of local patterns in the data, using standard Gaussian process techniques (see Rasmussen and Williams, 2006, p.112 §5.4.1). We use as generative model

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}\left(0, \sigma^2_{k_i}\right), \quad \sigma^2_{k_i} > 0, \quad x_i \in I^1 \times \cdots \times I^d, \quad y_i, \epsilon_i \in \mathbb{R}$$

we are given the training data set $\mathcal{D} = \{\tilde{x}_i, \tilde{y}_i\}_{i\in[1..N]}$, and we place a mean-zero additively separable *string GP* prior on f, namely

$$f(x) = \sum_{j=1}^{d} z^j_{x[j]}, \quad (z^j_t) \sim \mathcal{SGP}\left(\{a^j_k\}, \{0\}, \{k^j_k\}\right), \quad \forall j < l,\ (z^j_t) \perp (z^l_t),$$

which we assume to be independent of the measurement noise process. Moreover, the noise terms are assumed to be independent, and the noise variance $\sigma^2_{k_i}$ affecting $f(x_i)$ is assumed to be the same for any two inputs whose coordinates lie on the same string intervals. Such a heteroskedastic noise model fits nicely within the *string GP paradigm*, can be very useful when the dimension of the input space is small, and may be replaced by the typical constant noise variance assumption in high-dimensional input spaces.

Let us define $\mathbf{y} = (\tilde{y}_1, \ldots, \tilde{y}_N)$, $\mathbf{X} = (\tilde{x}_1, \ldots, \tilde{x}_N)$, $\mathbf{f} = (f(\tilde{x}_1), \ldots, f(\tilde{x}_N))$ and let $\bar{\mathbf{K}}_{\mathbf{X};\mathbf{X}}$ denote the auto-covariance matrix of $\mathbf{f}$ (which we have derived in Section 4.2), and let $\mathbf{D} = \text{diag}(\{\sigma^2_{k_i}\})$ denote the diagonal matrix of noise variances. It follows that $\mathbf{y}$ is a Gaussian vector with mean $\mathbf{0}$ and auto-covariance matrix $\mathbf{K}_\mathbf{y} := \bar{\mathbf{K}}_{\mathbf{X};\mathbf{X}} + \mathbf{D}$ and that the log marginal likelihood reads:

$$\log p\left(\mathbf{y}\middle|\mathbf{X}, \{\sigma_{k_i}\}, \{\theta^j_k\}, \{a^j_k\}\right) = -\frac{1}{2}\mathbf{y}^T\mathbf{K}_\mathbf{y}^{-1}\mathbf{y} - \frac{1}{2}\log\det\left(\mathbf{K}_\mathbf{y}\right) - \frac{n}{2}\log 2\pi. \tag{13}$$

We obtain estimates of the string measurement noise standard deviations $\{\hat{\sigma}_{k_i}\}$ and estimates of the string hyper-parameters $\{\hat{\theta}^j_k\}$ by maximising the marginal likelihood for a given domain partition $\{a^j_k\}$, using gradient-based methods. We deduce the predictive mean and covariance matrix of the latent function values $\mathbf{f}^*$ at test points $\mathbf{X}^*$, from the estimates $\{\hat{\theta}^j_k\}, \{\hat{\sigma}_{k_i}\}$ as

$$\mathrm{E}(\mathbf{f}^*|\mathbf{y}) = \bar{\mathbf{K}}_{\mathbf{X}^*;\mathbf{X}}\mathbf{K}_\mathbf{y}^{-1}\mathbf{y} \quad \text{and} \quad \mathrm{cov}(\mathbf{f}^*|\mathbf{y}) = \bar{\mathbf{K}}_{\mathbf{X}^*;\mathbf{X}^*} - \bar{\mathbf{K}}_{\mathbf{X}^*;\mathbf{X}}\mathbf{K}_\mathbf{y}^{-1}\bar{\mathbf{K}}_{\mathbf{X};\mathbf{X}^*}, \tag{14}$$

using the fact that $(\mathbf{f}^*, \mathbf{y})$ is jointly Gaussian, and that the cross-covariance matrix between $\mathbf{f}^*$ and $\mathbf{y}$ is $\bar{\mathbf{K}}_{\mathbf{X}^*;\mathbf{X}}$ as the additive measurement noise is assumed to be independent from the latent process f.

5.1.1 REMARKS

The above analysis and equations still hold when a GP prior is placed on f with one of the multivariate *string GP* kernels derived in Section 4.2 as covariance function.

It is also worth noting from the derivation of *string GP* kernels in Appendix I that the marginal likelihood Equation (13) is continuously differentiable in the locations of boundary times. Thus, for a given number of boundary times, the positions of the boundary times can be determined as part of the marginal likelihood maximisation. The derivatives of the marginal log-likelihood (Equation 13) with respect to the aforementioned locations $\{a_k^j\}$ can be determined from the recursions of Appendix I, or approximated numerically by finite differences. The number of boundary times in each input dimension can then be learned by trading off model fit (the maximum marginal log likelihood) and model simplicity (the number of boundary times or model parameters), for instance using information criteria such as AIC and BIC. When the input dimension is large, it might be advantageous to further constrain the hypothesis space of boundary times before using information criteria, for instance by assuming that the number of boundary times is the same in each dimension. An alternative Bayesian nonparametric approach to learning the number of boundary times will be discussed in section 5.4.

This method of inference cannot exploit the structure of *string GPs* to speed-up inference, and as a result it scales like the *standard GP paradigm*. In fact, any attempt to marginalize out univariate derivative processes, including in the prior, will inevitably destroy the conditional independence structure. Another perspective to this observation is found by noting from the derivation of global *string GP* covariance functions in Appendix I that the conditional independence structure does not easily translate in a matrix structure that may be exploited to speed-up matrix inversion, and that marginalizing out terms relating to derivatives processes as in Equation (13) can only make things worse.

5.2 Generic Reversible-Jump MCMC Sampler for Large Scale Inference

More generally, we consider learning a smooth real-valued latent function f, defined on a d-dimensional hyper-rectangle, under a generative model with likelihood $p\left(\mathcal{D}|\mathbf{f},\mathbf{u}\right)$, where $\mathbf{f}$ denotes values of f at training inputs points and $\mathbf{u}$ denotes other likelihood parameters that are not related to f. A large class of machine learning problems aiming at inferring a latent function have a likelihood model of this form. Examples include celebrated applications such as nonparametric regression and nonparametric binary classification problems, but also more recent applications such as learning a profitable portfolio generating-function in *stochastic portfolio theory* (Karatzas and Fernholz (2009)) from the data. In particular, we do not assume that $p\left(\mathcal{D}|\mathbf{f},\mathbf{u}\right)$ factorizes over training inputs. Extensions to likelihood models that depend on the values of multiple latent functions are straight-forward and will be discussed in Section 5.3.

5.2.1 PRIOR SPECIFICATION

We place a prior $p(\mathbf{u})$ on other likelihood parameters. For instance, in regression problems under a Gaussian noise model, $\mathbf{u}$ can be the noise variance and we may choose $p(\mathbf{u})$ to be the inverse-Gamma distribution for conjugacy. We place a mean-zero *string GP* prior on f

$$f(x) = \phi\left(z^1_{x[1]},\ldots,z^d_{x[d]}\right), \quad (z_t^j) \sim \mathcal{SGP}\left(\{a_k^j\},\{0\},\{k_k^j\}\right), \quad \forall j < l,\ (z_t^j) \perp (z_t^l). \tag{15}$$

As discussed in Section 3.5, the link function ϕ need not be inferred as the symmetric sum was found to yield a sufficiently flexible functional prior. Nonetheless, in this section we do not impose any restriction on the link function ϕ other than continuous differentiability. Denoting $\mathbf{z}$ the vector of univariate *string GP* processes and their derivatives, evaluated at all distinct input coordinate values, we may re-parametrize the likelihood as $p\left(\mathcal{D}|\mathbf{z},\mathbf{u}\right)$, with the understanding that $\mathbf{f}$ can be recovered from $\mathbf{z}$ through the link function ϕ. To complete our prior specification, we need to discuss the choice of boundary times $\{a_k^j\}$ and the choice of the corresponding unconditional kernel structures $\{k_k^j\}$. Before doing so, we would like to stress that key requirements of our sampler are that i) it should decouple the need for scalability from the need for flexibility, ii) it should scale linearly with the number of training and test inputs, and iii) the user should be able to express prior views on model complexity/flexibility in an intuitive way, but the sampler should be able to validate or invalidate the prior model complexity from the data. While the motivations for the last two requirements are obvious, the first requirement is motivated by the fact that a massive data set may well be more homogeneous than a much smaller data set.

5.2.2 SCALABLE CHOICE OF BOUNDARY TIMES

To motivate our choice of boundary times that achieves great scalability, we first note that the evaluation of the likelihood, which will naturally be needed by the MCMC sampler, will typically have at least linear time complexity and linear memory requirement, as it will require performing computations that use each training sample at least once. Thus, the best we can hope to achieve overall is linear time complexity and linear memory requirement. Second, in MCMC schemes with functional priors, the time complexity and memory requirements for sampling from the posterior

$$p\left(\mathbf{f}|\mathcal{D}\right) \propto p\left(\mathcal{D}|\mathbf{f}\right) p(\mathbf{f})$$

are often the same as the resource requirements for sampling from the prior $p\left(\mathbf{f}\right)$, as evaluating the model likelihood is rarely the bottleneck. Finally, we note from Algorithm 1 that, when each input coordinate in each dimension is a boundary time, the sampling scheme has time complexity and memory requirement that are linear in the maximum number of unique input coordinates across dimensions, which is at most the number of training samples. In effect, each univariate derivative *string GP* is sampled in *parallel* at as many times as there are unique input coordinates in that dimension, before being combined through the link function. In a given input dimension, univariate *derivative string GP* values are sampled sequentially, one boundary time conditional on the previous. The foregoing sampling operation is very scalable not only asymptotically but also in absolute terms; it merely requires storing and inverting at most as many 2×2 matrices as the number of input points. We will evaluate the actual overall time complexity and memory requirement when we discuss our MCMC sampler in greater details. For now, we would like to stress that i) choosing each distinct input coordinate value as a boundary time in the corresponding input dimension before training is a perfectly valid choice, ii) we expect this choice to result in resource requirements that grow linearly with the sample size and iii) in the *string GP* theory we have developed thus far there is no requirement that two adjacent strings be driven by different kernel hyper-parameters.

5.2.3 MODEL COMPLEXITY LEARNING AS A CHANGE-POINT PROBLEM

The remark iii) above pertains to model complexity. In the simplest case, all strings are driven by the same kernel and hyper-parameters as it was the case in Section 4.5, where we discussed how this

setup departs from postulating the unconditional string covariance function k_k^j globally similarly to the *standard GP paradigm*. The more distinct unconditional covariance structures there are, the more complex the model is, as it may account for more types of local patterns. Thus, we may identify model complexity to the number of different kernel configurations across input dimensions. In order to learn model complexity, we require that some (but not necessarily all) strings share their kernel configuration.[7] Moreover, we require kernel membership to be dimension-specific in that two strings in different input dimensions may not explicitly share a kernel configuration in the prior specification, although the posterior distribution over their hyper-parameters might be similar if the data support it.

In each input dimension j, kernel membership is defined by a partition of the corresponding domain operated by a (possibly empty) set of change-points,[8] as illustrated in Figure 4. When there is no change-point as in Figure 4-(a), all strings are driven by the same kernel and hyper-parameters. Each change-point c_p^j induces a new kernel configuration θ_p^j that is shared by all strings whose boundary times a_k^j and a_{k+1}^j both lie in $[c_p^j, c_{p+1}^j[$. When one or multiple change-points c_p^j occur between two adjacent boundary times as illustrated in Figures 4-(b-d), for instance $a_k^j \leq c_p^j \leq a_{k+1}^j$, the kernel configuration of the string defined on $[a_k^j, a_{k+1}^j]$ is that of the largest change-point that lies in $[a_k^j, a_{k+1}^j]$ (see for instance Figure 4-(d)). For consistency, we denote θ_0^j the kernel configuration driving the first string in the j-th dimension; it also drives strings that come before the first change-point, and all strings when there is no change-point.

To place a prior on model complexity, it suffices to define a joint probability measure on the set of change-points and the corresponding kernel configurations. As kernel configurations are not shared across input dimensions, we choose these priors to be independent across input dimensions. Moreover, $\{c_p^j\}$ being a random collection of points on an interval whose number and positions are both random, it is *de facto* a point process (Daley and Vere-Jones (2008)). To keep the prior specification of change-points uninformative, it is desirable that conditional on the number of change-points, the positions of change-points be i.i.d. uniform on the domain. As for the number of change-points, it is important that the support of its distribution not be bounded, so as to allow for an arbitrarily large model complexity if warranted. The two requirements above are satisfied by a homogeneous Poisson process or HPP (Daley and Vere-Jones (2008)) with constant intensity λ^j. More precisely, the prior probability measure on $\left(\{c_p^j, \theta_p^j\}, \lambda^j\right)$ is constructed as follows:

$$\begin{cases} \lambda^j \sim \Gamma(\alpha^j, \beta^j), \\ \{c_p^j\} \big| \lambda^j \sim \text{HPP}(\lambda^j) \\ \theta_p^j[i] \big| \{c_p^j\}, \lambda^j \overset{\text{i.i.d}}{\sim} \log \mathcal{N}(0, \rho^j) \\ \forall (j,p) \neq (l,q)\ \theta_p^j \perp \theta_q^l, \end{cases}, \tag{16}$$

where we choose the Gamma distribution Γ as prior on the intensity λ^j for conjugacy, we assume all kernel hyper-parameters are positive as is often the case in practice,[9] the coordinates of the hyper-parameters of a kernel configuration are assumed i.i.d., and kernel hyper-parameters are assumed

7. That is, the functional form of the unconditional kernel k_k^j and its hyper-parameters.
8. We would like to stress that change-points do not introduce new input points or boundary times, but solely define a partition of the domain of each input dimension.
9. This may easily be relaxed if needed, for instance by putting normal priors on parameters that may be negative and log-normal priors on positive parameters.

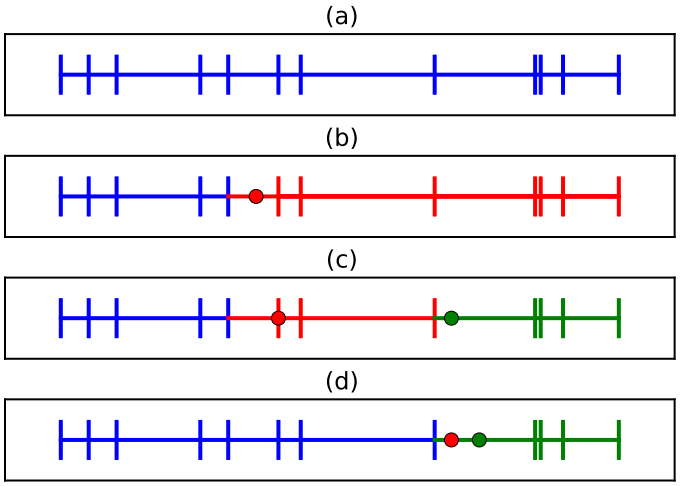

Figure 4: Effects of domain partition through change-points (coloured circles), on kernel membership. Each vertical bar corresponds to a distinct boundary time a_k^j. For the same collection of boundary times, we consider four scenarios: (a) no partition, (b) partition of the domain in two by a single change-point that does not coincide with any existing boundary time, (c) partition of the domain in three by two change-points, one of which coincides with an existing boundary time, and (d) partition of the domain in two by two distinct change-points. In each scenario, kernel membership is illustrated by colour-coding. The colour of the interval between two consecutive boundary times a_k^j and a_{k+1}^j reflects what kernel configuration drives the corresponding string; in particular, the colour of the vertical bar corresponding to boundary time a_{k+1}^j determines what kernel configuration should be used to compute the conditional distribution of the value of the *derivative string GP* $(z_t^j, z_t^{j\prime})$ at a_{k+1}^j, given its value at a_k^j.

independent between kernel configurations. Denoting the domain of the j-th input $[a^j, b^j]$, it follows from applying the laws of total expectation and total variance on Equation (16) that the expected number of change-points in the j-th dimension under our prior is

$$\mathrm{E}\left(\#\{c_p^j\}\right) = \left(b^j - a^j\right)\frac{\alpha^j}{\beta^j}, \tag{17}$$

and the variance of the number of change-points in the j-dimension under our prior is

$$\mathrm{Var}\left(\#\{c_p^j\}\right) = \left(b^j - a^j\right)\frac{\alpha^j}{\beta^j}\left(1 + \frac{\left(b^j - a^j\right)}{\beta^j}\right). \tag{18}$$

The two equations above may guide the user when setting the parameters α^j and β^j. For instance, these values may be set so that the expected number of change-points in a given input dimension be a fixed fraction of the number of boundary times in that input dimension, and so that the prior variance over the number of change-points be large enough that overall the prior isn't too informative.

We could have taken a different approach to construct our prior on change-points. In effect, assuming for the sake of the argument that the boundaries of the domain of the j-th input, namely a^j and b^j, are the first and last change-point in that input dimension, we note that the mapping

$$\left(\ldots, c_p^j, \ldots\right) \to \left(\ldots, p_p^j, \ldots\right) := \left(\ldots, \frac{c_{p+1}^j - c_p^j}{b^j - a^j}, \ldots\right)$$

defines a bijection between the set of possible change-points in the j-th dimension and the set of all discrete probability distributions. Thus, we could have placed as prior on $\left(\ldots, p_p^j, \ldots\right)$ a Dirichlet process (Ferguson (1973)), a Pitman-Yor process (Pitman and Yor (1997)), more generally *normalized completely random measures* (Kingman (1967)) or any other probability distribution over partitions. We prefer the point process approach primarily because it provides an easier way of expressing prior belief about model complexity through the expected number of change-points $\#\{c_p^j\}$, while remaining uninformative about positions thereof.

One might also be tempted to regard change-points in an input dimension j as inducing a partition, not of the domain $[a^j, b^j]$, but of the set of boundary times a_k^j in the same dimension, so that one may define a prior over kernel memberships through a prior over partitions of the set of boundary times. However, this approach would be inconsistent with the aim to learn local patterns in the data if the corresponding random measure is *exchangeable*. In effect, as boundary times are all input coordinates, local patterns may only arise in the data as a result of adjacent strings sharing kernel configurations. An exchangeable random measure would postulate a priori that two kernel membership assignments that have the same kernel configurations (i.e. the same number of configurations and the same set of hyper-parameters) and the same *number* of boundary times in each kernel cluster (although not exactly the same boundary times), are equally likely to occur, thereby possibly putting more probability mass on kernel membership assignments that do not respect boundary time adjacency. Unfortunately, *exchangeable* random measures (among which the Dirichlet process and the Pitman-Yor process) are by far more widely adopted by the machine learning community than non-exchangeable random measures. Thus, this approach might be perceived as overly complex. That being said, as noted by Foti and Williamson (2015), non-exchangeable normalized random measures may be regarded as Poisson point processes (with varying intensity functions) on some

augmented spaces, which makes this choice of prior specification somewhat similar, but stronger (that is more informative) than the one we adopt in this paper.

Before deriving the sampling algorithm, it is worth noting that the prior defined in Equation (16) does not admit a density with respect to the same base measure,[10] as the number of change-points $\#\{c_p^j\}$, and subsequently the number of kernel configurations, may vary from one sample to another. Nevertheless, the joint distribution over the data $\mathcal{D}$ and all other model parameters is well defined and, as we will see later, we may leverage reversible-jump MCMC techniques (Green (1995); Green and Hastie (2009)) to construct a Markov chain that converges to the posterior distribution.

5.2.4 OVERALL STRUCTURE OF THE MCMC SAMPLER

To ease notations, we denote $\boldsymbol{c}$ the set of all change-points in all input dimensions, we denote $\boldsymbol{n} = \left(\dots, \#\{c_p^j\}, \dots\right) \in \mathbb{N}^d$ the vector of the numbers of change-points in each input dimension, we denote $\boldsymbol{\theta}$ the set of kernel hyper-parameters,[11] and $\boldsymbol{\rho} := (\dots, \rho^j, \dots)$ the vector of variances of the independent log-normal priors on $\boldsymbol{\theta}$. We denote $\boldsymbol{\lambda} := (\dots, \lambda^j, \dots)$ the vector of change-points intensities, we denote $\boldsymbol{\alpha} := (\dots, \alpha^j, \dots)$ and $\boldsymbol{\beta} := (\dots, \beta^j, \dots)$ the vectors of parameters of the Gamma priors we placed on the change-points intensities across the d input dimensions, and we recall that $\mathbf{u}$ denotes the vector of likelihood parameters other than the values of the latent function f.

We would like to sample from the posterior distribution $p(\mathbf{f}, \mathbf{f}^*, \nabla\mathbf{f}, \nabla\mathbf{f}^* | \mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})$, where $\mathbf{f}$ and $\mathbf{f}^*$ are the vectors of values of the latent function f at training and test inputs respectively, and $\nabla\mathbf{f}, \nabla\mathbf{f}^*$ the corresponding gradients. Denoting $\mathbf{z}$ the vector of univariate *string GP* processes and their derivatives, evaluated at all distinct training and test input coordinate values, we note that to sample from $p(\mathbf{f}, \mathbf{f}^*, \nabla\mathbf{f}, \nabla\mathbf{f}^* | \mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})$, it suffices to sample from $p(\mathbf{z} | \mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})$, compute $\mathbf{f}$ and $\mathbf{f}^*$ using the link function, and compute the gradients using Equation (11). To sample from $p(\mathbf{z} | \mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})$, we may sample from the target distribution

$$\pi(\boldsymbol{n}, \boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{z}, \mathbf{u}) := p(\boldsymbol{n}, \boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{z}, \mathbf{u} | \mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho}), \tag{19}$$

and discard variables that are not of interest. As previously discussed, π is not absolutely continuous with respect to the same base measure, though we may still decompose it as

$$\pi(\boldsymbol{n}, \boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{z}, \mathbf{u}) = \frac{1}{p(\mathcal{D} | \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})} p(\boldsymbol{n} | \boldsymbol{\lambda}) p(\boldsymbol{\lambda} | \boldsymbol{\alpha}, \boldsymbol{\beta}) p(\boldsymbol{c} | \boldsymbol{n}) p(\boldsymbol{\theta} | \boldsymbol{n}, \boldsymbol{\rho}) p(\mathbf{u}) p(\mathbf{z} | \mathbf{c}, \boldsymbol{\theta}) p(\mathcal{D} | \mathbf{z}, \mathbf{u}), \tag{20}$$

where we use the notation $p(.)$ and $p(.|.)$ to denote probability measures rather than probability density functions or probability mass functions, and where product and scaling operations are usual measure operations. Before proceeding any further, we will introduce a slight re-parametrization of Equation (20) that will improve the inference scheme.

Let $\mathbf{n}_a = (\dots, \#\{a_k^j\}_k, \dots)$ be the vector of the numbers of unique boundary times in all d input dimensions. We recall from our prior on f that

$$p(\mathbf{z} | \mathbf{c}, \boldsymbol{\theta}) = \prod_{j=1}^{d} p\left(z^j_{a_0^j}, z^{j\prime}_{a_0^j}\right) \prod_{k=1}^{\boldsymbol{n}_a[j]-1} p\left(z^j_{a_k^j}, z^{j\prime}_{a_k^j} \middle| z^j_{a_{k-1}^j}, z^{j\prime}_{a_{k-1}^j}\right), \tag{21}$$

10. That is the joint prior probability measure is neither discrete, nor continuous.
11. To simplify the exposition, we assume without loss of generality that each kernel configuration has the same kernel functional form, so that configurations are defined by kernel hyper-parameters.

where each factor in the decomposition above is a bivariate Gaussian density whose mean vector and covariance matrix is obtained from the partitions $\mathbf{c}$, the kernel hyper-parameters $\boldsymbol{\theta}$, and the kernel membership scheme described in Section 5.2.3 and illustrated in Figure 4, and using Equations (6-7). Let ${}^{j}_{k}\mathbf{K}_{u;v}$ be the unconditional covariance matrix between $\left(z^j_u, z^{j\prime}_u\right)$ and $\left(z^j_v, z^{j\prime}_v\right)$ as per the unconditional kernel structure driving the string defined on the interval $[a^j_k, a^j_{k+1}[$. Let $\Sigma^j_0 := {}^{j}_{0}\mathbf{K}_{a^j_0;a^j_0}$ be the auto-covariance matrix of $\left(z^j_{a^j_0}, z^{j\prime}_{a^j_0}\right)$. Let

$$\Sigma^j_k := {}^{j}_{k}\mathbf{K}_{a^j_k;a^j_k} - {}^{j}_{k}\mathbf{K}_{a^j_k;a^j_{k-1}} {}^{j}_{k}\mathbf{K}^{-1}_{a^j_{k-1};a^j_{k-1}} {}^{j}_{k}\mathbf{K}^{T}_{a^j_k;a^j_{k-1}}$$

be the covariance matrix of $\left(z^j_{a^j_k}, z^{j\prime}_{a^j_k}\right)$ given $\left(z^j_{a^j_{k-1}}, z^{j\prime}_{a^j_{k-1}}\right)$, and

$$M^j_k = {}^{j}_{k}\mathbf{K}_{a^j_k;a^j_{k-1}} {}^{j}_{k}\mathbf{K}^{-1}_{a^j_{k-1};a^j_{k-1}}.$$

Finally, let $L^j_k := U^j_k (D^j_k)^{\frac{1}{2}}$ with $\Sigma^j_k = U^j_k D^j_k (U^j_k)^T$ the singular value decomposition (SVD) of Σ^j_k. We may choose to represent $\left(z^j_{a^j_0}, z^{j\prime}_{a^j_0}\right)$ as

$$\begin{bmatrix} z^j_{a^j_0} \\ z^{j\prime}_{a^j_0} \end{bmatrix} = L^j_0 x^j_0, \tag{22}$$

and for $k > 0$ we may also choose to represent $\left(z^j_{a^j_k}, z^{j\prime}_{a^j_k}\right)$ as

$$\begin{bmatrix} z^j_{a^j_k} \\ z^{j\prime}_{a^j_k} \end{bmatrix} = M^j_k \begin{bmatrix} z^j_{a^j_{k-1}} \\ z^{j\prime}_{a^j_{k-1}} \end{bmatrix} + L^j_k x^j_k, \tag{23}$$

where $\{x^j_k\}$ are independent bivariate standard normal vectors. Equations (22-23) provide an equivalent representation. In effect, we recall that if $Z = M + LX$, where $X \sim \mathcal{N}(0, I)$ is a standard multivariate Gaussian, M is a real vector, and L is a real matrix, then $Z \sim \mathcal{N}(M, LL^T)$. Equations (22-23) result from applying this result to $\left(z^j_{a^j_0}, z^{j\prime}_{a^j_0}\right)$ and $\left(z^j_{a^j_k}, z^{j\prime}_{a^j_k}\right) \Big| \left(z^j_{a^j_{k-1}}, z^{j\prime}_{a^j_{k-1}}\right)$. We note that at training time, M^j_k and L^j_k only depend on kernel hyper-parameters. Denoting $\mathbf{x}$ the vector of all x^j_k, $\mathbf{x}$ is a so-called 'whitened' representation of $\mathbf{z}$, which we prefer for reasons we will discuss shortly. In the whitened representation, the target distribution π is re-parameterized as

$$\pi(\boldsymbol{n}, \boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}) = \frac{1}{p(\mathcal{D}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})} p(\boldsymbol{n}|\boldsymbol{\lambda}) p(\boldsymbol{\lambda}|\boldsymbol{\alpha}, \boldsymbol{\beta}) p(\boldsymbol{c}|\boldsymbol{n}) p(\boldsymbol{\theta}|\boldsymbol{n}, \boldsymbol{\rho}) p(\mathbf{u}) p(\mathbf{x}) p(\mathcal{D}|\mathbf{x}, \mathbf{c}, \boldsymbol{\theta}, \mathbf{u}), \tag{24}$$

where the dependency of the likelihood term on the partitions and the hyper-parameters stems from the need to recover $\mathbf{z}$ and subsequently $\mathbf{f}$ from $\mathbf{x}$ through Equations (22) and (23). The whitened representation Equation (24) has two primary advantages. Firstly, it is robust to ill-conditioning of

Σ_k^j, which would typically occur when two adjacent boundary times are too close to each other. In the representation of Equation (20), as one needs to evaluate the density $p(\mathbf{z}|\mathbf{c}, \boldsymbol{\theta})$, ill-conditioning of Σ_k^j would result in numerical instabilities. In contrast, in the whitened representation, one needs to evaluate the density $p(\mathbf{x})$, which is that of i.i.d. standard Gaussians and as such can be evaluated robustly. Moreover, the SVD required to evaluate L_k^j is also robust to ill-conditioning of Σ_k^j, so that Equations (22) and (23) hold and can be robustly evaluated for degenerate Gaussians too. The second advantage of the whitened representation is that it improves mixing by establishing a link between kernel hyper-parameters and the likelihood.

Equation (24) allows us to cast our inference problem as a Bayesian model selection problem under a countable family of models indexed by $\boldsymbol{n} \in \mathbb{N}^d$, each defined on a different parameter subspace $\mathcal{C}_{\boldsymbol{n}}$, with cross-model normalizing constant $p(\mathcal{D}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})$, model probability driven by $p(\boldsymbol{n}|\boldsymbol{\lambda})p(\boldsymbol{\lambda}|\boldsymbol{\alpha}, \boldsymbol{\beta})$, model-specific prior $p(\boldsymbol{c}|\boldsymbol{n})p(\boldsymbol{\theta}|\boldsymbol{n}, \boldsymbol{\rho})p(\mathbf{u})p(\mathbf{x})$, and likelihood $p(\mathcal{D}|\mathbf{x}, \mathbf{c}, \boldsymbol{\theta}, \mathbf{u})$. Critically, it can be seen from Equation (24) that the conditional probability distribution

$$\pi(\boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}|\boldsymbol{n})$$

admits a density with respect to Lebesgue's measure on $\mathcal{C}_{\boldsymbol{n}}$.

Our setup is therefore analogous to that which motivated the seminal paper Green (1995), so that to sample from the posterior $\pi(\boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, \boldsymbol{n})$ we may use any Reversible-Jump Metropolis-Hastings (RJ-MH) scheme satisfying detailed balance and dimension-matching as described in section 3.3 of Green (1995). To improve mixing of the Markov chain, we will alternate between a *between-models* RJ-MH update with target distribution $\pi(\boldsymbol{n}, \boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u})$, and a *within-model* MCMC-within-Gibbs sampler with target distribution $\pi(\boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}|\boldsymbol{n})$. Constructing reversible-jump samplers by alternating between within-model sampling and between-models sampling is standard practice, and it is well-known that doing so yields a Markov chain that converges to the target distribution of interest (see Brooks et al., 2011, p. 50).

In a slight abuse of notation, in the following we might use the notations $p(.|.)$ and $p(.)$, which we previously used to denote probability measures, to refer to the corresponding probability density functions or probability mass functions.

5.2.5 WITHIN-MODEL UPDATES

We recall from Equation (24) that $\boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}|\mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho}, \boldsymbol{n}$ has probability density function

$$p(\boldsymbol{n}|\boldsymbol{\lambda})p(\boldsymbol{\lambda}|\boldsymbol{\alpha}, \boldsymbol{\beta})p(\boldsymbol{c}|\boldsymbol{n})p(\boldsymbol{\theta}|\boldsymbol{n}, \boldsymbol{\rho})p(\mathbf{u})p(\mathbf{x})p(\mathcal{D}|\mathbf{x}, \mathbf{c}, \boldsymbol{\theta}, \mathbf{u}), \tag{25}$$

up to a normalizing constant.

Updating $\boldsymbol{\lambda}$: By independence of the priors over $(\boldsymbol{\lambda}[j], \boldsymbol{n}[j])$, the distributions $\boldsymbol{\lambda}[j] \mid \boldsymbol{n}[j]$ are also independent, so that the updates may be performed in *parallel*. Moreover, recalling that the prior number of change-points in the j-th input dimension is Poisson distributed with intensity $\boldsymbol{\lambda}[j]\left(b^j - a^j\right)$, and by conjugacy of the Gamma distribution to the Poisson likelihood, it follows that

$$\boldsymbol{\lambda}[j] \mid \boldsymbol{n}[j] \sim \Gamma\left(\frac{\boldsymbol{n}[j]}{b^j - a^j} + \boldsymbol{\alpha}[j], 1 + \boldsymbol{\beta}[j]\right). \tag{26}$$

This update step has memory requirement and time complexity both constant in the number of training and test samples.

Updating $\mathbf{u}$: When the likelihood has additional parameters $\mathbf{u}$, they may be updated with a Metropolis-Hastings step. Denoting $q(\mathbf{u} \to \mathbf{u}')$ the proposal probability density function, the acceptance ratio reads

$$r_{\mathbf{u}} = \min \left(1, \frac{p(\mathbf{u}')p\left(\mathcal{D}|\mathbf{x}, \mathbf{c}, \boldsymbol{\theta}, \mathbf{u}'\right) q(\mathbf{u}' \to \mathbf{u})}{p(\mathbf{u})p\left(\mathcal{D}|\mathbf{x}, \mathbf{c}, \boldsymbol{\theta}, \mathbf{u}\right) q(\mathbf{u} \to \mathbf{u}')}\right). \tag{27}$$

In some cases however, it might be possible and more convenient to choose $p(\mathbf{u})$ to be conjugate to the likelihood $p\left(\mathcal{D}|\mathbf{x}, \mathbf{c}, \boldsymbol{\theta}, \mathbf{u}\right)$. For instance, in regression problems under a Gaussian noise model, we may take $\mathbf{u}$ to be the noise variance on which we may place an inverse-gamma prior. Either way, the computational bottleneck of this step is the evaluation of the likelihood $p\left(\mathcal{D}|\mathbf{x}, \mathbf{c}, \boldsymbol{\theta}, \mathbf{u}'\right)$, which in most cases can be done with a time complexity and memory requirement that are both linear in the number of training samples.

Updating $\mathbf{c}$: We update the positions of change-points sequentially using the Metropolis-Hastings algorithm, one input dimension j at a time, and for each input dimension we proceed in increasing order of change-points. The proposal new position for the change-point c_p^j is sampled uniformly at random on the interval $]c_{p-1}^j, c_{p+1}^j[$, where c_{p-1}^j (resp. c_{p+1}^j) is replaced by a^j (resp. b^j) for the first (resp. last) change-point. The acceptance probability of this proposal is easily found to be

$$r_{c_p^j} = \min \left(1, \frac{p\left(\mathcal{D}|\mathbf{x}, \mathbf{c}', \boldsymbol{\theta}, \mathbf{u}\right)}{p\left(\mathcal{D}|\mathbf{x}, \mathbf{c}, \boldsymbol{\theta}, \mathbf{u}\right)}\right), \tag{28}$$

where $\mathbf{c}'$ is identical to $\mathbf{c}$ except for the change-point to update. This step requires computing the factors $\{L_k^j, M_k^j\}$ corresponding to inputs in j-th dimension whose kernel configuration would change if the proposal were to be accepted, the corresponding vector of *derivative string GP* values $\mathbf{z}$, and the observation likelihood under the proposal $p\left(\mathcal{D}|\mathbf{x}, \mathbf{c}', \boldsymbol{\theta}, \mathbf{u}\right)$. The computational bottleneck of this step is therefore once again the evaluation of the new likelihood $p\left(\mathcal{D}|\mathbf{x}, \mathbf{c}', \boldsymbol{\theta}, \mathbf{u}\right)$.

Updating $\mathbf{x}$: The target conditional density of $\mathbf{x}$ is proportional to

$$p(\mathbf{x})p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u}). \tag{29}$$

Recalling that $p(\mathbf{x})$ is a multivariate standard normal, it follows that the form of Equation (29) makes it convenient to use elliptical slice sampling (Murray et al. (2010)) to sample from the unormalized conditional $p(\mathbf{x})p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})$. The two bottlenecks of this update step are sampling a new proposal from $p(\mathbf{x})$ and evaluating the likelihood $p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})$. Sampling from the multivariate standard normal $p(\mathbf{x})$ may be *massively parallelized*, for instance by using GPU Gaussian random number generators. When no parallelism is available, the overall time complexity reads $\mathcal{O}\left(\sum_{j=1}^d \mathbf{n}_a[j]\right)$, where we recall that $\mathbf{n}_a[j]$ denotes the number of distinct training and testing input coordinates in the j-th dimension. In particular, if we denote N the total number of training and testing d-dimensional input samples, then $\sum_{j=1}^d \mathbf{n}_a[j] \leq dN$, although for many classes of data sets with sparse input values such as images, where each input (single-colour pixel value) may have at most 256 distinct values, we might have $\sum_{j=1}^d \mathbf{n}_a[j] \ll dN$. As for the memory required to sample from $p(\mathbf{x})$, it grows proportionally to the size of $\mathbf{x}$, that is in $\mathcal{O}\left(\sum_{j=1}^d \mathbf{n}_a[j]\right)$. In regards to the evaluation of the likelihood $p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})$, as previously discussed its resource requirements are application-specific, but it will typically have time complexity that grows in $\mathcal{O}\left(N\right)$ and memory requirement that grows in $\mathcal{O}\left(dN\right)$. For instance, the foregoing resource requirements always hold

for i.i.d. observation models such as in nonparametric regression and nonparametric classification problems.

Updating $\boldsymbol{\theta}$: We note from Equation (25) that the conditional distribution of $\boldsymbol{\theta}$ given everything else has unormalized density

$$p(\boldsymbol{\theta}|\boldsymbol{n}, \boldsymbol{\rho})p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u}), \tag{30}$$

which we may choose to represent as

$$p(\log \boldsymbol{\theta}|\boldsymbol{n}, \boldsymbol{\rho})p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \log \boldsymbol{\theta}, \mathbf{u}). \tag{31}$$

As we have put independent log-normal priors on the coordinates of $\boldsymbol{\theta}$ (see Equation 16), we may once again use elliptical slice sampling to sample from $\log \boldsymbol{\theta}$ before taking the exponential. The time complexity of generating a new sample from $p(\log \boldsymbol{\theta}|\boldsymbol{n}, \boldsymbol{\rho})$ will typically be at most linear in the total number of distinct kernel hyper-parameters. Overall, the bottleneck of this update is the evaluation of the likelihood $p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \log \boldsymbol{\theta}, \mathbf{u})$. In this update, the latter operation requires recomputing the factors M_k^j and L_k^j of Equations (22) and (23), which requires computing and taking the SVD of unrelated 2×2 matrices, computations we may perform in *parallel*. Once the foregoing factors have been computed, we evaluate $\mathbf{z}$, the *derivative string GP* values at boundary times, parallelizing over input dimensions, and running a sequential update within an input dimension using Equations (22) and (23). Updating $\mathbf{z}$ therefore has time complexity that is, in the worst case where no distributed computing is available, $\mathcal{O}(dN)$, and $\mathcal{O}(N)$ when there are up to d computing cores. The foregoing time complexity will also be that of this update step, unless the observation likelihood is more expensive to evaluate. The memory requirement, as in previous updates, is $\mathcal{O}(dN)$.

Overall resource requirement: To summarize previous remarks, the overall computational bottleneck of a *within-model* iteration is the evaluation of the likelihood $p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})$. For i.i.d. observation models such as classification and regression problems for instance, the corresponding time complexity grows in $\mathcal{O}(N)$ when d computing cores are available, or $\mathcal{O}(dN)$ otherwise, and the memory requirement grows in $\mathcal{O}(dN)$.

5.2.6 BETWEEN-MODELS UPDATES

Our reversible-jump Metropolis-Hastings update proceeds as follows. We choose an input dimension, say j, uniformly at random. If j has no change-points, that is $\mathbf{n}[j] = 0$, we randomly choose between not doing anything, and adding a change-point, each outcome having the same probability. If $\mathbf{n}[j] > 0$, we either do nothing, add a change-point, or delete a change-point, each outcome having the same probability of occurrence.

Whenever we choose not to do anything, the acceptance ratio is easily found to be one:

$$r_{j0} = 1. \tag{32}$$

Whenever we choose to add a change-point, we sample the position c_*^j of the proposal new change-point uniformly at random on the domain $[a^j, b^j]$ of the j-th input dimension. This proposal will almost surely break an existing kernel membership cluster, say the p-th, into two; that is $c_p^j < c_*^j < c_{p+1}^j$ where we may have $a^j = c_p^j$ and/or $b^j = c_{p+1}^j$. In the event c_*^j coincides with an existing change-point, which should happen with probability 0, we do nothing. When adding a change-point, we sample a new vector of hyper-parameters θ_*^j from the log-normal prior of Equation (16), and we

propose as hyper-parameters for the tentative new clusters $[c_p^j, c_*^j[$ and $[c_*^j, c_{p+1}^j[$ the vectors $\theta^j_{\text{add-left}}$ and $\theta^j_{\text{add-right}}$ defined as

$$\log \theta^j_{\text{add-left}} := \cos(\alpha) \log \theta_p^j - \sin(\alpha) \log \theta_*^j \tag{33}$$

and

$$\log \theta^j_{\text{add-right}} := \sin(\alpha) \log \theta_p^j + \cos(\alpha) \log \theta_*^j \tag{34}$$

respectively, where $\alpha \in [0, \frac{\pi}{2}]$ and θ_p^j is the vector of hyper-parameters currently driving the kernel membership defined by the cluster $[c_p^j, c_{p+1}^j[$. We note that if θ_p^j is distributed as per the prior in Equation (16) then $\theta^j_{\text{add-left}}$ and $\theta^j_{\text{add-right}}$ are i.i.d. distributed as per the foregoing prior. More generally, this elliptical transformation determines the extent to which the new proposal kernel configurations should deviate from the current configuration θ_p^j. α is restricted to $[0, \frac{\pi}{2}]$ so as to give a positive weight the the current vector of hyper-parameters θ_p^j. When $\alpha = 0$, the left hand-side cluster $[c_p^j, c_*^j[$ will fully exploit the current kernel configuration, while the right hand-side cluster $[c_*^j, c_{p+1}^j[$ will use the prior to explore a new set of hyper-parameters. When $\alpha = \frac{\pi}{2}$ the reverse occurs. To preserve symmetry between the left and right hand-side kernel configurations, we choose

$$\alpha = \frac{\pi}{4}. \tag{35}$$

Whenever we choose to delete a change-point, we choose an existing change-point uniformly at random, say c_p^j. Deleting c_p^j, would merge the clusters $[c_{p-1}^j, c_p^j[$ and $[c_p^j, c_{p+1}^j[$, where we may have $a^j = c_{p-1}^j$ and/or $b^j = c_{p+1}^j$. We propose as vector of hyper-parameters for the tentative merged cluster $[c_{p-1}^j, c_{p+1}^j[$ the vector $\theta^j_{\text{del-merged}}$ satisfying:

$$\log \theta^j_{\text{del-merged}} = \cos(\alpha) \log \theta_{p-1}^j + \sin(\alpha) \log \theta_p^j, \tag{36}$$

which together with

$$\log \theta^j_{\text{del-}*} = -\sin(\alpha) \log \theta_{p-1}^j + \cos(\alpha) \log \theta_p^j, \tag{37}$$

constitute the inverse of the transformation defined by Equations (33) and (34).

Whenever a proposal to add or delete a change-point occurs, the factors L_k^j and M_k^j that would be affected by the change in kernel membership structure are recomputed, and so are the affected coordinates of $\mathbf{z}$.

This scheme satisfies the reversibility and dimension-matching requirements of Green (1995). Moreover, the absolute value of the Jacobian of the mapping

$$\left(\log \theta_p^j,\ \log \theta_*^j\right) \to \left(\log \theta^j_{\text{add-left}},\ \log \theta^j_{\text{add-right}}\right)$$

of the move to add a change-point in $[c_p^j, c_{p+1}^j[$ reads

$$\left| \frac{\partial \left(\log \theta^j_{\text{add-left}}, \log \theta^j_{\text{add-right}} \right)}{\partial \left(\log \theta_p^j, \log \theta_*^j \right)} \right| = 1. \tag{38}$$

Similarly, the absolute value of the Jacobian of the mapping corresponding to a move to delete change-point c_p^j, namely

$$\left(\log \theta_{p-1}^j,\ \log \theta_p^j\right) \to \left(\log \theta_{\text{del-merged}}^j,\ \log \theta_{\text{del-*}}^j\right),$$

reads:

$$\left| \frac{\partial \left(\log \theta_{\text{del-merged}}^j, \log \theta_{\text{del-*}}^j\right)}{\partial \left(\log \theta_{p-1}^j, \log \theta_p^j\right)} \right| = 1. \tag{39}$$

Applying the standard result Equation (8) of Green (1995), the acceptance ratio of the move to add a change-point is found to be

$$r_{j+} = \min\left(1, \frac{p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}_+, \boldsymbol{\theta}_+, \mathbf{u})}{p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})} \frac{\boldsymbol{\lambda}[j]\left(b^j - a^j\right)}{1 + \mathbf{n}[j]} \frac{p_{\log \theta^j}\left(\log \theta_{\text{add-left}}^j\right) p_{\log \theta^j}\left(\log \theta_{\text{add-right}}^j\right)}{p_{\log \theta^j}\left(\log \theta_p^j\right) p_{\log \theta^j}\left(\log \theta_*^j\right)}\right) \tag{40}$$

where $p_{\log \theta^j}$ is the prior over log hyper-parameters in the j-th input dimension (as per the prior specification Equation 16), which we recall is i.i.d. centred Gaussian with variance $\boldsymbol{\rho}[j]$, and $\boldsymbol{c}_+$ and $\boldsymbol{\theta}_+$ denote the proposal new vector of change-points and the corresponding vector of hyper-parameters. The three coloured terms in the acceptance probability are very intuitive. The green term $\frac{p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}_+, \boldsymbol{\theta}_+, \mathbf{u})}{p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})}$ represents the fit improvement that would occur if the new proposal is accepted. In the red term $\frac{\boldsymbol{\lambda}[j]\left(b^j - a^j\right)}{1+\mathbf{n}[j]}$, $\boldsymbol{\lambda}[j]\left(b^j - a^j\right)$ represents the average number of change-points in the j-th input dimension as per the HPP prior, while $1 + \mathbf{n}[j]$ corresponds to the proposal new number of change-points in the j-th dimension, so that the whole red term acts as a complexity regulariser. Finally, the blue term $\frac{p_{\log \theta^j}\left(\log \theta_{\text{add-left}}^j\right) p_{\log \theta^j}\left(\log \theta_{\text{add-right}}^j\right)}{p_{\log \theta^j}\left(\log \theta_p^j\right) p_{\log \theta^j}\left(\log \theta_*^j\right)}$ plays the role of hyper-parameters regulariser.

Similarly, the acceptance ratio of the move to delete change-point c_p^j, thereby changing the number of change-points in the j-th input dimension from $\mathbf{n}[j]$ to $\mathbf{n}[j] - 1$, is found to be

$$r_{j-} = \min\left(1, \frac{p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}_-, \boldsymbol{\theta}_-, \mathbf{u})}{p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})} \frac{\mathbf{n}[j]}{\boldsymbol{\lambda}[j]\left(b^j - a^j\right)} \frac{p_{\log \theta^j}\left(\log \theta_{\text{del-merged}}^j\right) p_{\log \theta^j}\left(\log \theta_{\text{del-*}}^j\right)}{p_{\log \theta^j}\left(\log \theta_{p-1}^j\right) p_{\log \theta^j}\left(\log \theta_p^j\right)}\right), \tag{41}$$

where $\boldsymbol{c}_-$ and $\boldsymbol{\theta}_-$ denote the proposal new vector of change-points and the corresponding vector of hyper-parameters. Once more, each coloured term plays the same intuitive role as its counterpart in Equation (40).

Overall resource requirement: The bottleneck of between-models updates is the evaluation of the new likelihoods $p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}_+, \boldsymbol{\theta}_+, \mathbf{u})$ or $p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}_-, \boldsymbol{\theta}_-, \mathbf{u})$, whose resource requirements, which are the same as those of within-models updates, we already discussed.

Algorithm 2 summarises the proposed MCMC sampler.

5.3 Multi-Output Problems

Although we have restricted ourselves to cases where the likelihood model depends on a single real-valued function for brevity and to ease notations, cases where the likelihood depends on vector-valued functions, or equivalently multiple real-valued functions, present no additional theoretical or

Algorithm 2 MCMC sampler for nonparametric Bayesian inference of a real-valued latent function under a *string GP* prior

Inputs: Likelihood model $p(\mathcal{D}|\mathbf{f}, \mathbf{u})$, link function ϕ, training data $\mathcal{D}$, test inputs, type of unconditional kernel, prior parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho}$.
Outputs: Posterior samples of the values of the latent function at training and test inputs $\mathbf{f}$ and $\mathbf{f}^*$, and the corresponding gradients $\nabla\mathbf{f}$ and $\nabla\mathbf{f}^*$.

Step 0: Set $\boldsymbol{n} = 0$ and $\boldsymbol{c} = \emptyset$, and sample $\boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}$ from their priors.
repeat
Step 1: Perform a within-model update.
1.1: Update each $\boldsymbol{\lambda}[j]$ by sampling from the Gamma distribution in Equation (26).
1.2: Update $\mathbf{u}$, the vector of other likelihood parameters, if any, using Metropolis-Hastings (MH) with proposal q and acceptance ratio Equation (27) or by sampling directly from the posterior when $p(\mathbf{u})$ is conjugate to the likelihood model.
1.3: Update $\boldsymbol{\theta}$, using elliptical slice sampling (ESS) with target distribution Equation (31), and record the newly computed factors $\{L_k^j, M_k^j\}$ that relate $\mathbf{z}$ to its whitened representation $\mathbf{x}$.
1.4: Update $\mathbf{x}$ using ESS with target distribution Equation (29).
1.5: Update change-point positions $\mathbf{c}$ sequentially using MH, drawing a proposal update for c_p^j uniformly at random on $]c_{p-1}^J, c_{p+1}^j[$, and accepting the update with probability $r_{c_p^j}$ (defined Equation 28). On accept, update the factors $\{L_k^j, M_k^j\}$.
Step 2: Perform a between-models update.
2.1: Sample a dimension to update, say j, uniformly at random.
2.2: Consider adding or deleting a change-point
if $\mathbf{n}[j] = 0$ **then**
Randomly choose to add a change-point with probability $1/2$.
if we should consider adding a change-point **then**
Construct proposals to update following Section 5.2.6.
Accept proposals with probability r_+^j (see Equation 40).
end if
else
Randomly choose to add/delete a change-point with probability $1/3$.
if we should consider adding a change-point **then**
Construct proposals to update following Section 5.2.6.
Accept proposals with probability r_+^j (see Equation 40).
else if we should consider deleting a change-point **then**
Construct proposals to update following Section 5.2.6.
Accept proposals with probability r_-^j (see Equation 41).
else
Continue.
end if
end if
Step 3: Compute $\mathbf{f}, \mathbf{f}^*, \nabla\mathbf{f}$ and $\nabla\mathbf{f}^*$, first recovering $\mathbf{z}$ from $\mathbf{x}$, and then recalling that $f(x) = \phi\left(z^1_{x[1]}, \ldots, z^d_{x[d]}\right)$ and $\nabla f(x) = \left(z^{1\prime}_{x[1]}\frac{\partial\phi}{\partial x[1]}(x), \ldots, z^{d\prime}_{x[d]}\frac{\partial\phi}{\partial x[d]}(x)\right)$.
until enough samples are generated after mixing.

practical challenge. We may simply put independent *string GP* priors on each of the latent functions. An MCMC sampler almost identical to the one introduced herein may be used to sample from the posterior. All that is required to adapt the proposed MCMC sampler to multi-outputs problems is to redefine $\mathbf{z}$ to include all univariate *derivative string GP* values across input dimensions and across latent functions, perform step 1.1 of Algorithm 2 for each of the latent function, and update step 2.1 so as to sample uniformly at random not only what dimension to update but also what latent function. Previous analyses and derived acceptance ratios remain unchanged. The resource requirements of the resulting multi-outputs MCMC sampler on a problem with K latent functions, N training and test d-dimensional inputs, are the same as those of the MCMC sampler for a single output (Algorithm 2) with N training and test dK-dimensional inputs. The time complexity is $\mathcal{O}(N)$ when dK computing cores are available, $\mathcal{O}(dKN)$ when no distributed computing is available, and the memory requirement becomes $\mathcal{O}(dKN)$.

5.4 Flashback to Small Scale GP Regressions with String GP Kernels

In Section 5.1 we discussed maximum marginal likelihood inference in Bayesian nonparametric regressions under additively separable *string GP* priors, or GP priors with *string GP* covariance functions. We proposed learning the positions of boundary times, conditional on their number, jointly with kernel hyper-parameters and noise variances by maximizing the marginal likelihood using gradient-based techniques. We then suggested learning the number of strings in each input dimension by trading off goodness-of-fit with model simplicity using information criteria such as AIC and BIC. In this section, we propose a fully Bayesian nonparametric alternative.

Let us consider the Gaussian process regression model

$$y_i = f(x_i) + \epsilon_i, \quad f \sim \mathcal{GP}\left(0, k_{\text{SGP}}(.,.)\right), \quad \epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right), \tag{42}$$

$$x_i \in [a^1, b^1] \times \cdots \times [a^d, b^d], \quad y_i, \epsilon_i \in \mathbb{R}, \tag{43}$$

where k_{SGP} is the covariance function of some *string GP* with boundary times $\{a_k^j\}$ and corresponding unconditional kernels $\{k_k^j\}$ in the j-th input dimension. It is worth stressing that we place a GP (not *string GP*) prior on the latent function f, but the covariance function of the GP is a *string GP* covariance function (as discussed in Section 4.2 and as derived in Appendix I). Of course when the *string GP* covariance function k_{SGP} is separately additive, the two functional priors are the same. However, we impose no restriction on the link function of the *string GP* that k_{SGP} is the covariance function of, other than continuous differentiability. To make full Bayesian nonparametric inference, we may place on the boundary times $\{a_k^j\}$ independent homogeneous Poisson process priors, each with intensity λ^j. Similarly to the previous section (Equation 16) our full prior specification of the *string GP* kernel reads

$$\begin{cases} \lambda^j \sim \Gamma(\alpha^j, \beta^j), \\ \{a_k^j\} \big| \lambda^j \sim \text{HPP}(\lambda^j) \\ \theta_k^j[i] \big| \{a_k^j\}, \lambda^j \overset{\text{i.i.d}}{\sim} \log \mathcal{N}(0, \rho^j) \\ \forall (j,k) \neq (l,p)\ \theta_k^j \perp \theta_p^l, \end{cases}, \tag{44}$$

where θ_k^j is the vector of hyper-parameters driving the unconditional kernel k_k^j. The method developed in the previous section and the resulting MCMC sampling scheme (Algorithm 2) may be

reused to sample from the posterior over function values, pending the following two changes. First, gradients $\nabla \mathbf{f}$ and $\nabla \mathbf{f}^*$ are no longer necessary. Second, we may work with function values $(\mathbf{f}, \mathbf{f}^*)$ directly (that is in the original as opposed to whitened space). The resulting (Gaussian) distribution of function values $(\mathbf{f}, \mathbf{f}^*)$ conditional on all other variables is then analytically derived using standard Gaussian identities, like it is done in vanilla Gaussian process regression, so that the within-model update of $(\mathbf{f}, \mathbf{f}^*)$ is performed using a single draw from a multivariate Gaussian.

This approach to model complexity learning is advantageous over the information criteria alternative of Section 5.1 in that it scales better with large input-dimensions. Indeed, rather than performing complete maximum marginal likelihood inference a number of times that grows exponentially with the input dimension, the approach of this section alternates between exploring a new combination of numbers of kernel configurations in each input dimension, and exploring function values and kernel hyper-parameters (given their number). That being said, this approach should only be considered as an alternative to commonly used kernels for *small scale* regression problems to enable the learning of local patterns. Crucially, it scales as poorly as the *standard GP paradigm*, and Algorithm 2 should be preferred for large scale problems.

6. Experiments

We now move on to presenting empirical evidence for the efficacy of *string GPs* in coping with local patterns in data sets, and in doing so in a scalable manner. Firstly we consider maximum marginal likelihood inference on two small scale problems exhibiting local patterns. We begin with a toy experiment that illustrates the limitations of the *standard GP paradigm* in extrapolating and interpolating simple local periodic patterns. Then, we move on to comparing the accuracy of Bayesian nonparametric regression under a *string GP* prior to that of the standard Gaussian process regression model and existing mixture-of-experts alternatives on the motorcycle data set of Silverman (1985), commonly used for the local patterns and heteroskedasticity it exhibits. Finally, we illustrate the performance of the previously derived MCMC sampler on two large scale Bayesian inference problems, namely the prediction of U.S. commercial airline arrival delays of Hensman et al. (2013) and a new large scale dynamic asset allocation problem.

6.1 Extrapolation and Interpolation of Synthetic Local Patterns

In our first experiment, we illustrate a limitation of the standard approach consisting of postulating a global covariance structure on the domain, namely that this approach might result in unwanted global extrapolation of local patterns, and we show that this limitation is addressed by the *string GP paradigm*. To this aim, we use 2 toy regression problems. We consider the following functions:

$$f_0(t) = \begin{cases} \sin(60\pi t) & t \in [0, 0.5] \\ \frac{15}{4}\sin(16\pi t) & t \in]0.5, 1] \end{cases}, \quad f_1(t) = \begin{cases} \sin(16\pi t) & t \in [0, 0.5] \\ \frac{1}{2}\sin(32\pi t) & t \in]0.5, 1] \end{cases}. \tag{45}$$

f_0 (resp. f_1) undergoes a sharp (resp. mild) change in frequency and amplitude at $t = 0.5$. We consider using their restrictions to $[0.25, 0.75]$ for training. We sample those restrictions with frequency 300, and we would like to extrapolate the functions to the rest of their domains using Bayesian nonparametric regression.

We compare marginal likelihood *string GP* regression models, as described in Section 5.1, to vanilla GP regression models using popular and expressive kernels. All *string GP* models have two strings and the partition is learned in the marginal likelihood maximisation. Figure 5 illustrates

plots of the posterior means for each kernel used, and Table 2 compares predictive errors. Overall, it can be noted that the *string GP kernel* with the periodic kernel (MacKay (1998)) as building block outperforms competing kernels, including the expressive spectral mixture kernel

$$k_{\text{SM}}(r) = \sum_{k=1}^{K} \sigma_k^2 \exp(-2\pi^2 r^2 \gamma_k^2) \cos(2\pi r \mu_k)$$

of Wilson and Adams (2013) with $K = 5$ mixture components.[12]

The comparison between the spectral mixture kernel and the string spectral mixture kernel is of particular interest, since spectral mixture kernels are pointwise dense in the family of stationary kernels, and thus can be regarded as flexible enough for learning *stationary* kernels from the data. In our experiment, the *string spectral mixture kernel* with a single mixture component per string significantly outperforms the spectral mixture kernel with 5 mixture components. This intuitively can be attributed to the fact that, regardless of the number of mixture components in the spectral mixture kernel, the learned kernel must account for both types of patterns present in each training data set. Hence, each local extrapolation on each side of 0.5 will attempt to make use of both amplitudes and both frequencies evidenced in the corresponding training data set, and will struggle to recover the true local sine function. We would expect that the performance of the spectral mixture kernel in this experiment will not improve drastically as the number of mixture components increases. However, under a *string GP* prior, the left and right hand side strings are independent conditional on the (unknown) boundary conditions. Therefore, when the *string GP* domain partition occurs at time 0.5, the training data set on $[0.25, 0.5]$ influences the hyper-parameters of the string to the right of 0.5 only to the extent that both strings should agree on the value of the latent function and its derivative at 0.5. To see why this is a weaker condition, we consider the family of pair of functions:

$$(\alpha\omega_1 \sin(\omega_2 t),\ \alpha\omega_2 \sin(\omega_1 t)), \quad \omega_i = 2\pi k_i, \quad k_i \in \mathbb{N}, \quad \alpha \in \mathbb{R}.$$

Such functions always have the same value and derivative at 0.5, regardless of their frequencies, and they are plausible GP paths under a spectral mixture kernel with one single mixture component ($\mu_k = k_i$ and $\gamma_k \ll 1$), and under a periodic kernel. As such it is not surprising that extrapolation under a string spectral mixture kernel or a string periodic kernel should perform well.

To further illustrate that *string GPs* are able to learn local patterns that GPs with commonly used and expressive kernels can't, we consider interpolating two bivariate functions f_2 and f_3 that exhibit local patterns. The functions are defined as:

$$\forall u, v \in [0.0, 1.0] \quad f_2(u, v) = f_0(u) f_1(v), \quad f_3(u, v) = \sqrt{f_0(u)^2 + f_1(v)^2}. \tag{46}$$

We consider recovering the original functions as the posterior mean of a GP regression model trained on $[0.0, 0.4] \cup [0.6, 1.0] \times [0.0, 0.4] \cup [0.6, 1.0]$. Each bivariate kernel used is a product of two univariate kernels in the same family, and we used standard Kronecker techniques to speed-up inference (see Saatchi, 2011, p.134). The univariate kernels we consider are the same as previously. Each univariate *string GP* kernel has one change-point (two strings) whose position is learned by maximum marginal likelihood. Results are illustrated in Figures 6 and 7. Once again it can be seen that unlike any competing kernel, the product of string periodic kernels recover both functions almost perfectly. In particular, it is impressive to see that, despite f_3 not being a separable function,

12. The sparse spectrum kernel of Lazaro-Gredilla et al. (2010) can be thought of as the special case $\gamma_k \ll 1$.

a product of string periodic kernels recovered it almost perfectly. The interpolations performed by the spectral mixture kernel (see Figures 6 and 7) provide further evidence for our previously developed narrative: the spectral mixture kernel tries to blend all local patterns found in the training data during the interpolation. The periodic kernel learns a single global frequency characteristic of the whole data set, ignoring local patterns, while the squared exponential, Matérn and rational quadratic kernels merely attempt to perform interpolation by smoothing.

Although we used synthetic data to ease illustrating our argument, it is reasonable to expect that in real-life problems the bigger the data set, the more likely there might be local patterns that should not be interpreted as noise and yet are not indicative of the data set as whole.

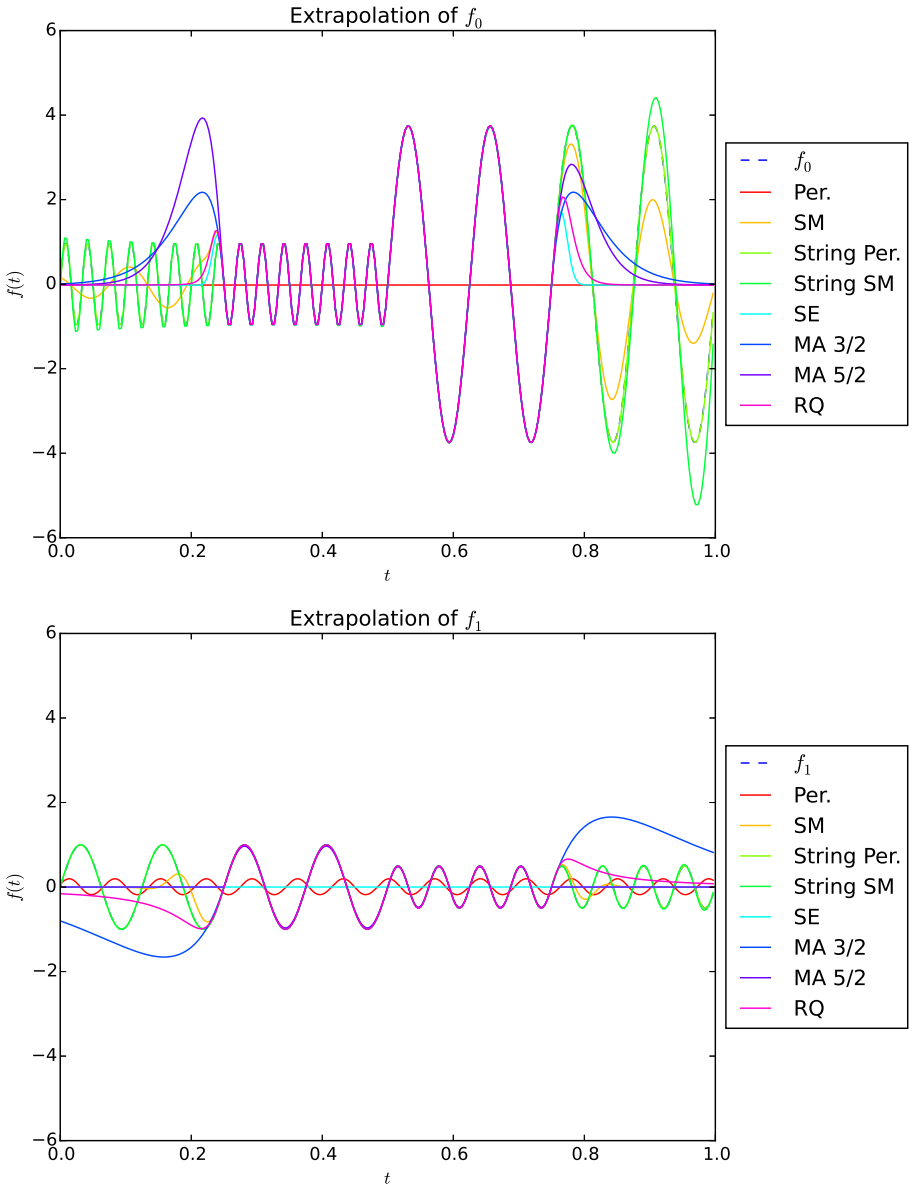

Figure 5: Extrapolation of two functions f_0 and f_1 through Bayesian nonparametric regression under *string GP* priors and vanilla GP priors with popular and expressive kernels. Each model is trained on $[0.25, 0.5]$ and extrapolates to $[0, 1.0]$.

	Absolute Error		Squared Error	
Kernel	f_0	f_1	f_0	f_1
Squared exponential	1.44 ± 2.40	0.48 ± 0.58	3.50 ± 9.20	0.31 ± 0.64
Rational quadratic	1.39 ± 2.31	0.51 ± 0.83	3.28 ± 8.79	0.43 ± 1.15
Matérn 3/2	1.63 ± 2.53	1.26 ± 1.37	4.26 ± 11.07	2.06 ± 3.55
Matérn 5/2	1.75 ± 2.77	0.48 ± 0.58	5.00 ± 12.18	0.31 ± 0.64
Periodic	1.51 ± 2.45	0.53 ± 0.60	3.79 ± 9.62	0.37 ± 0.72
Spec. Mix. (5 comp.)	0.75 ± 1.15	0.39 ± 0.57	0.94 ± 2.46	0.24 ± 0.58
String Spec. Mix. (2 strings, 1 comp.)	0.23 ± 0.84	0.01 ± 0.03	0.21 ± 1.07	$\mathbf{0.00 \pm 0.00}$
String Periodic	$\mathbf{0.02 \pm 0.02}$	$\mathbf{0.00 \pm 0.01}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$

Table 2: Predictive accuracies in the extrapolation of the two functions f_0 and f_1 of Section 6.1 through Bayesian nonparametric regression under *string GP* priors and vanilla GP priors with popular and expressive kernels. Each model is trained on $[0.25, 0.5]$ and extrapolates to $[0, 1.0]$. The predictive errors are reported as average $\pm$ 2 standard deviations.

6.2 Small Scale Heteroskedastic Regression

In our second experiment, we consider illustrating the advantage of the *string GP paradigm* over the *standard GP paradigm*, but also over the alternatives of Kim et al. (2005), Gramacy and Lee (2008), Tresp (2000) and Deisenroth and Ng (2015) that consist of considering independent GP experts on disjoint parts of the domain or handling disjoint subsets of the data. Using the motorcycle data set of Silverman (1985), commonly used for the local patterns and heteroskedasticity it exhibits, we show that our approach outperforms the aforementioned competing alternatives, thereby providing empirical evidence that the collaboration between consecutive GP experts introduced in the *string GP paradigm* vastly improves predictive accuracy and certainty in regression problems with local patterns. We also illustrate learning of the derivative of the latent function, solely from noisy measurements of the latent function.

The observations consist of accelerometer readings taken through time in an experiment on the efficacy of crash helmets. It can be seen at a glance in Figure 8 that the data set exhibits roughly 4 regimes. Firstly, between 0ms and 15ms the acceleration was negligible. Secondly, the impact slowed down the helmet, resulting in a sharp deceleration between 15ms and 28ms. Thirdly, the helmet seems to have bounced back between 28ms and 32ms, before it finally gradually slowed down and came to a stop between 32ms and 60ms. It can also be noted that the measurement noise seems to have been higher in the second half of the experiment.

We ran 50 independent random experiments, leaving out 5 points selected uniformly at random from the data set for prediction, the rest being used for training. The models we considered include the vanilla GP regression model, the *string GP* regression model with marginal maximum likelihood inference as described in Section 5.1, mixtures of independent GP experts acting on disjoint subsets of the data both for training and testing, the Bayesian committee machine (Tresp (2000)), and the robust Bayesian committee machine (Deisenroth and Ng (2015)). We considered *string GPs* with 4 and 6 strings whose boundary times are learned as part of the maximum likelihood inference. For consistency, we used the resulting partitions of the domain to define the independent experts in the competing alternatives we considered. The Matérn 3/2 kernel was used throughout. The results are reported in Table 3. To gauge the ability of each model to capture the physics of the helmets crash experiment, we have also trained all models with all data points. The results are illustrated in Figures 8 and 9.

It can be seen at a glance from Figure 9 that mixtures of independent GP experts are inappropriate for this experiment as i) the resulting posterior means exhibit discontinuities (for instance at $t = 30$ms and $t = 40$ms) that are inconsistent with the physics of the underlying phenomenon, and ii) they overfit the data towards the end. The foregoing discontinuities do not come as a surprise as each GP regression expert acts on a specific subset of the domain that is disjoint from the ones used by the other experts, both for training and prediction. Thus, there is no guarantee of consistency between expert predictions at the boundaries of the domain partition. Another perspective to this observation is found in noting that postulating independent GP experts, each acting on an element of a partition of the domain, is equivalent to putting as prior on the whole function a stochastic process that is discontinuous at the boundaries of the partition. Thus, the posterior stochastic process should not be expected to be continuous at the boundaries of the domain either.

This discontinuity issue is addressed by the Bayesian committee machine (BCM) and the robust Bayesian committee machine (rBCM) because, despite each independent expert being trained on a disjoint subset of the data, each expert is tasked with making predictions about all test inputs, not

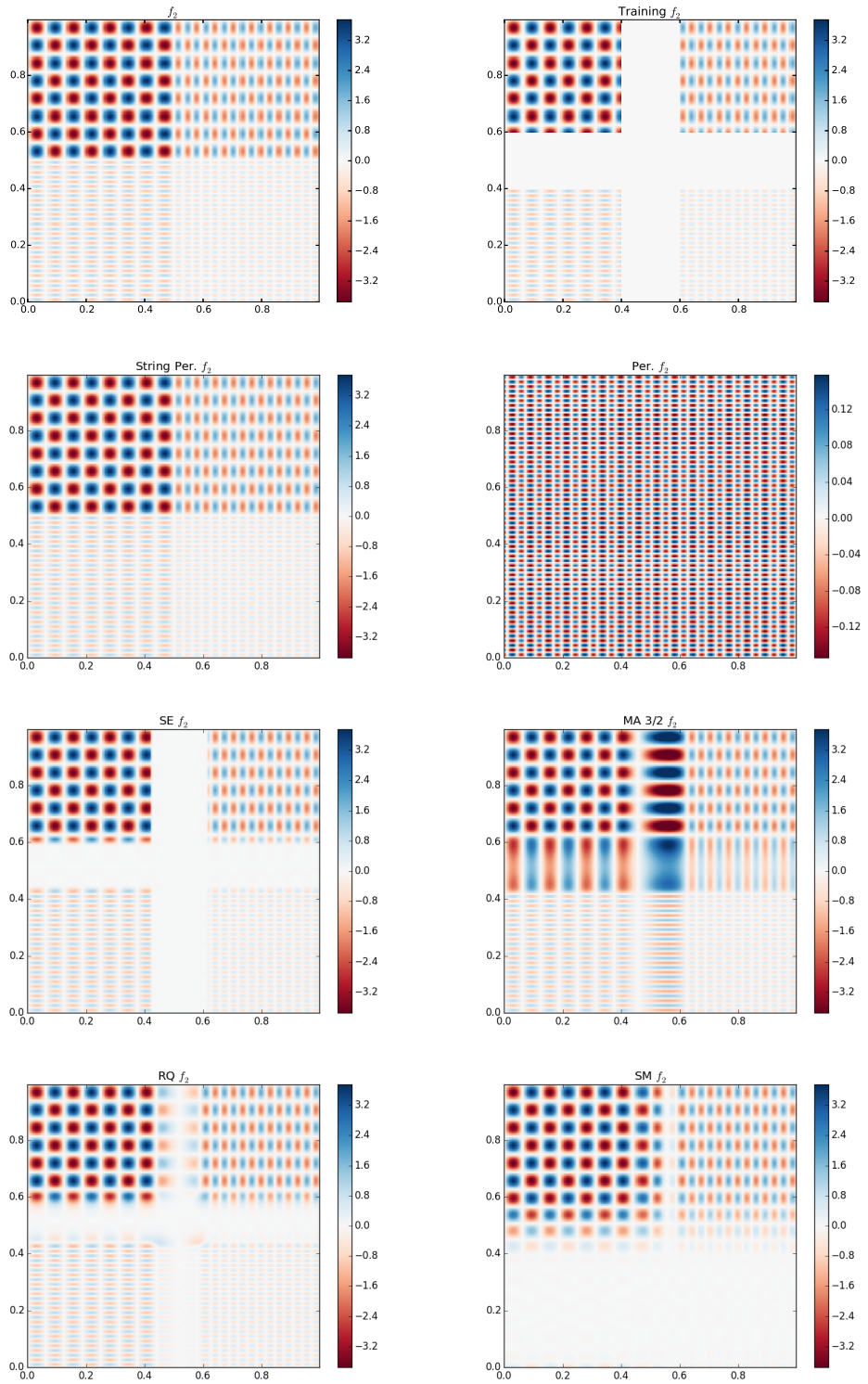

Figure 6: Extrapolation of a synthetic function f_2 (top left corner), cropped in the middle for training (top right corner), using *string GP* regression and vanilla GP regression with various popular and expressive kernels.

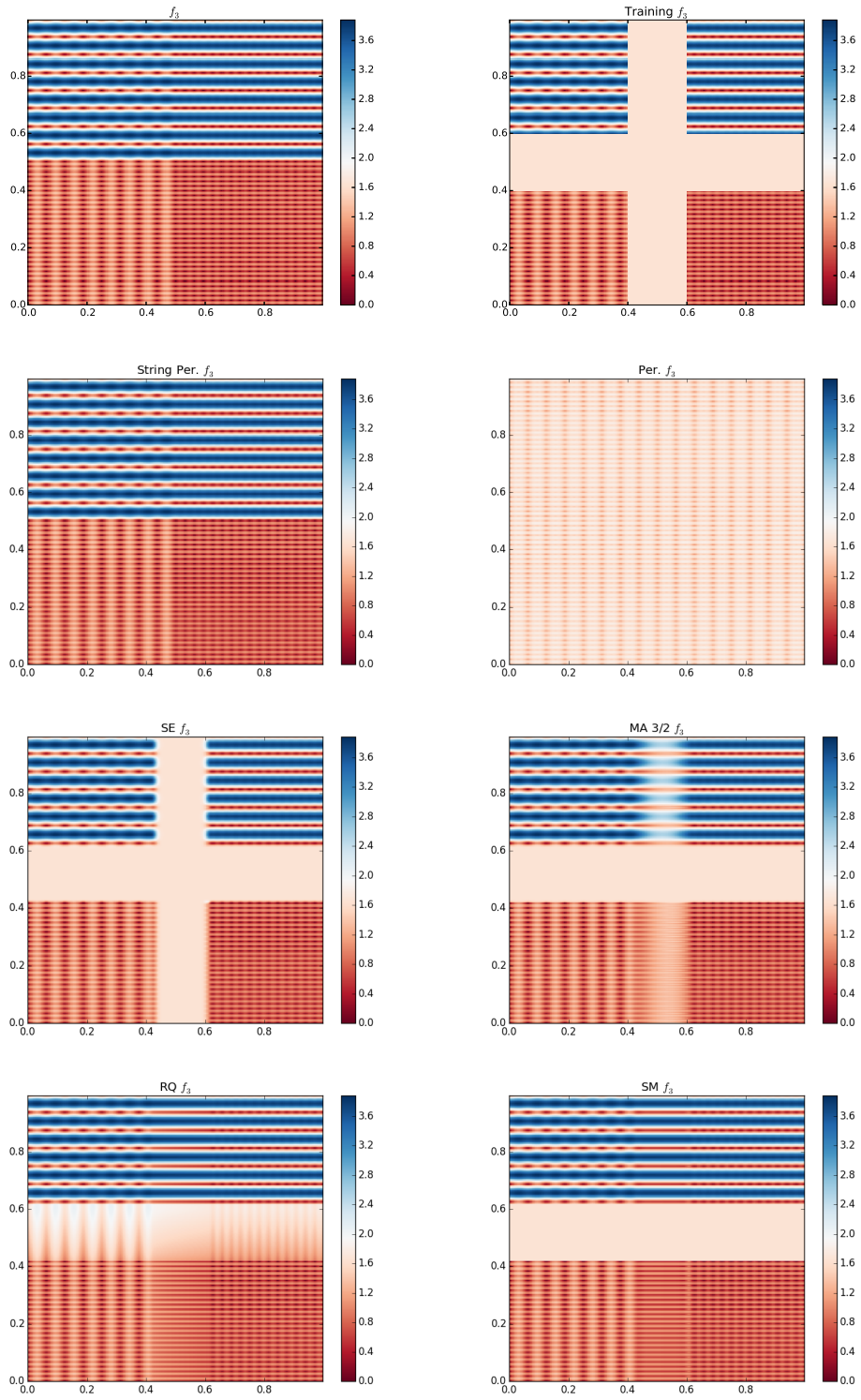

Figure 7: Extrapolation of a synthetic function f_3 (top left corner), cropped in the middle for training (top right corner), using *string GP* regression and vanilla GP regression with various popular and expressive kernels.

just the ones that fall into its input subspace. Each GP expert prediction is therefore continuous on the whole input domain,[13] and the linear weighting schemes operated by the BCM and the rBCM on expert predictions to construct the overall predictive mean preserve continuity. However, we found that the BCM and the rBCM suffer from three pitfalls. First, we found them to be less accurate than any other alternative out-of-sample on this data set (see Table 3). Second, their predictions of latent function values are overly uncertain. This might be due to the fact that, each GP expert being trained only with training samples that lie on its input subspace, its predictions about test inputs that lie farther away from its input subspace will typically be much more uncertain, so that, despite the weighting scheme of the Bayesian committee machine putting more mass on 'confident' experts, overall the posterior variance over latent function values might still be much higher than in the *standard GP paradigm* for instance. This is well illustrated by both the last column of Table 3 and the BCM and rBCM plots in Figure 9. On the contrary, no *string GP* model suffers from this excess uncertainty problem. Third, the posterior means of the BCM, the rBCM and the vanilla GP regression exhibit oscillations towards the end ($t > 40$ms) that are inconsistent with the experimental setup; the increases in acceleration as the helmet slows down suggested by these posterior means would require an additional source of energy after the bounce.

In addition to being more accurate and more certain about predictions than vanilla GP regression, the BCM and the rBCM (see Table 3), *string GP* regressions yield posterior mean acceleration profiles that are more consistent with the physics of the experiment: steady speed prior to the shock, followed by a deceleration resulting from the shock, a brief acceleration resulting from the change in direction after the bounce, and finally a smooth slow down due to the dissipation of kinetic energy. Moreover, unlike the vanilla GP regression, the rBCM and the BCM, *string GP* regressions yield smaller posterior variances towards the beginning and the end of the experiment than in the middle, which is consistent with the fact that the operator would be less uncertain about the acceleration at the beginning and at the end of the experiment—one would indeed expect the acceleration to be null at the beginning and at the end of the experiment. This desirable property can be attributed to the heteroskedasticity of the noise structure in the *string GP* regression model.

We also learned the derivative of the latent acceleration with respect to time, purely from noisy acceleration measurements using the joint law of a *string GP* and its derivative (Theorem 2). This is illustrated in Figure 8.

13. So long as the functional prior is continuous, which is the case here.

	Training	Prediction			
	Log. lik.	Log. lik.	Absolute Error	Squared Error	Pred. Std
String GP (4 strings)	-388.36 ± 0.36	-22.16 ± 0.41	$\mathbf{15.70 \pm 1.05}$	$\mathbf{466.47 \pm 50.74}$	$0.70/2.25/3.39$
String GP (6 strings)	$\mathbf{-367.21 \pm 0.43}$	-21.99 ± 0.37	15.89 ± 1.06	475.59 ± 51.95	$\mathbf{0.64/2.21/3.46}$
Vanilla GP	-420.69 ± 0.24	-22.77 ± 0.24	16.84 ± 1.09	524.18 ± 58.33	$2.66/3.09/4.94$
Mix. of 4 GPs	-388.37 ± 0.38	-20.90 ± 0.38	16.61 ± 1.10	512.30 ± 56.08	$1.67/2.85/4.59$
Mix. of 6 GPs	-369.05 ± 0.45	$\mathbf{-20.11 \pm 0.45}$	16.05 ± 1.11	500.43 ± 58.26	$0.62/2.83/4.63$
BCM with 4 GPs	-419.08 ± 0.30	-22.94 ± 0.26	17.17 ± 1.13	538.94 ± 61.91	$7.20/9.92/22.92$
BCM with 6 GPs	-422.15 ± 0.30	-22.91 ± 0.26	16.93 ± 1.12	533.21 ± 61.78	$7.09/9.93/25.10$
rBCM with 4 GPs	-419.08 ± 0.30	-22.99 ± 0.27	17.29 ± 1.11	546.95 ± 61.21	$5.86/9.08/27.52$
rBCM with 6 GPs	-422.15 ± 0.30	-22.96 ± 0.28	16.79 ± 1.12	542.95 ± 61.95	$5.15/8.61/29.15$

Table 3: Performance comparison between *string GPs*, vanilla GPs, mixture of independent GPs, the Bayesian committee machine (Tresp (2000)) and the robust Bayesian committee machine (Deisenroth and Ng (2015)) on the motorcycle data set of Silverman (1985). The Matérn 3/2 kernel was used throughout. The domain partitions were learned in the *string GP* experiments by maximum likelihood. The learned partitions were then reused to allocate data between GP experts in other models. 50 random runs were performed, each run leaving 5 data points out for testing and using the rest for training. All results (except for predictive standard deviations) are reported as average over the 50 runs $\pm$ standard error. The last column contains the minimum, average and maximum of the predictive standard deviation of the values of the latent (noise-free) function at all test points across random runs.

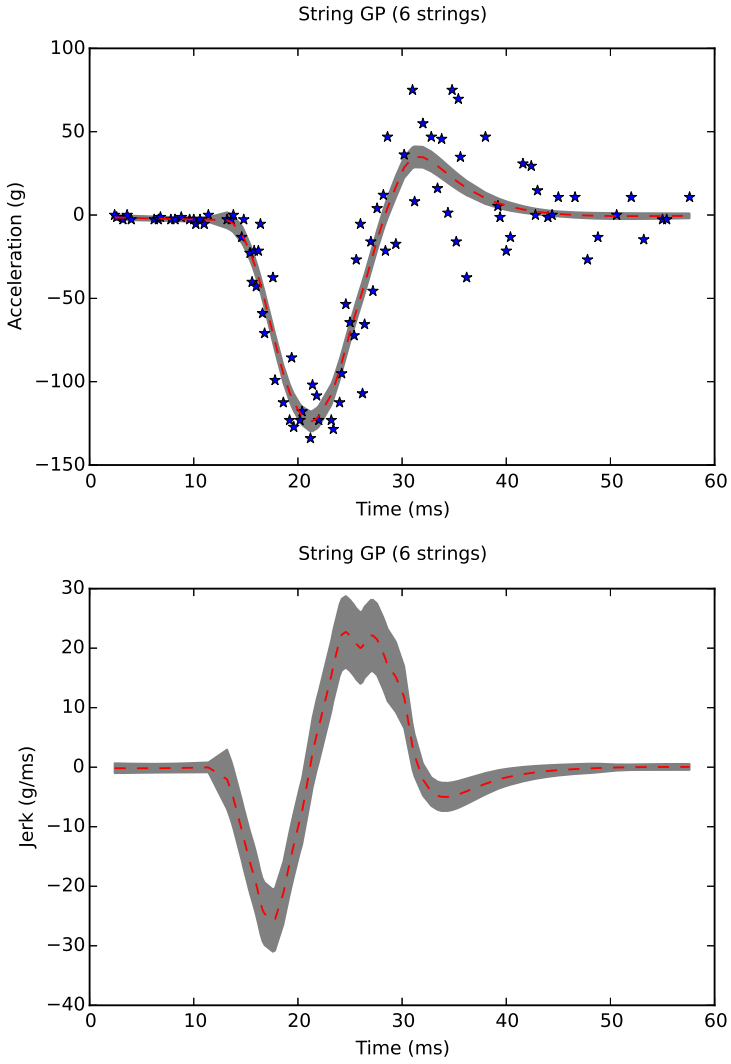

Figure 8: Posterior mean $\pm$ 2 predictive standard deviations on the motorcycle data set (see Silverman, 1985), under a Matérn 3/2 derivative *string GP* prior with 6 learned strings. The top figure shows the noisy accelerations measurements and the learned latent function. The bottom function illustrates the derivative of the acceleration with respect to time learned from noisy acceleration samples. Posterior credible bands are over the latent functions rather than noisy measurements, and as such they do not include the measurement noise.

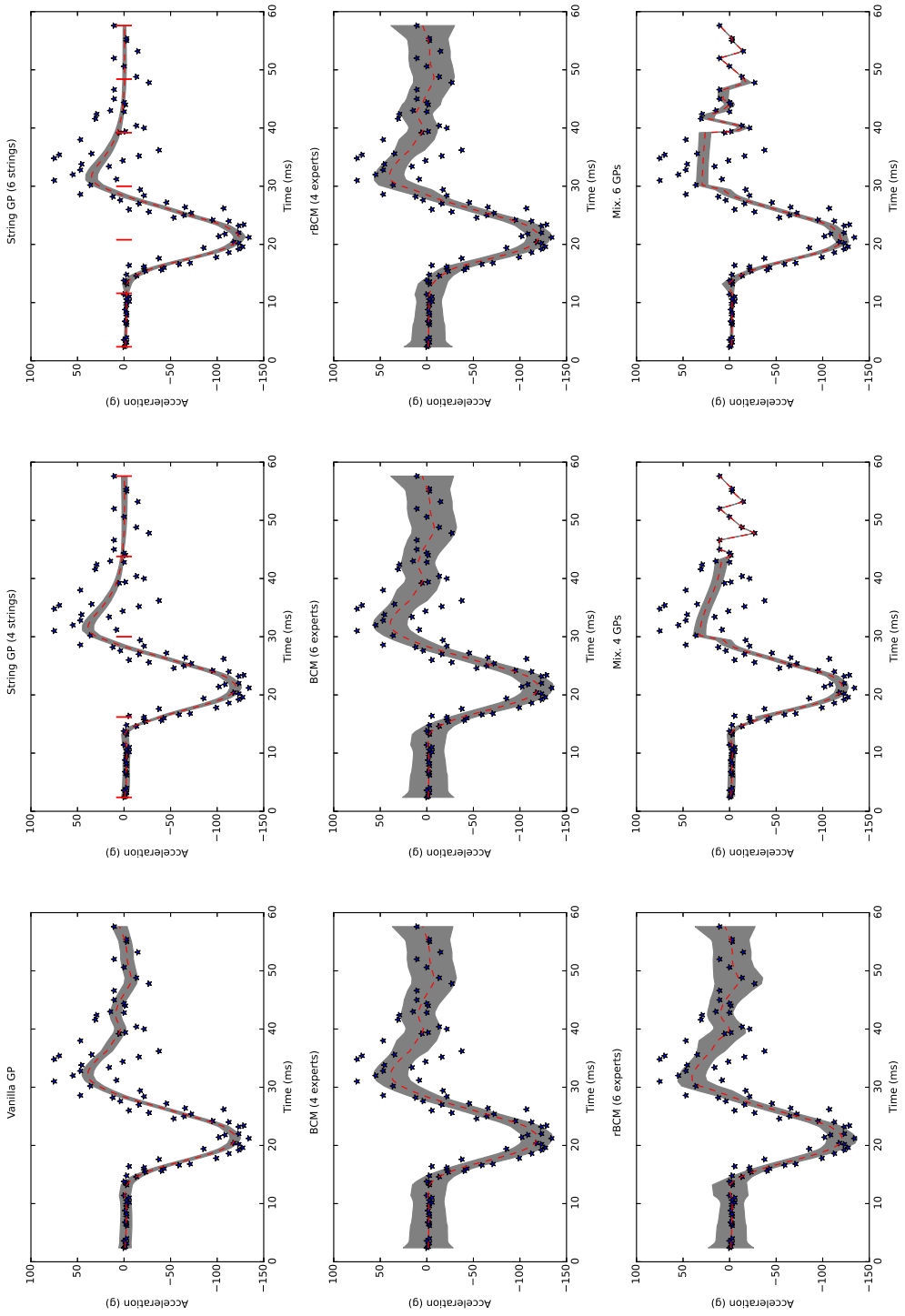

Figure 9: Bayesian nonparametric regressions on the motorcycle data set of Silverman (1985). Models compared are *string GP* regression, vanilla GP regression, mixture of independent GP regression experts on a partition of the domain, the Bayesian committee machine (BCM) and the robust Bayesian committee machine (rBCM). Domain partitions were learned during *string GP* maximum likelihood inference (red vertical bars), and reused in other experiments. Blue stars are noisy samples, red lines are posterior means of the latent function and grey bands correspond to ± 2 predictive standard deviations of the (noise-free) latent function about its posterior mean.

6.3 Large Scale Regression

To illustrate how our approach fares against competing alternatives on a standard large scale problem, we consider predicting arrival delays of commercial flights in the USA in 2008 as studied by Hensman et al. (2013). We choose the same covariates as in Hensman et al. (2013), namely the age of the aircraft (number of years since deployment), distance that needs to be covered, airtime, departure time, arrival time, day of the week, day of the month and month. Unlike Hensman et al. (2013) who only considered commercial flights between January 2008 and April 2008, we consider commercial throughout the whole year, for a total of 5.93 million records. In addition to the whole data set, we also consider subsets so as to empirically illustrate the sensitivity of computational time to the number of samples. Selected subsets consist of $10,000$, $100,000$ and $1,000,000$ records selected uniformly at random. For each data set, we use $2/3$ of the records selected uniformly at random for training and we use the remaining $1/3$ for testing. In order to level the playing field between stationary and nonstationary approaches, we normalize training and testing data sets.[14] As competing alternatives to *string GPs* we consider the SVIGP of Hensman et al. (2013), the Bayesian committee machines (BCM) of Tresp (2000), and the robust Bayesian committee machines (rBCM) of Deisenroth and Ng (2015).

As previously discussed the prediction scheme operated by the BCM is Kolmogorov-inconsistent in that the resulting predictive distributions are not consistent by marginalization.[15] Moreover, jointly predicting all function values by using the set of all test inputs as query set, as originally suggested in Tresp (2000), would be impractical in this experiment given that the BCM requires inverting a covariance matrix of the size of the query set which, considering the numbers of test inputs in this experiment (which we recall can be as high as 1.97 million), would be computationally intractable. To circumvent this problem we use the BCM algorithm to query one test input at a time. This approach is in-line with that adopted by Deisenroth and Ng (2015), where the authors did not address determining joint predictive distributions over multiple latent function values. For the BCM and rBCM, the number of experts is chosen so that each expert processes 200 training points. For SVIGP we use the implementation made available by the The GPy authors (2012–2016), and we use the same configuration as in Hensman et al. (2013). As for *string GPs*, we use the symmetric sum as link function, and we run two types of experiments, one allowing for inference of change-points (String GP), and the other enforcing a single kernel configuration per input dimension (String GP*). The parameters α and β are chosen so that the prior mean number of change-points in each input dimension is 5% of the number of distinct training and testing values in that input dimension, and so that the prior variance of the foregoing number of change-points is 50 times the prior mean—the aim is to be uninformative about the number of change-points. We run $10,000$ iterations of our RJ-MCMC sampler and discarded the first $5,000$ as 'burn-in'. After burn-in we record the states of the Markov chains for analysis using a 1-in-100 thinning rate. Predictive accuracies are reported in Table 4 and CPU time requirements[16] are illustrated in Figure 10. We stress that all experiments were run on a multi-core machine, and that we prefer using the cumulative CPU clock resource

14. More precisely, we substract from every feature sample (both in-sample and out-of-sample) the in-sample mean of the feature and we divide the result by the in-sample standard deviation of the feature.
15. For instance the predictive distribution of the value of the latent function at a test input x_1, namely $f(x_1)$, obtained by using $\{x_1\}$ as set of test inputs in the BCM, differs from the predictive distribution obtained by using $\{x_1, x_2\}$ as set of test inputs in the BCM and then marginalising with respect to the second input x_2.
16. We define CPU time as the cumulative CPU clock resource usage of the whole experiment (training and testing), across child processes and threads, and across CPU cores.

as time complexity metric, instead of wall-clock time, so as to be agnostic to the number of CPU cores used in the experiments. This metric has the merit of illustrating how the number of CPU cores required grows as a function of the number of training samples for a fixed/desired wall-clock execution time, but also how the wall-clock execution time grows as a function of the number of training samples for a given number of available CPU cores.

The BCM and the rBCM perform the worst in this experiment both in terms of predictive accuracy (Table 4) and total CPU time (Figure 10). The poor scalability of the BCM and the rBCM is primarily due to the testing phase. Indeed, if we denote M the total number of experts, then $M = \lceil \frac{N}{300} \rceil$, as each expert processes 200 training points, of which there are $\frac{2}{3}N$. In the prediction phase, each expert is required to make predictions about all $\frac{1}{3}N$ test inputs, which requires evaluating M products of an $\frac{1}{3}N \times 200$ matrix with a 200×200 matrix, which results in a total CPU time requirement that grows in $\mathcal{O}(M\frac{1}{3}N200^2)$, which is the same as $\mathcal{O}(N^2)$. Given that training CPU time grows linearly in N the cumulative training and testing CPU time grows quadratically in N. This is well illustrated in Figure 10, where it can be seen that the slopes of total CPU time profiles of the BCM and the rBCM in log-log scale are approximately 2. The airline delays data set was also considered by Deisenroth and Ng (2015), but the authors restricted themselves to a fixed size of the test set of $100,000$ points. However, this limitation might be restrictive as in many 'smoothing' applications, the test data set can be as large as the training data set—neither the BCM nor the rBCM would be sufficiently scalable in such applications.

As for SVIGP, although it was slightly more accurate than *string GPs* on this data set, it can be noted from Figure 10 that *string GPs* required 10 times less CPU resources. In fact we were unable to run the experiment on the full data set with SVIGP—we gave up after 500 CPU hours, or more than a couple of weeks wall-clock time given that the GPy implementation of SVIGP makes little use of multiple cores. As a comparison, the full experiment took 91.0 hours total CPU time (≈ 15 hours wall-clock time on our 8 cores machine) when change-points were inferred and 83.11 hours total CPU time (≈ 14 hours wall-clock time on our 8 cores machine) when change-points were not inferred. Another advantage of additively separable *string GPs* over GPs, and subsequently over SVIGP, is that they are more interpretable. Indeed, one can determine at a glance from the learned posterior mean *string GPs* of Figure 11 the effect of each of the 8 covariates considered on arrival delays. It turns out that the three most informative factors in predicting arrival delays are departure time, distance and arrival time, while the age of the aircraft, the day of the week and the day of the month seem to have little to no effect. Finally, posterior distributions of the number of change-points are illustrated in Figure 12, and posterior distributions of the locations of change-points are illustrated in Figure 13.

N	String GP	String GP*	BCM	rBCM	SVIGP
$10,000$	1.03 ± 0.10	1.06 ± 0.10	1.06 ± 0.10	1.06 ± 0.10	$\mathbf{0.90 \pm 0.09}$
$100,000$	0.93 ± 0.03	0.96 ± 0.03	1.66 ± 0.03	1.04 ± 0.04	$\mathbf{0.88 \pm 0.03}$
$1,000,000$	0.93 ± 0.01	0.92 ± 0.01	N/A	N/A	$\mathbf{0.82 \pm 0.01}$
$5,929,413$	0.90 ± 0.01	0.93 ± 0.01	N/A	N/A	N/A

Table 4: Predictive mean squared errors (MSEs) $\pm$ one standard error on the airline arrival delays experiment. Squared errors are expressed as fraction of the sample variance of airline arrival delays, and hence are unitless. With this normalisation, a MSE of 1.00 is as good as using the training mean arrival delays as predictor. The * in String GP* indicates that inference was performed without allowing for change-points. N/A entries correspond to experiments that were not over after 500 CPU hours.

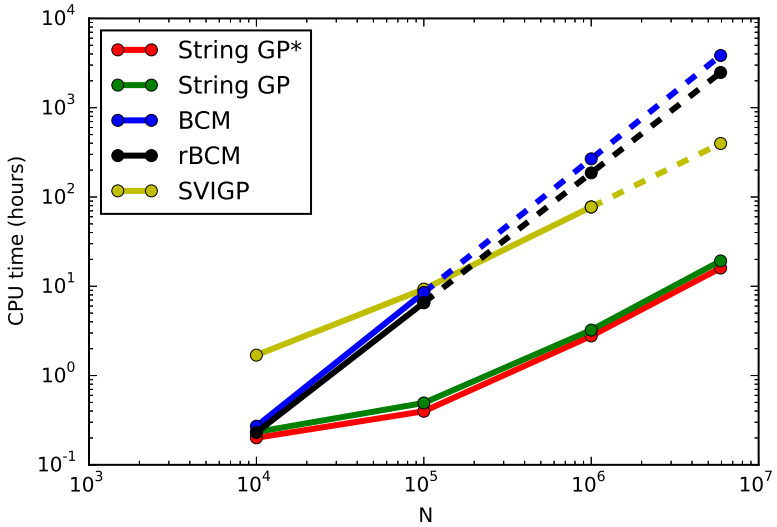

Figure 10: Total CPU time (training and testing) taken by various regression approaches on the airline delays data set as a function of the size of the subset considered, in log-log scale. The experimental setup is described in Section 6.3. The CPU time reflects actual CPU clock resource usage in each experiment, and is therefore agnostic to the number of CPU cores used. It can be regarded as the wall-clock time the experiment would have taken to complete on a single-core computer (with the same CPU frequency). Dashed lines are extrapolated values, and correspond to experiments that did not complete after 500 hours of CPU time.

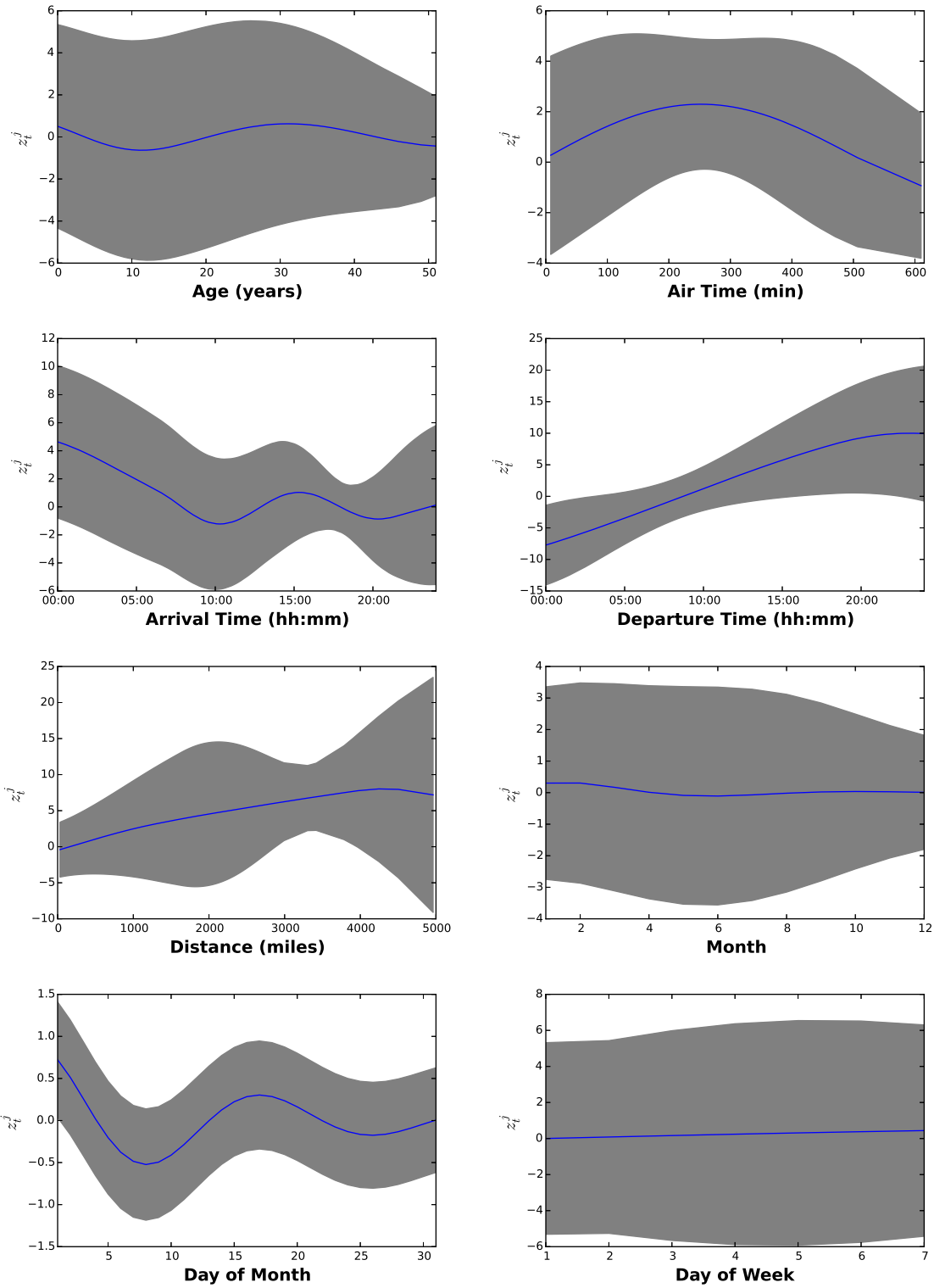

Figure 11: Posterior mean $\pm$ one posterior standard deviation of univariate *string GPs* in the airline delays experiment of Section 6.3. Change-points were automatically inferred in this experiment.

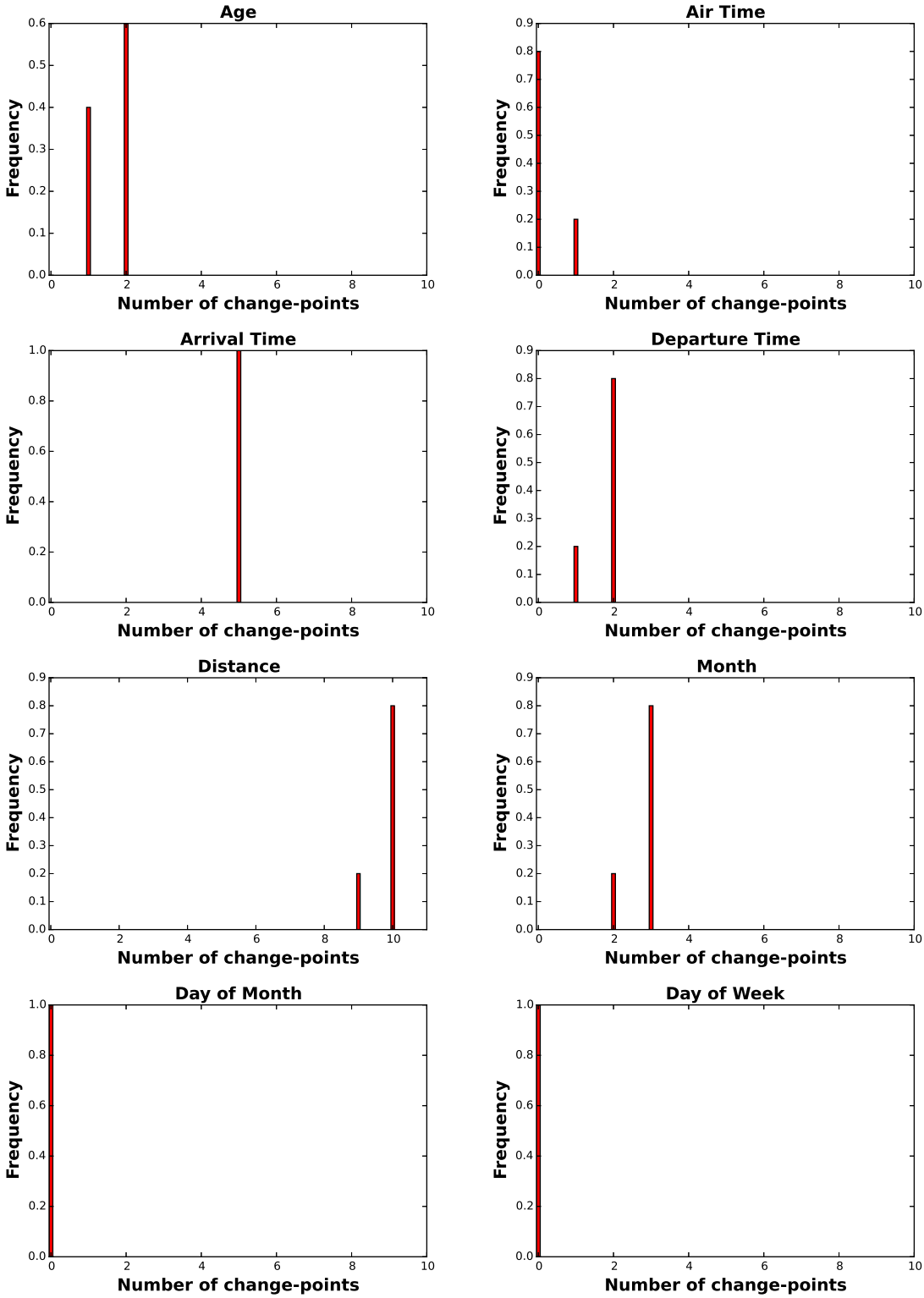

Figure 12: Posterior distributions of the numbers of change-points in each input dimension in the airline delays experiment of Section 6.3.

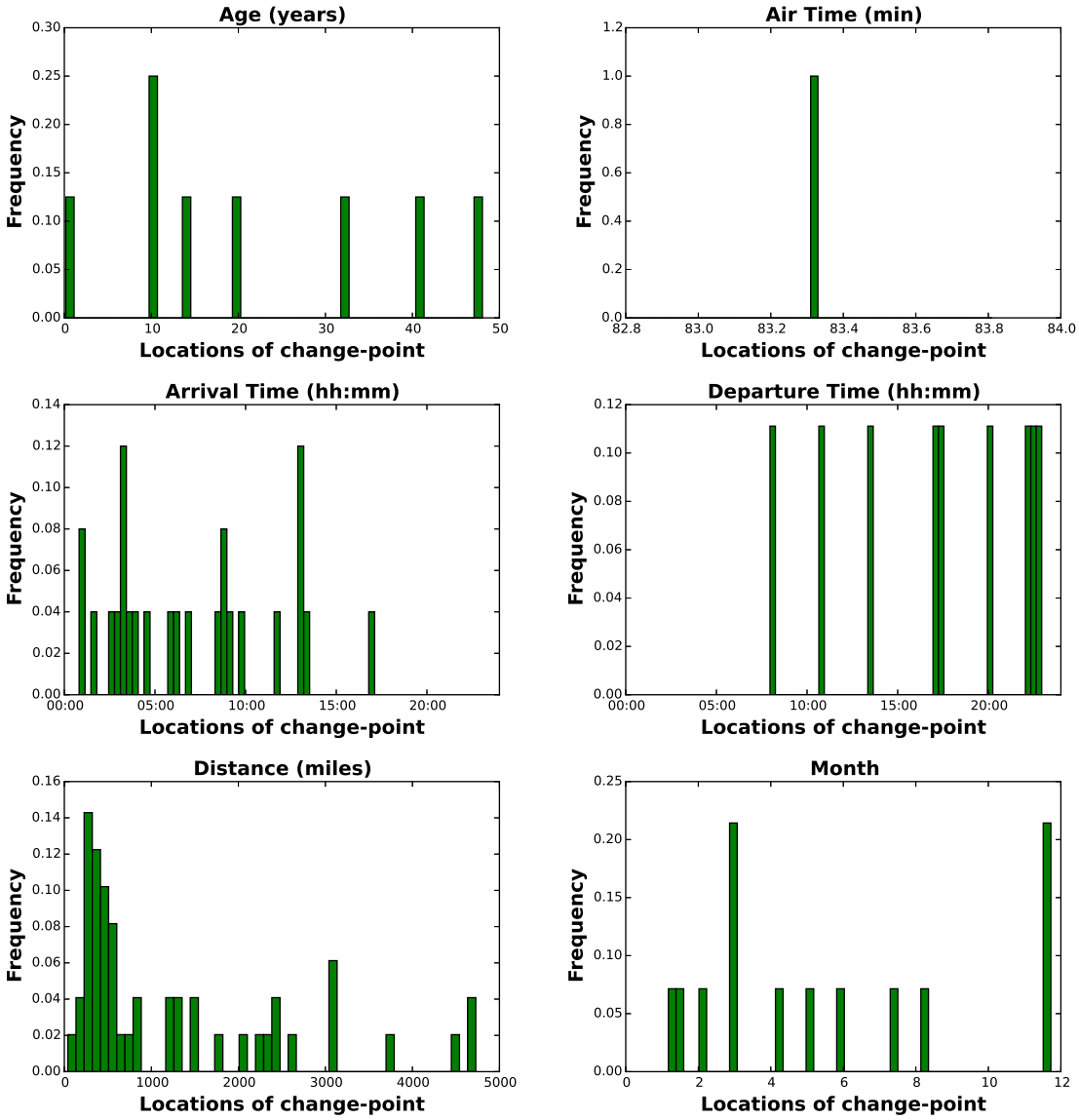

Figure 13: Posterior distributions of the locations of change-points in each input dimension in the airline delays experiment of Section 6.3. Dimensions that were learned to exhibit no change-point have been omitted here.

6.4 Large Scale Dynamic Asset Allocation

An important feature of our proposed RJ-MCMC sampler (Algorithm 2) is that, unlike the BCM, the rBCM and SVIGP, which are restricted to Bayesian nonparametric regression and classification, Algorithm 2 is agnostic with regard to the likelihood model, so long as it takes the form $p(\mathcal{D}|\mathbf{f},\mathbf{u})$. Thus, it may be used as is on a wide variety of problems that go beyond classification and regression. In this experiment we aim to illustrate the efficacy of our approach on one such large scale problem in quantitative finance.

6.4.1 BACKGROUND

Let $(x_i(t))_{t>0}$ for $i=1,\ldots,n$ be n stock price processes. Let $(X_i(t))_{t>0}$ for $i=1,\ldots,n$ denote the *market capitalisation* processes, that is $X_i(t)=n_i(t)x_i(t)$ where $n_i(t)$ is the number of shares in company i trading in the market at time t. We call *long-only portfolio* any vector-valued stochastic process $\pi=(\pi_1,\ldots,\pi_n)$ taking value on the unit simplex on $\mathbb{R}^n$, that is

$$\forall i,t,\ \pi_i(t)\geq 0 \quad \text{and} \quad \sum_{i=1}^{n}\pi_i(t)=1.$$

Each process π_i represents the proportion of an investor's wealth invested in (holding) shares in asset i. An example long-only portfolio is the market portfolio $\mu=(\mu_1,\ldots,\mu_n)$ where

$$\mu_i(t)=\frac{X_i(t)}{X_1(t)+\cdots+X_n(t)} \tag{47}$$

is the market weight of company i at time t, that is its size relative to the total market size (or that of the universe of stocks considered). The market portfolio is very important to practitioners as it is often perceived not be to subject to idiosyncracies, but only to systemic risk. It is often used as an indicator of how the stock market (or a specific universe of stocks) performs as a whole. We denote Z_π the value process of a portfolio π with initial capital $Z_\pi(0)$. That is, $Z_\pi(t)$ is the wealth at time t of an investor who had an initial wealth of $Z_\pi(0)$, and dynamically re-allocated all his wealth between the n stocks in our universe up to time t following the continuous-time strategy π.

A mathematical theory has recently emerged, namely *stochastic portfolio theory* (SPT) (see Karatzas and Fernholz, 2009) that studies the stochastic properties of the wealth processes of certain portfolios called *functionally-generated portfolio* under realistic assumptions on the market capitalisation processes $(X_i(t))_{t>0}$. Functionally-generated portfolios are rather specific in that the allocation at time t, namely $(\pi_1(t),\ldots,\pi_n(t))$, solely depends on the market weights vector $(\mu_1(t),\ldots,\mu_n(t))$. Nonetheless, some functionally-generated portfolios π^* have been found that, under the mild (so-called diversity) condition

$$\exists \mu_{\max}, 0<\mu_{\max}<1 \quad \text{s.t.} \quad \forall i\leq n,\ t\leq T, \quad \mu_i(t)\leq \mu_{\max}, \tag{48}$$

outperform the market portfolio over the time horizon $[0,T]$ with probability one (see Vervuurt and Karatzas, 2015; Karatzas and Fernholz, 2009). More precisely,

$$\mathbb{P}\left(Z_{\pi^*}(T)\geq Z_\mu(T)\right)=1 \quad \text{and} \quad \mathbb{P}\left(Z_{\pi^*}(T)>Z_\mu(T)\right)>0. \tag{49}$$

Galvanized by this result, we here consider the inverse problem consisting of learning from historical market data a portfolio whose wealth process has desirable user-specified properties. This

inverse problem is perhaps more akin to the problems faced by investment professionals: i) their benchmarks depend on the investment vehicles pitched to investors and may vary from one vehicle to another, ii) they have to take into account liquidity costs, and iii) they often find it more valuable to go beyond market weights and leverage multiple company characteristics in their investment strategies.

6.4.2 MODEL CONSTRUCTION

We consider portfolios $\pi^f = \left(\pi_1^f, \dots, \pi_n^f\right)$ of the form

$$\pi_i^f(t) = \frac{f(c_i(t))}{f(c_1(t)) + \cdots + f(c_n(t))}, \tag{50}$$

where $c_i(t) \in \mathbb{R}^d$ are some quantifiable characteristics of asset i that may be observed in the market at time t, and f is a positive-valued function. Portfolios of this form include all functionally-generated portfolios studied in SPT as a special case.[17] A crucial departure of our approach from the aforementioned type of portfolios is that the market characteristics processes c_i need not be restricted to size-based information, and may contain additional information such as social media sentiments, stock price path-properties, but also characteristics relative to other stocks such as performance relative to the best/worst performing stock last week/month/year etc. We place a mean-zero *string GP* prior on $\log f$. Given some historical data $\mathcal{D}$ corresponding to a training time horizon $[0, T]$, the likelihood model $p\left(\mathcal{D}|\pi^f\right)$ is defined by the investment professional and reflects the extent to which applying the investment strategy π^f over the training time horizon would have achieved a specific investment objective. An example investment objective is to achieve a high excess return relative to a benchmark portfolio α

$$\mathcal{U}_{\text{ER}}\left(\pi^f\right) = \log Z_{\pi^f}(T) - \log Z_\alpha(T). \tag{51}$$

α can be the market portfolio (as in SPT) or any stock index. Other risk-adjusted investment objectives may also be used. One such objective is to achieve a high Sharpe-ratio, defined as

$$\mathcal{U}_{\text{SR}}\left(\pi^f\right) = \frac{\bar{r}\sqrt{252}}{\sqrt{\frac{1}{T}\sum_{t=1}^{T}(r(t) - \bar{r})^2}}, \tag{52}$$

where the time t is in days, $r(t) := \log Z_{\pi^f}(t) - \log Z_{\pi^f}(t-1)$ are the daily returns the portfolio π^f and $\bar{r} = \frac{1}{T}\sum_{t=1}^{T} r(t)$ its average daily return. More generally, denoting $\mathcal{U}\left(\pi^f\right)$ the performance of the portfolio π^f over the training horizon $[0, T]$ (as per the user-defined investment objective), we may choose as likelihood model a distribution over $\mathcal{U}\left(\pi^f\right)$ that reflects what the investment professional considers good and bad performance. For instance, in the case of the excess return relative to a benchmark portfolio or the Sharpe ratio, we may choose $\mathcal{U}\left(\pi^f\right)$ to be supported on $]0, +\infty[$ (for instance $\mathcal{U}\left(\pi^f\right)$ can be chosen to be Gamma distributed) so as to express that portfolios that do not outperform the benchmark or loose money overall in the training data are not of interest. We may then choose the mean and standard deviation of the Gamma distribution based on our

17. We refer the reader to Karatzas and Fernholz (2009) for the definition of functionally-generated portfolios.

expectation as to what performance a good candidate portfolio can achieve, and how confident we feel about this expectation. Overall we have,

$$p\left(\mathcal{D}|\pi^f\right) = \gamma\left(\mathcal{U}\left(\pi^f\right); \alpha_e, \beta_e\right), \tag{53}$$

where $\gamma(.;\alpha,\beta)$ is the probability density function of the Gamma distribution. Noting, from Equation (50) that $\pi^f(t)$ only depends on f through its values at $(c_1(t),\ldots,c_n(t))$, and assuming that $\mathcal{U}\left(\pi^f\right)$ only depends on π^f evaluated at a finite number of times (as it is the case for excess returns and the Sharpe ratio), it follows that $\mathcal{U}(\pi^f)$ only depends on $\mathbf{f}$, a vector of values of f at a finite number of points. Hence the likelihood model, which we may rewrite as

$$p(\mathcal{D}|\mathbf{f}) = \gamma\left(\mathcal{U}\left(\pi_i^f\right); \alpha_e, \beta_e\right), \tag{54}$$

is of the form required by the RJ-MCMC sampler previously developed. By sampling from the posterior distribution $p(\mathbf{f}, \mathbf{f}^*, \nabla\mathbf{f}, \nabla\mathbf{f}^*|\mathcal{D})$, the hope is to learn a portfolio that did well during the training horizon, to analyse the sensitivity of its investment strategy to the underlying market characteristics through the gradient of f, and to evaluate the learned investment policy on future market conditions.

6.4.3 EXPERIMENTAL SETUP

The universe of stocks we considered for this experiment are the constituents of the S&P 500 index, accounting for changes in constituents over time and corporate events. We used the period 1st January 1990 to 31st December 2004 for training and we tested the learned portfolio during the period 1st January 2005 to 31st December 2014. We rebalanced the portfolio daily, for a total of 2.52 million input points at which the latent function f must be learned. We considered as market characteristics the market weight (CAP), the latest return on asset (ROA) defined as the ratio between the net yearly income and the total assets as per the latest balance sheet of the company known at the time of investment, the previous close-to-close return (PR), the close-to-close return before the previous (PR2), and the S&P long and short term credit rating (LCR and SCR). While the market weight is a company size characteristic, the ROA reflects how well a company performs relative to its size, and we hope that S&P credit ratings will help further discriminate successful companies from others. The close-to-close returns are used to learn possible 'momentum' patterns from the data. The data originate from the CRSP and Compustat databases. In the experiments we considered as performance metric the annualised excess return $\mathcal{U}_{\text{ER-EWP}}$ relative to the equally-weighted portfolio. We found the equally-weighted portfolio to be a harder benchmark to outperform than the market portfolio. We chose α_e and β_e in Equation (54) so that the mean of the Gamma distribution is 10.0 and its variance 0.5, which expresses a very greedy investment target.

It is worth pointing out that none of the scalable GP alternatives previously considered can cope with our likelihood model Equation (54). We compared the performance of the learned *string GP* portfolio out-of-sample to those of the best three SPT portfolios studied in Vervuurt and Karatzas (2015), namely the equally weighted portfolio

$$\pi_i^{\text{EWP}}(t) = \frac{1}{n}, \tag{55}$$

and the diversity weighted portfolios

$$\pi_i^{\text{DWP}}(t;p) = \frac{\mu_i(t)^p}{\mu_1(t)^p + \cdots + \mu_n(t)^p}, \tag{56}$$

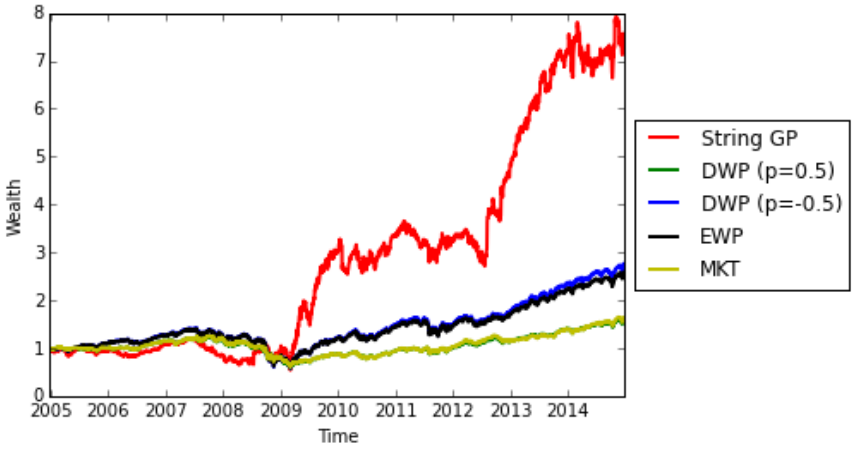

Figure 14: Evolution of the wealth processes of various long-only trading strategies on the S&P 500 universe of stocks between 1st January 2005 (where we assume a starting wealth of 1) and 31st December 2014. The String GP strategy was learned using market data from 1st January 1990 to 31st December 2004 as descried in Section 6.4. EWP refers to the equally-weighted portfolio, MKT refers to the market portfolio (which weights stocks proportionally to their market capitalisations) and DWP (p) refers to the diversity-weighted portfolio with exponent p (which weights stocks proportionally to the p-th power of their market weights).

with parameter p equals to -0.5 and 0.5, and the market portfolio. Results are provided Table 5, and Figure 14 displays the evolution of the wealth process of each strategy. It can be seen that the learned *string GP* strategy considerably outperforms the next best SPT portfolio. This experiment not only demonstrates (once more) that *string GPs* scale to large scale problems, it also illustrates that our inference scheme is able to unlock commercial value in new intricate large scale applications where no alternative is readily available. In effect, this application was first introduced by Kom Samo and Vervuurt (2016), where the authors used a Gaussian process prior on a Cartesian grid and under a separable covariance function so as to speed up inference with Kronecker techniques. Although the resulting inference scheme has time complexity that is linear in the total number of points on the grid, for a given grid resolution, the time complexity grows *exponentially* in the dimension of the input space (that is the number of trading characteristics), which is impractical for $d \geq 4$. On the other hand, *string GPs* allow for a time complexity that grows linearly with the number of trading characteristics, thereby enabling the learning of subtler market inefficiencies from the data.

Strategy	Sharpe Ratio	$Z_\pi(T)/Z_{\text{EWP}}(T)$	Avg. Ann. Ret.
String GP	**0.73**	**2.87**	**22.07%**
DWP ($p = -0.5$)	0.55	1.07	10.56%
EWP	0.53	1.00	9.84%
MKT	0.34	0.62	4.77%
DWP ($p = 0.5$)	0.33	0.61	4.51%

Table 5: Performance of various long-only trading strategies on the S&P 500 universe of stocks between 1st January 2005 (where we assume a starting wealth of 1) and 31st December 2014. The String GP strategy was learned using market data from 1st January 1990 to 31st December 2004 as descried in Section 6.4. EWP refers to the equally-weighted portfolio, MKT refers to the market portfolio (which weights stocks proportionally to their market capitalisations) and DWP (p) refers to the diversity-weighted portfolio with exponent p (which weights stocks proportionally to the p-th power of the market weight of the asset). $Z_\pi(T)$ denotes the terminal wealth of strategy π, and Avg. Ann. Ret. is the strategy's equivalent constant annual return over the test horizon.

7. Discussion

In this paper, we introduce a novel class of smooth functional priors (or stochastic processes), which we refer to as *string GPs*, with the aim of simultaneously addressing the lack of scalability and the lack of flexibility of Bayesian kernel methods. Unlike existing approaches, such as Gaussian process priors (Rasmussen and Williams (2006)) or student-t process priors (Shah et al. (2014)), which are parametrised by *global* mean and covariance functions, and which postulate *fully dependent* finite-dimensional marginals, the alternative construction we propose adopts a *local* perspective and the resulting finite-dimensional marginals exhibit *conditional independence* structures. Our local approach to constructing *string GPs* provides a principled way of postulating that the latent function we wish to learn might exhibit locally homogeneous patterns, while the conditional independence structures constitute the core ingredient needed for developing scalable inference methods. Moreover, we provide theoretical results relating our approach to Gaussian processes, and we illustrate that our approach can often be regarded as a more scalable and/or more flexible extension. We argue and empirically illustrate that *string GPs* present an unparalleled opportunity for learning local patterns in small scale regression problems using nothing but standard Gaussian process regression techniques. More importantly, we propose a novel *scalable* RJ-MCMC inference scheme to learn latent functions in a wide variety of machine learning tasks, while simultaneously determining *whether* the data set exhibits local patterns, *how many* types of local patterns the data might exhibit, and *where* do changes in these patterns are likely to occur. The proposed scheme has time complexity and memory requirement that are both *linear* in the sample size N. When the number of available computing cores is at least equal to the dimension d of the input space, the time complexity is independent from the dimension of the input space. Else, the time complexity grows in $\mathcal{O}(dN)$. The memory requirement grows in $\mathcal{O}(dN)$. We empirically illustrate that our approach scales considerably better than competing alternatives on a standard benchmark data set, and is able to process data sizes that competing approaches cannot handle in a reasonable time.

7.1 Limitations

The main limitation of our approach is that, unlike the *standard GP paradigm* in which the time complexity of marginal likelihood evaluation does not depend on the dimension of the input space (other than through the evaluation of the Gram matrix), the *string GP paradigm* requires a number of computing cores that increases linearly with the dimension of the input space, or alternatively has a time complexity linear in the input space dimension on single-core machines. This is a by-product of the fact that in the *string GP paradigm*, we jointly infer the latent function and its gradient. If the gradient of the latent function is inferred in the *standard GP paradigm*, the resulting complexity will also be linear in the input dimension. That being said, overall our RJ-MCMC inference scheme will typically scale better per iteration to large input dimensions than gradient-based marginal likelihood inference in the *standard GP paradigm*, as the latter typically requires numerically evaluating an Hessian matrix, which requires computing the marginal likelihood a number of times per iterative update that grows quadratically with the input dimension. In contrast, a Gibbs cycle in our MCMC sampler has worst case time complexity that is linear in the input dimension.

7.2 Extensions

Some of the assumptions we have made in the construction of *string GPs* and *membrane GPs* can be relaxed, which we consider in detail below.

7.2.1 Stronger Global Regularity

We could have imposed more (multiple continuous differentiability) or less (continuity) regularity as boundary conditions in the construction of *string GPs*. We chose continuous differentiability as it is a relatively mild condition guaranteed by most popular kernels, and yet the corresponding treatment can be easily generalised to other regularity requirements. It is also possible to allow for discontinuity at a boundary time a_k by replacing μ_k^b and Σ_k^b in Equation (5) with ${}_kM_{a_k}$ and ${}_k\mathbf{K}_{a_k;a_k}$ respectively, or equivalently by preventing any communication between the k-th and the $(k+1)$-th strings. This would effectively be equivalent to having two independent *string GPs* on $[a_0, a_k]$ and $]a_k, a_K]$.

7.2.2 Differential Operators as Link Functions

Our framework can be further extended to allow differential operators as link functions, thereby considering the latent multivariate function to infer as the response of a differential system to independent univariate *string GP* excitations. The RJ-MCMC sampler we propose in Section 5 would still work in this framework, with the only exception that, when the differential operator is of first order, the latent multivariate function will be continuous but not differentiable, except if global regularity is upgraded as discussed above. Moreover, Proposition 5 can be generalised to first order linear differential operators.

7.2.3 DISTRIBUTED STRING GPS

The RJ-MCMC inference scheme we propose may be easily adapted to handle applications where the data set is so big that it has to be stored across multiple clusters, and inference techniques have to be developed as data flow graphs[18] (for instance using libraries such as TensorFlow).

To do so, the choice of string boundary times can be adapted so that each string has the same number of inner input coordinates, and such that in total there are as many strings across dimensions as a target number of available computing cores. We may then place a prior on kernel memberships similar to that of previous sections. Here, the change-points may be restricted to coincide with boundary times, and we may choose priors such that the sets of change-points are independent between input dimensions. In each input dimension the prior on the number of change-points can be chosen to be a truncated Poisson distribution (truncated to never exceed the total number of boundary times), and conditional on their number we may choose change-points to be uniformly distributed in the set of boundary times. In so doing, any two strings whose shared boundary time is not a change-point will be driven by the same kernel configuration.

This new setup presents no additional theoretical or practical challenges, and the RJ-MCMC techniques previously developed are easily adaptable to jointly learn change-points and function values. Unlike the case we developed in previous sections where an update of the univariate *string GP* corresponding to an input dimension, say the j-th, requires looping through all distinct j-th input coordinates, here no step in the inference scheme requires a full view of the data set in any input dimension. Full RJ-MCMC inference can be constructed as a data flow graph. An example such graph is constructed as follows. The leaves correspond to computing cores responsible for generating change-points and kernel configurations, and mapping strings to kernel configurations. The following layer is made of compute cores that use kernel configurations coming out of the previous layer to sequentially compute boundary conditions corresponding to a specific input dimension—there are d such compute cores, where d is the input dimension. These compute cores then pass computed boundary conditions to subsequent compute cores we refer to as string compute cores. Each string compute core is tasked with computing *derivative string GP* values for a specific input dimension and for a specific string in that input dimension, conditional on previously computed boundary conditions. These values are then passed to a fourth layer of compute cores, each of which being tasked with computing function and gradient values corresponding to a small subset of training inputs from previously computed *derivative string GP* values. The final layers then computes the log-likelihood using a distributed algorithm such as Map-Reduce when possible. This proposal data flow graph is illustrated Figure 15.

We note that the approaches of Kim et al. (2005), Gramacy and Lee (2008), Tresp (2000), and Deisenroth and Ng (2015) also allow for fully-distributed inference on regression problems. Distributed *string GP* RJ-MCMC inference improves on these in that it places little restriction on the type of likelihood. Moreover, unlike Kim et al. (2005) and Gramacy and Lee (2008) that yield discontinuous latent functions, *string GPs* are continuously differentiable, and unlike Tresp (2000) and Deisenroth and Ng (2015), local experts in the *string GP paradigm* (i.e. strings) are driven by possibly different sets of hyper-parameters, which facilitates the learning of local patterns.

18. A data flow graph is a computational (directed) graph whose nodes represent calculations (possibly taking place on different computing units) and directed edges correspond to data flowing between calculations or computing units.

7.2.4 APPROXIMATE MCMC FOR I.I.D. OBSERVATIONS LIKELIHOODS

As discussed in Section 5.2, the bottleneck of our proposed inference scheme is the evaluation of likelihood. When the likelihood factorises across training samples, the linear time complexity of our proposed approach can be further reduced using a Monte Carlo approximation of the log-likelihood (see for instance Bardenet et al. (2014) and references therein). Although the resulting Markov chain will typically not converge to the true posterior distribution, in practice its stationary distribution can be sufficiently close to the true posterior when reasonable Monte Carlo sample sizes are used. Convergence results of such approximations have recently been studied by Bardenet et al. (2014) and Alquier et al. (2016). We expect this extension to speed-up inference when the number of compute cores is in the order of magnitude of the input dimension, but we would recommend the previously mentioned fully-distributed string GP inference extension when compute cores are not scarce.

7.2.5 VARIATIONAL INFERENCE

It would be useful to develop suitable variational methods for inference under *string GP* priors, that we hope will scale similarly to our proposed RJ-MCMC sampler but will converge faster. We anticipate that the main challenge here will perhaps be the learning of model complexity, that is the number of distinct kernel configurations in each input dimension.

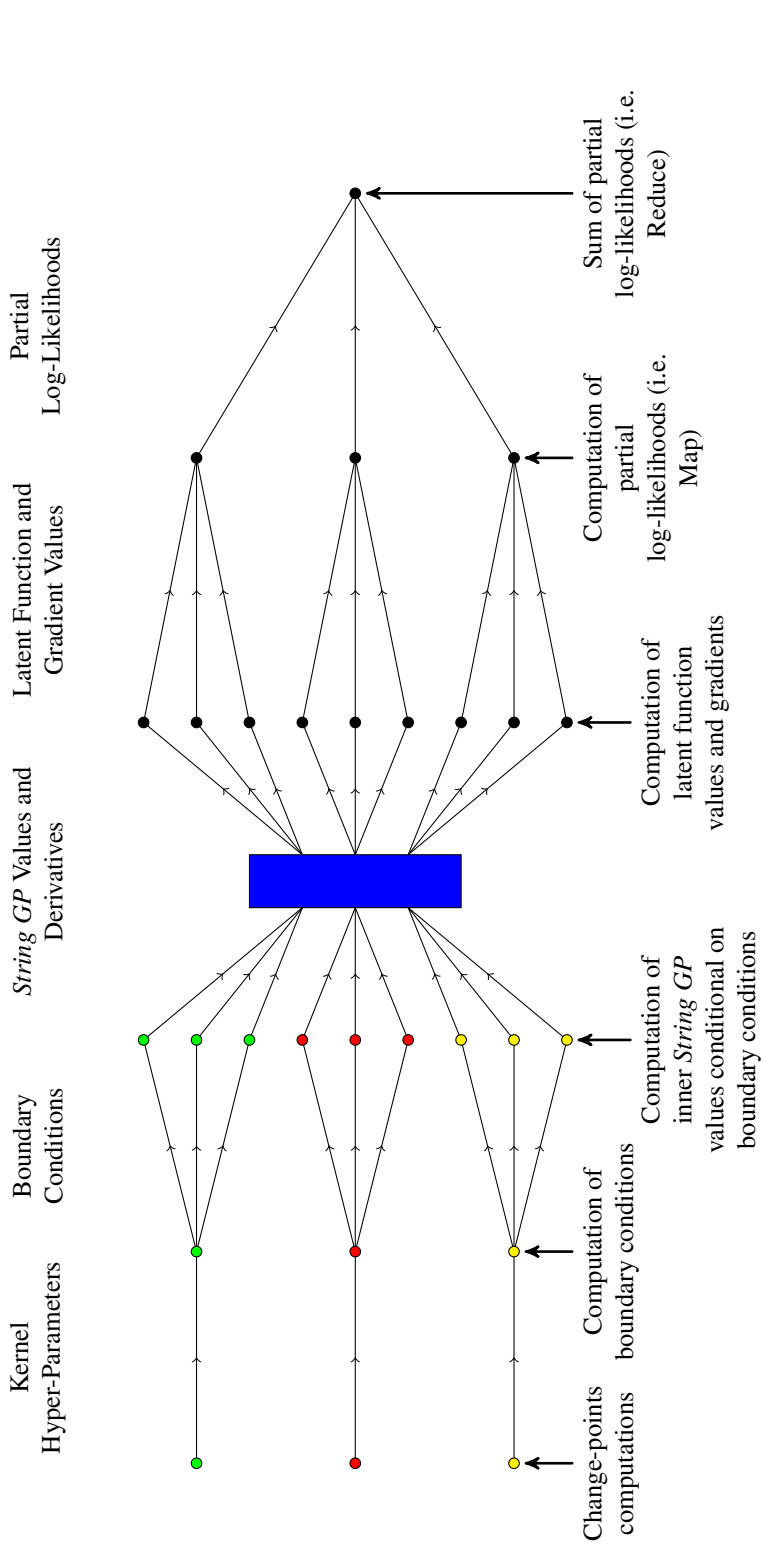

Figure 15: Example data flow graph for fully-distributed *string GP* inference under an i.i.d observations likelihood model. Here the input space is three-dimensional to ease illustration. Filled circles represent compute cores, and edges correspond to flows of data. Compute cores with the same colour (green, red or yellow) perform operations pertaining to the same input dimension, while black-filled circles represent compute cores performing cross-dimensional operations. The blue rectangle plays the role of a hub that relays *string GP* values to the compute cores that need them to compute the subset of latent function values and gradients they are responsible for. These values are then used to compute the log-likelihood in a distributed fashion using the Map-Reduce algorithm. Each calculation in the corresponding RJ-MCMC sampler would be initiated at one of the compute cores, and would trigger updates of all edges accessible from that compute core.

Acknowledgments

Yves-Laurent is a Google Fellow in Machine Learning and would like to acknowledge support from the Oxford-Man Institute. Wharton Research Data Services (WRDS) was used in preparing the data for Section 6.4 of this paper. This service and the data available thereon constitute valuable intellectual property and trade secrets of WRDS and/or its third-party suppliers.

Appendix A.

We begin by recalling Kolmogorov's extension theorem, which we will use to prove the existence of *derivative Gaussian processes* and *string Gaussian processes*.

Theorem 7 *(Kolmogorov's extension theorem, (Øksendal, 2003, Theorem 2.1.5))*
Let I be an interval, let all $t_1, \ldots, t_i \in I,\ i, n \in \mathbb{N}^*$, let $\nu_{t_1,\ldots,t_i}$ be probability measures on $\mathbb{R}^{ni}$ such that:

$$\nu_{t_{\pi(1)},\ldots,t_{\pi(i)}}(F_{\pi(1)}, \ldots, F_{\pi(i)}) = \nu_{t_1,\ldots,t_i}(F_1, \ldots, F_i) \tag{57}$$

for all permutations π on $\{1, \ldots, i\}$ and

$$\nu_{t_1,\ldots,t_i}(F_1, \ldots, F_i) = \nu_{t_1,\ldots,t_i,t_{i+1},\ldots,t_{i+m}}(F_1, \ldots, F_i, \mathbb{R}^n, \ldots, \mathbb{R}^n) \tag{58}$$

for all $m \in \mathbb{N}^*$ where the set on the right hand side has a total of $i + m$ factors. Then there exists a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and an $\mathbb{R}^n$ valued stochastic process $(X_t)_{t\in I}$ on Ω,

$$X_t : \Omega \to \mathbb{R}^n$$

such that

$$\nu_{t_1,\ldots,t_i}(F_1, \ldots, F_i) = \mathbb{P}(X_{t_1} \in F_1, \ldots, X_{t_i} \in F_i) \tag{59}$$

for all $t_1, \ldots, t_i \in I,\ i \in \mathbb{N}^*$ and for all Borel sets $F_1, \ldots, F_i$.

It is easy to see that every stochastic process satisfies the permutation and marginalisation conditions (57) and (58). The power of Kolmogorov's extension theorem is that it states that those two conditions are sufficient to guarantee the existence of a stochastic process.

Appendix B. Proof of Proposition 1

In this section we prove Proposition 1, which we recall below.

Proposition 1 (Derivative Gaussian processes)
Let I be an interval, $k : I \times I \to \mathbb{R}$ a $\mathcal{C}^2$ symmetric positive semi-definite function,[19] $m : I \to \mathbb{R}$ a $\mathcal{C}^1$ function.
(A) There exists a $\mathbb{R}^2$-valued stochastic process $(D_t)_{t\in I}$, $D_t = (z_t, z'_t)$, such that for all $t_1, \ldots, t_n \in I$,

$$(z_{t_1}, \ldots, z_{t_n}, z'_{t_1}, \ldots, z'_{t_n})$$

19. $\mathcal{C}^1$ (resp. $\mathcal{C}^2$) functions denote functions that are once (resp. twice) continuously differentiable on their domains.

is a Gaussian vector with mean

$$\left(m(t_1),\ldots,m(t_n),\frac{\mathrm{d}m}{\mathrm{dt}}(t_1),\ldots,\frac{\mathrm{d}m}{\mathrm{dt}}(t_n)\right)$$

and covariance matrix such that

$$\mathrm{cov}(z_{t_i},z_{t_j})=k(t_i,t_j),\quad \mathrm{cov}(z_{t_i},z'_{t_j})=\frac{\partial k}{\partial y}(t_i,t_j),\quad \text{and}\quad \mathrm{cov}(z'_{t_i},z'_{t_j})=\frac{\partial^2 k}{\partial x\partial y}(t_i,t_j).$$

We herein refer to $(D_t)_{t\in I}$ as a **derivative Gaussian process**.
(B) $(z_t)_{t\in I}$ is a Gaussian process with mean function m, covariance function k and that is $\mathcal{C}^1$ in the L^2 (mean square) sense.
(C) $(z'_t)_{t\in I}$ is a Gaussian process with mean function $\frac{\mathrm{d}m}{\mathrm{dt}}$ and covariance function $\frac{\partial^2 k}{\partial x\partial y}$. Moreover, $(z'_t)_{t\in I}$ is the L^2 derivative of the process $(z_t)_{t\in I}$.

Proof

Appendix B.1 Proof of Proposition 1 (A)

Firstly, we need to show that the matrix suggested in the proposition as the covariance matrix of $(z_{t_1},\ldots,z_{t_n},z'_{t_1},\ldots,z'_{t_n})$ is indeed positive semi-definite. To do so, we will show that it is the limit of positive definite matrices (which is sufficient to conclude it is positive semi-definite, as $x^T M_n x\geq 0$ for a convergent sequence of positive definite matrices implies $x^T M_\infty x\geq 0$).

Let k be as in the proposition, h such that $\forall i\leq n,\ t_i+h\in I$ and $(\tilde{z}_t)_{t\in I}$ be a Gaussian process with covariance function k. The vector

$$\left(\tilde{z}_{t_1},\ldots,\tilde{z}_{t_n},\frac{\tilde{z}_{t_1+h}-\tilde{z}_{t_1}}{h},\ldots,\frac{\tilde{z}_{t_n+h}-\tilde{z}_{t_n}}{h}\right)$$

is a Gaussian vector whose covariance matrix is positive definite and such that

$$\mathrm{cov}\left(\tilde{z}_{t_i},\tilde{z}_{t_j}\right)=k(t_i,t_j), \tag{60}$$

$$\mathrm{cov}\left(\tilde{z}_{t_i},\frac{\tilde{z}_{t_j+h}-\tilde{z}_{t_j}}{h}\right)=\frac{k(t_i,t_j+h)-k(t_i,t_j)}{h}, \tag{61}$$

and

$$\begin{aligned}&\mathrm{cov}\left(\frac{\tilde{z}_{t_i+h}-\tilde{z}_{t_i}}{h},\frac{\tilde{z}_{t_j+h}-\tilde{z}_{t_j}}{h}\right)\\&=\frac{1}{h^2}\left(k(t_i+h,t_j+h)-k(t_i+h,t_j)-k(t_i,t_j+h)+k(t_i,t_j)\right).\end{aligned} \tag{62}$$

As k is $\mathcal{C}^2$, $h\to k(x,y+h)$ admits a second order Taylor expansion about $h=0$ for every x, and we have:

$$k(x,y+h)=k(x,y)+\frac{\partial k}{\partial y}(x,y)h+\frac{1}{2}\frac{\partial^2 k}{\partial y^2}(x,y)h^2+o(h^2)=k(y+h,x). \tag{63}$$

Similarly, $h \to k(x+h, y+h)$ admits a second order Taylor expansion about $h = 0$ for every $x,\ y$ and we have:

$$k(x+h, y+h) = k(x,y) + \left[\frac{\partial k}{\partial x}(x,y) + \frac{\partial k}{\partial y}(x,y)\right] h + \left[\frac{\partial^2 k}{\partial x \partial y}(x,y) + \frac{1}{2}\frac{\partial^2 k}{\partial x^2}(x,y) + \frac{1}{2}\frac{\partial^2 k}{\partial y^2}(x,y)\right] h^2 + o(h^2). \tag{64}$$

Hence,

$$k(t_i, t_j + h) - k(t_i, t_j) = \frac{\partial k}{\partial y}(t_i, t_j) h + o(h), \tag{65}$$

and

$$k(t_i + h, t_j + h) - k(t_i + h, t_j) - k(t_i, t_j + h) + k(t_i, t_j) = \frac{\partial^2 k}{\partial x \partial y}(t_i, t_j) h^2 + o(h^2). \tag{66}$$

Dividing Equation (65) by h, dividing Equation (66) by h^2, and taking the limits, we obtain:

$$\lim_{h \to 0} \operatorname{cov}\left(\tilde{z}_{t_i}, \frac{\tilde{z}_{t_j+h} - \tilde{z}_{t_j}}{h}\right) = \frac{\partial k}{\partial y}(t_i, t_j),$$

and

$$\lim_{h \to 0} \operatorname{cov}\left(\frac{\tilde{z}_{t_i+h} - \tilde{z}_{t_i}}{h}, \frac{\tilde{z}_{t_j+h} - \tilde{z}_{t_j}}{h}\right) = \frac{\partial^2 k}{\partial x \partial y}(t_i, t_j),$$

which corresponds to the covariance structure of Proposition 1. In other words the proposed covariance structure is indeed positive semi-definite.

Let $\nu^{\mathcal{N}}_{t_1,\ldots,t_n}$ be the Gaussian probability measure corresponding to the joint distribution of $(z_{t_1}, \ldots, z_{t_n}, z'_{t_1}, \ldots, z'_{t_n})$ as per the Proposition 1, and let $\nu^{D}_{t_1,\ldots,t_n}$ be the measure on the Borel σ-algebra $\mathcal{B}(\underbrace{\mathbb{R}^2 \times \cdots \times \mathbb{R}^2}_{n \text{ times}})$ such that for any $2n$ intervals $I_{11}, I_{12}, \ldots, I_{n1}, I_{n2}$,

$$\nu^{D}_{t_1,\ldots,t_n}(I_{11} \times I_{12}, \ldots, I_{n1} \times I_{n2}) := \nu^{\mathcal{N}}_{t_1,\ldots,t_n}(I_{11}, \ldots, I_{n1}, I_{12}, \ldots, I_{n2}). \tag{67}$$

The measures $\nu^{D}_{t_1,\ldots,t_n}$ are the finite dimensional measures corresponding to the stochastic object $(D_t)_{t \in I}$ sampled at times $t_1, \ldots, t_n$. They satisfy the time permutation and marginalisation conditions of Kolmogorov's extension theorem as the Gaussian measures $\nu^{\mathcal{N}}_{t_1,\ldots,t_n}$ do. Hence, the $\mathbb{R}^2$-valued stochastic process $(D_t)_{t \in I}$ defined in Proposition 1 does exist.

Appendix B.2 Proof of Proposition 1 (B)

That $(z_t)_{t \in I}$ is a Gaussian process results from the fact that the marginals $(z_{t_1}, \ldots, z_{t_n})$ are Gaussian vectors with mean $(m(t_1), \ldots, m(t_n))$ and covariance matrix $[k(t_i, t_j)]_{i,j \in [1..n]}$. The fact that $(z_t)_{t \in I}$ is $\mathcal{C}^1$ in the L^2 sense is a direct consequence of the twice continuous differentiability of k.

Appendix B.3 Proof of Proposition 1 (C)

In effect, it follows from Proposition 1(A) that $\frac{z_{t+h}-z_t}{h} - z'_t$ is a Gaussian random variable with mean

$$\frac{m(t+h)-m(t)}{h} - \frac{\mathrm{d}m}{\mathrm{d}t}(t)$$

and variance

$$\frac{k(t+h,t+h) - 2k(t+h,t) + k(t,t) - 2\frac{\partial k}{\partial y}(t+h,t)h + 2\frac{\partial k}{\partial y}(t,t)h + \frac{\partial^2 k}{\partial x \partial y}(t,t)h^2}{h^2}.$$

Taking the second order Taylor expansion of the numerator in the fraction above about $h = 0$ we get $o(h^2)$, hence

$$\lim_{h\to 0} \mathrm{Var}\left(\frac{z_{t+h}-z_t}{h} - z'_t\right) = 0.$$

We also have

$$\lim_{h\to 0} \mathrm{E}\left(\frac{z_{t+h}-z_t}{h} - z'_t\right) = \frac{\mathrm{d}m}{dt}(t) - \mathrm{E}(z'_t) = 0.$$

Therefore,

$$\lim_{h\to 0} \mathrm{E}\left[\left(\frac{z_{t+h}-z_t}{h} - z'_t\right)^2\right] = 0,$$

which proves that (z'_t) is the L^2 derivative of (z_t). The fact that (z'_t) is a Gaussian process with mean function $\frac{\mathrm{d}m}{dt}$ and covariance function $\frac{\partial^2 k}{\partial x \partial y}$ is a direct consequence of the distribution of the marginals $(z'_{t_1}, \ldots, z'_{t_n})$. Moreover, the continuity of (z'_t) in the L^2 sense is a direct consequence of the continuity of $\frac{\partial^2 k}{\partial x \partial y}$ (see Rasmussen and Williams, 2006, p. 81 4.1.1). ■

Appendix C. Proof of Theorem 2

In this section we prove Theorem 2 which we recall below.

Theorem 2 (String Gaussian process)
Let $a_0 < \cdots < a_k < \cdots < a_K$, $I = [a_0, a_K]$ and let $p_{\mathcal{N}}(x; \mu, \Sigma)$ be the multivariate Gaussian density with mean vector μ and covariance matrix Σ. Furthermore, let $(m_k : [a_{k-1}, a_k] \to \mathbb{R})_{k\in[1..K]}$ be $\mathcal{C}^1$ functions, and $(k_k : [a_{k-1}, a_k] \times [a_{k-1}, a_k] \to \mathbb{R})_{k\in[1..K]}$ be $\mathcal{C}^3$ symmetric positive semi-definite functions, neither degenerate at a_{k-1}, nor degenerate at a_k given a_{k-1}.
(A) There exists an $\mathbb{R}^2$-valued stochastic process $(SD_t)_{t\in I}$, $SD_t = (z_t, z'_t)$ satisfying the following conditions:
1) The probability density of $(SD_{a_0}, \ldots, SD_{a_K})$ reads:

$$p_b(x_0, \ldots, x_K) := \prod_{k=0}^{K} p_{\mathcal{N}}\left(x_k; \mu_k^b, \Sigma_k^b\right)$$

where: $\Sigma_0^b = {}_1\mathbf{K}_{a_0;a_0}$, $\forall\, k > 0$ $\Sigma_k^b = {}_k\mathbf{K}_{a_k;a_k} - {}_k\mathbf{K}_{a_k;a_{k-1}}\, {}_k\mathbf{K}^{-1}_{a_{k-1};a_{k-1}}\, {}_k\mathbf{K}^{T}_{a_k;a_{k-1}}$,

$$\mu_0^b = {}_1\mathbf{M}_{a_0}, \quad \forall\, k > 0 \quad \mu_k^b = {}_k\mathbf{M}_{a_k} + {}_k\mathbf{K}_{a_k;a_{k-1}}\, {}_k\mathbf{K}^{-1}_{a_{k-1};a_{k-1}}(x_{k-1} - {}_k\mathbf{M}_{a_{k-1}}),$$

$$\text{with} \quad {}_k\mathbf{K}_{u;v} = \begin{bmatrix} k_k(u,v) & \frac{\partial k_k}{\partial y}(u,v) \\ \frac{\partial k_k}{\partial x}(u,v) & \frac{\partial^2 k_k}{\partial x \partial y}(u,v) \end{bmatrix}, \quad \text{and} \quad {}_k\mathbf{M}_u = \begin{bmatrix} m_k(u) \\ \frac{dm_k}{dt}(u) \end{bmatrix}.$$

2) Conditional on $(SD_{a_k} = x_k)_{k\in[0..K]}$, the restrictions $(SD_t)_{t\in]a_{k-1},a_k[},\ k \in [1..K]$ are **independent conditional derivative Gaussian processes**, respectively with unconditional mean function m_k and unconditional covariance function k_k and that are conditioned to take values x_{k-1} and x_k at a_{k-1} and a_k respectively. We refer to $(SD_t)_{t\in I}$ as a **string derivative Gaussian process**, and to its first coordinate $(z_t)_{t\in I}$ as a **string Gaussian process** namely,

$$(z_t)_{t\in I} \sim \mathcal{SGP}(\{a_k\}, \{m_k\}, \{k_k\}).$$

(B) The **string Gaussian process** $(z_t)_{t\in I}$ defined in (A) is $\mathcal{C}^1$ in the L^2 sense and its L^2 derivative is the process $(z'_t)_{t\in I}$ defined in (A).

Proof

Appendix C.1 Proof of Theorem 2 (A)

We will once again turn to Kolmogorov's extension theorem to prove the existence of the stochastic process $(SD_t)_{t\in I}$. The core of the proof is in the finite dimensional measures implied by Theorem 2 (A-1) and (A-2). Let $\left\{t_i^k \in]a_{k-1}, a_k[\right\}_{i\in[1..N_k],k\in[1..K]}$ be n times. We first formally construct the finite dimensional measures implied by Theorem 2 (A-1) and (A-2), and then verify that they satisfy the conditions of Kolmogorov's extension theorem.
Let us define the measure $\nu^{SD}_{t_1^1,\ldots,t^1_{N_1},\ldots,t_1^K,\ldots,t^K_{N_K},a_0,\ldots,a_K}$ as the probability measure having density with respect to the Lebesgue measure on $\mathcal{B}(\underbrace{\mathbb{R}^2 \times \cdots \times \mathbb{R}^2}_{1+n+K \text{ times}})$ that reads:

$$p_{SD}(x_{t_1^1}, \ldots, x_{t^1_{N_1}}, \ldots, x_{t_1^K}, \ldots, x_{t^K_{N_K}}, x_{a_0}, \ldots, x_{a_K}) = p_b(x_{a_0}, \ldots, x_{a_K}) \times \prod_{k=1}^{K} p_{\mathcal{N}}^{x_{a_{k-1}}, x_{a_k}}(x_{t_1^k}, \ldots, x_{t^k_{N_k}}) \tag{68}$$

where p_b is as per Theorem 2 (A-1) and $p_{\mathcal{N}}^{x_{a_{k-1}}, x_{a_k}}(x_{t_1^k}, \ldots, x_{t^k_{N_k}})$ is the (Gaussian) pdf of the joint distribution of the values at times $\{t_i^k \in]a_{k-1}, a_k[\}$ of the *conditional derivative Gaussian process* with unconditional mean functions m_k and unconditional covariance functions k_k that is conditioned to take values $x_{a_{k-1}} = (z_{a_{k-1}}, z'_{a_{k-1}})$ and $x_{a_k} = (z_{a_k}, z'_{a_k})$ at times a_{k-1} and a_k respectively (the corresponding—conditional—mean and covariance functions are derived from Equations (3 and 4). Let us extend the family of measures ν^{SD} to cases where some or all boundary times a_k are missing, by integrating out the corresponding variables in Equation (68). For instance when a_0 and a_1 are missing,

$$\begin{aligned} &\nu^{SD}_{t_1^1,\ldots,t^1_{N_1},\ldots,t_1^K,\ldots,t^K_{N_K},a_2,\ldots,a_K}(T_1^1, \ldots, T^1_{N_1}, \ldots, T_1^K, \ldots, T^K_{N_K}, A_2, \ldots, A_K) \\ &:= \nu^{SD}_{t_1^1,\ldots,t^1_{N_1},\ldots,t_1^K,\ldots,t^K_{N_K},a_0,\ldots,a_K}(T_1^1, \ldots, T^1_{N_1}, \ldots, T_1^K, \ldots, T^K_{N_K}, \mathbb{R}^2, \mathbb{R}^2, A_2, \ldots, A_K) \end{aligned} \tag{69}$$

where A_i and T^i_j are rectangle in $\mathbb{R}^2$. Finally, we extend the family of measures ν^{SD} to any arbitrary set of indices $\{t_1, \dots, t_n\}$ as follows:

$$\nu^{SD}_{t_1,\dots,t_n}(T_1, \dots, T_n) := \nu^{SD}_{t_{\pi^*(1)},\dots,t_{\pi^*(n)}}(T_{\pi^*(1)}, \dots, T_{\pi^*(n)}), \tag{70}$$

where π^* is a permutation of $\{1, \dots, n\}$ such that $\{t_{\pi^*(1)}, \dots, t_{\pi^*(n)}\}$ verify the following conditions:

1. $\forall\ i,\ j,$ if $t_i \in]a_{k_1-1}, a_{k_1}[,\ t_j \in]a_{k_2-1}, a_{k_2}[,$ and $k_1 < k_2$, then $\text{Idx}(t_i) < \text{Idx}(t_j)$. Where $\text{Idx}(t_i)$ stands for the index of t_i in $\{t_{\pi^*(1)}, \dots, t_{\pi^*(n)}\}$;
2. if $t_i \notin \{a_0, \dots, a_K\}$ and $t_j \in \{a_0, \dots, a_K\}$ then $\text{Idx}(t_i) < \text{Idx}(t_j)$;
3. if $t_i \in \{a_0, \dots, a_K\}$ and $t_j \in \{a_0, \dots, a_K\}$ then $\text{Idx}(t_i) < \text{Idx}(t_j)$ if and only if $t_i < t_j$.

Any such measure $\nu^{SD}_{t_{\pi^*(1)},\dots,t_{\pi^*(n)}}$ will fall in the category of either Equation (68) or Equation (69). Although π^* is not unique, any two permutations satisfying the above conditions will only differ by a permutation of times belonging to the same string interval $]a_{k-1}, a_k[$. Moreover, it follows from Equations (68) and (69) that the measures $\nu^{SD}_{t_{\pi^*(1)},\dots,t_{\pi^*(n)}}$ are invariant by permutation of times belonging to the same string interval $]a_{k-1}, a_k[$, and as a result any two π^* satisfying the above conditions will yield the same probability measure.

The finite dimensional probability measures $\nu^{SD}_{t_1,\dots,t_n}$ are the measures implied by Theorem 2. The permutation condition (57) of Kolmogorov's extension theorem is met by virtue of Equation (70). In effect for every permutation π of $\{1, \dots, n\}$, if we let $\pi' : \{\pi(1), \dots, \pi(n)\} \to \{\pi^*(1), \dots, \pi^*(n)\}$, then

$$\begin{aligned}
\nu^{SD}_{t_{\pi(1)},\dots,t_{\pi(n)}}(T_{\pi(1)}, \dots, T_{\pi(n)}) &:= \nu^{SD}_{t_{\pi'\circ\pi(1)},\dots,t_{\pi'\circ\pi(n)}}(T_{\pi'\circ\pi(1)}, \dots, T_{\pi'\circ\pi(n)}) \\
&= \nu^{SD}_{t_{\pi^*(1)},\dots,t_{\pi^*(n)}}(T_{\pi^*(1)}, \dots, T_{\pi^*(n)}) \\
&= \nu^{SD}_{t_1,\dots,t_n}(T_1, \dots, T_n).
\end{aligned}$$

As for the marginalisation condition (58), it is met for every boundary time by virtue of how we extended ν^{SD} to missing boundary times. All we need to prove now is that the marginalisation condition is also met at any non-boundary time. To do so, it is sufficient to prove that the marginalisation condition holds for t^1_1, that is:

$$\begin{aligned}
&\nu^{SD}_{t^1_1,\dots,t^1_{N_1},\dots,t^K_1,\dots,t^K_{N_K},a_0,\dots,a_K}(\mathbb{R}^2, T^1_2, \dots, T^1_{N_1}, \dots, T^K_1, \dots, T^K_{N_K}, A_0, \dots, A_K) \\
&= \nu^{SD}_{t^1_2,\dots,t^1_{N_1},\dots,t^K_1,\dots,t^K_{N_K},a_0,\dots,a_K}(T^1_2, \dots, T^1_{N_1}, \dots, T^K_1, \dots, T^K_{N_K}, A_0, \dots, A_K)
\end{aligned} \tag{71}$$

for every rectangles A_i and T^i_j in $\mathbb{R}^2$. In effect, cases where some boundary times are missing are special cases with the corresponding rectangles A_j set to $\mathbb{R}^2$. Moreover, if we prove Equation (71), the permutation property (57) will allow us to conclude that the marginalisation also holds true for any other (single) non-boundary time. Furthermore, if Equation (71) holds true, it can be shown that the marginalisation condition will also hold over multiple non-boundary times by using the permutation property (57) and marginalising one non-boundary time after another.

By Fubini's theorem, and considering Equation (68), showing that Equation (71) holds true is equivalent to showing that:

$$\int_{\mathbb{R}^2} p_{\mathcal{N}}^{x_{a_0},x_{a_1}}(x_{t_1^1},\ldots,x_{t_{N_1}^1})dx_{t_1^1} = p_{\mathcal{N}}^{x_{a_0},x_{a_1}}(x_{t_2^1},\ldots,x_{t_{N_1}^1}) \tag{72}$$

which holds true as $p_{\mathcal{N}}^{x_{a_0},x_{a_1}}(x_{t_1^1},\ldots,x_{t_{N_1}^1})$ is a multivariate Gaussian density, and the corresponding marginal is indeed the density of the same *conditional derivative Gaussian process* at times $t_2^1,\ldots,t_{N_1}^1$.

This concludes the proof of the existence of the stochastic process $(SD_t)_{t\in I}$.

Appendix C.2 Proof of Theorem 2 (B)

As conditional on boundary conditions the restriction of a *string derivative Gaussian process* on a string interval $[a_{k-1},a_k]$ is a *derivative Gaussian process*, it follows from Proposition 1 (C) that

$$\begin{aligned}&\forall\, \tilde{x}_{a_0},\ldots,\tilde{x}_{a_K},\ \forall\, t,t+h\in[a_{k-1},a_k],\\ &\lim_{h\to 0}\mathrm{E}\left(\left[\frac{z_{t+h}-z_t}{h}-z_t'\right]^2\middle|x_{a_0}=\tilde{x}_{a_0},\ldots,x_{a_K}=\tilde{x}_{a_K}\right)=0,\end{aligned} \tag{73}$$

or equivalently that:

$$\Delta z_h := \mathrm{E}\left(\left[\frac{z_{t+h}-z_t}{h}-z_t'\right]^2\middle|x_{a_0},\ldots,x_{a_K}\right)\xrightarrow[h\to 0]{a.s.}0. \tag{74}$$

Moreover,

$$\Delta z_h = \mathrm{Var}\left(\frac{z_{t+h}-z_t}{h}-z_t'\middle|x_{a_0},\ldots,x_{a_K}\right)+\mathrm{E}\left(\frac{z_{t+h}-z_t}{h}-z_t'\middle|x_{a_0},\ldots,x_{a_K}\right)^2. \tag{75}$$

As both terms in the sum of the above equation are non-negative, it follows that

$$\mathrm{Var}\left(\frac{z_{t+h}-z_t}{h}-z_t'\middle|x_{a_0},\ldots,x_{a_K}\right)\xrightarrow[h\to 0]{a.s.}0 \quad\text{and}\quad \mathrm{E}\left(\frac{z_{t+h}-z_t}{h}-z_t'\middle|x_{a_0},\ldots,x_{a_K}\right)^2\xrightarrow[h\to 0]{a.s.}0.$$

From which we deduce

$$\mathrm{E}\left(\frac{z_{t+h}-z_t}{h}-z_t'\middle|x_{a_0},\ldots,x_{a_K}\right)\xrightarrow[h\to 0]{a.s.}0.$$

As $\mathrm{E}\left(\frac{z_{t+h}-z_t}{h}-z_t'\middle|x_{a_0},\ldots,x_{a_K}\right)$ depends linearly on the boundary conditions, and as the boundary conditions are jointly-Gaussian (see Appendix H step 1), it follows that

$$\mathrm{E}\left(\frac{z_{t+h}-z_t}{h}-z_t'\middle|x_{a_0},\ldots,x_{a_K}\right)$$

is Gaussian. Finally we note that

$$\mathrm{Var}\left(\frac{z_{t+h}-z_t}{h}-z_t'\middle|x_{a_0},\ldots,x_{a_K}\right)$$

does not depend on the values of the boundary conditions x_{a_k} (but rather on the boundary times), and we recall that convergence almost sure of Gaussian random variables implies convergence in L^2. Hence, taking the expectation on both side of Equation (75) and then the limit as h goes to 0 we get

$$\mathrm{E}\left(\left[\frac{z_{t+h}-z_t}{h}-z'_t\right]^2\right)=\mathrm{E}(\Delta z_h)\underset{h\to 0}{\longrightarrow}0,$$

which proves that the *string GP* $(z_t)_{t\in I}$ is differentiable in the L^2 sense on I and has derivative $(z'_t)_{t\in I}$.

We prove the continuity in the L^2 sense of $(z'_t)_{t\in I}$ in a similar fashion, noting that conditional on the boundary conditions, $(z'_t)_{t\in I}$ is a Gaussian process whose mean function $\frac{dm_{ck}^{a_{k-1},a_k}}{dt}$ and covariance function $\frac{\partial^2 k_{ck}^{a_{k-1},a_k}}{\partial x\partial y}$ are continuous, thus is continuous in the L^2 sense on $[a_{k-1},a_k]$ (conditional on the boundary conditions). We therefore have that:

$$\forall\,\tilde{x}_{a_0},\ldots,\tilde{x}_{a_K},\ \forall\,t,t+h\in[a_{k-1},a_k],\ \lim_{h\to 0}\mathrm{E}\left((z'_{t+h}-z'_t)^2\middle|x_{a_0}=\tilde{x}_{a_0},\ldots,x_{a_K}=\tilde{x}_{a_K}\right)=0, \tag{76}$$

from which we get that:

$$\Delta z'_h:=\mathrm{E}\left(\left[z'_{t+h}-z'_t\right]^2\middle|x_{a_0},\ldots,x_{a_K}\right)\underset{h\to 0}{\overset{a.s.}{\longrightarrow}}0. \tag{77}$$

Moreover,

$$\Delta z'_h=\mathrm{Var}\left(z'_{t+h}-z'_t\middle|x_{a_0},\ldots,x_{a_K}\right)+\mathrm{E}\left(z'_{t+h}-z'_t\middle|x_{a_0},\ldots,x_{a_K}\right)^2, \tag{78}$$

which implies that

$$\mathrm{Var}\left(z'_{t+h}-z'_t\middle|x_{a_0},\ldots,x_{a_K}\right)\underset{h\to 0}{\overset{a.s.}{\longrightarrow}}0$$

and

$$\mathrm{E}\left(z'_{t+h}-z'_t\middle|x_{a_0},\ldots,x_{a_K}\right)^2\underset{h\to 0}{\overset{a.s.}{\longrightarrow}}0,$$

as both terms in the sum in Equation (78) are non-negative. Finally,

$$\mathrm{Var}\left(z'_{t+h}-z'_t\middle|x_{a_0},\ldots,x_{a_K}\right)$$

does not depend on the values of the boundary conditions, and

$$\mathrm{E}\left(z'_{t+h}-z'_t\middle|x_{a_0},\ldots,x_{a_K}\right)$$

is Gaussian for the same reason as before. Hence, taking the expectation on both sides of Equation (78), we get that

$$\mathrm{E}\left(\left[z'_{t+h}-z'_t\right]^2\right)=\mathrm{E}(\Delta z'_h)\underset{h\to 0}{\longrightarrow}0,$$

which proves that (z'_t) is continuous in the L^2 sense. ■

Appendix D. Proof of the Condition for Pathwise Regularity Upgrade of *String GPs* from L^2

In this section we prove that a sufficient condition for the process $(z'_t)_{t\in I}$ in Theorem 2 to be almost surely continuous and to be the almost sure derivative of the string Gaussian process $(z_t)_{t\in I}$, is that the Gaussian processes on $I_k = [a_{k-1}, a_k]$ with mean and covariance functions $m_{ck}^{a_{k-1},a_k}$ and $k_{ck}^{a_{k-1},a_k}$ (as per Equations 3 and 4 with $m := m_k$ and $k := k_k$) are themselves almost surely $\mathcal{C}^1$ for every boundary condition.

Firstly we note that the above condition guarantees that the result holds at non-boundary times. As for boundary times, the condition implies that the *string GP* is almost surely right differentiable (resp. left differentiable) at every left (resp. right) boundary time, including a_0 and a_K. Moreover, the *string GP* being differentiable in L^2, the right hand side and left hand side almost sure derivatives are the same, and are equal to the L^2 derivative, which proves that the L^2 derivatives at inner boundary times are also in the almost sure sense. A similar argument holds to conclude that the right (resp. left) hand side derivative at a_0 (resp. a_K) is also in the almost sure sense. Moreover, the derivative process $(z'_t)_{t\in I}$ admits an almost sure right hand side limit and an almost sure left hand side limit at every inner boundary time and both are equal as the derivative is continuous in L^2, which proves its almost sure continuity at inner boundary times. Almost sure continuity of $(z'_t)_{t\in I}$ on the right (resp. left) of a_0 (resp. a_K) is a direct consequence of the above condition.

Appendix E. Proof of Proposition 4

In this section, we prove Proposition 4, which we recall below.

Proposition 4 (Additively separable *string GPs* are flexible)
Let $k(x,y) := \rho\left(||x-y||^2_{L^2}\right)$ be a stationary covariance function generating a.s. $\mathcal{C}^1$ GP paths indexed on $\mathbb{R}^d$, $d>0$, and ρ a function that is $\mathcal{C}^2$ on $]0,+\infty[$ and continuous at 0. Let $\phi_s(x_1,\dots,x_d) = \sum_{j=1}^d x_j$, let $(z_t^j)_{t\in I^j,\ j\in[1..d]}$ be independent stationary Gaussian processes with mean 0 and covariance function k (where the L^2 norm is on $\mathbb{R}$), and let $f(t_1,\dots,t_d) = \phi_s(z^1_{t_1},\dots,z^d_{t_d})$ be the corresponding stationary string GP. Finally, let g be an isotropic Gaussian process indexed on $I^1\times\cdots\times I^d$ with mean 0 and covariance function k (where the L^2 norm is on $\mathbb{R}^d$). Then:
1) $\forall\, x\in I^1\times\cdots\times I^d,\ H(\nabla f(x)) = H(\nabla g(x))$,
2) $\forall\, x\neq y\in I^1\times\cdots\times I^d,\ I(\nabla f(x);\nabla f(y)) \leq I(\nabla g(x);\nabla g(y))$.

To prove Proposition 4 we need a lemma, which we state and prove below.

Lemma 8 *Let X_n be a sequence of Gaussian random vectors with auto-covariance matrix Σ_n and mean μ_n, converging almost surely to X_∞. If $\Sigma_n\to\Sigma_\infty$ and $\mu_n\to\mu_\infty$ then X_∞ is Gaussian with mean μ_∞ and auto-covariance matrix Σ_∞.*

Proof We need to show that the characteristic function of X_∞ is

$$\phi_{X_\infty}(t) := \mathrm{E}(e^{it^TX_\infty}) = e^{it^T\mu_\infty - \frac{1}{2}t^T\Sigma_\infty t}.$$

As Σ_n is positive semi-definite, $\forall n,\ |e^{it^T\mu_n-\frac{1}{2}t^T\Sigma_n t}| = e^{-\frac{1}{2}t^T\Sigma_n t} \leq 1$. Hence, by Lebesgue's dominated convergence theorem,

$$\phi_{X_\infty}(t) = \mathrm{E}(\lim_{n\to+\infty} e^{it^TX_n}) = \lim_{n\to+\infty}\mathrm{E}(e^{it^TX_n}) = \lim_{n\to+\infty} e^{it^T\mu_n-\frac{1}{2}t^T\Sigma_n t} = e^{it^T\mu_\infty-\frac{1}{2}t^T\Sigma_\infty t}.$$

■

Appendix E.1 Proof of Proposition 4 1)

Let $x = (t_1^x, \dots, t_d^x) \in I^1 \times \cdots \times I^d$. We want to show that $H(\nabla f(x)) = H(\nabla g(x))$ where f and g are as per Proposition 4, and H is the entropy operator. Firstly, we note from Equation (11) that

$$\nabla f(x) = \left(z_{t_1^x}^{1\prime}, \dots, z_{t_d^x}^{d\prime} \right), \tag{79}$$

where the joint law of the GP $(z_t^j)_{t \in I^j}$ and its derivative $(z_t^{j\prime})_{t \in I^j}$ is provided in Proposition 1. As the processes $\left(z_t^j, z_t^{j\prime} \right)_{t \in I^j}$, $j \in [1..d]$ are assumed to be independent of each other, $\nabla f(x)$ is a Gaussian vector and its covariance matrix reads:

$$\Sigma_{\nabla f(x)} = -2 \frac{\mathrm{d}\rho}{\mathrm{d}x}(0) \mathrm{I}_d, \tag{80}$$

where I_d is the $d \times d$ identity matrix. Hence,

$$H(\nabla f(x)) = \frac{d}{2} \left(1 + \ln(2\pi) \right) + \frac{1}{2} \ln |\Sigma_{\nabla f(x)}|. \tag{81}$$

Secondly, let e_j denote the d-dimensional vector whose j-th coordinate is 1 and every other coordinate is 0, and let $h \in \mathbb{R}$. As the proposition assumes the covariance function k generates almost surely $\mathcal{C}^1$ surfaces, the vectors $\left(\frac{g(x+he_1)-g(x)}{h}, \dots, \frac{g(x+he_d)-g(x)}{h} \right)$ are Gaussian vectors converging almost surely as $h \to 0$. Moreover, their mean is 0 and their covariance matrices have as element on the i-th row and j-th column ($i \neq j$):

$$\operatorname{cov}\left(\frac{g(x+he_i)-g(x)}{h}, \frac{g(x+he_j)-g(x)}{h} \right) = \frac{\rho(2h^2) - 2\rho(h^2) + \rho(0)}{h^2} \tag{82}$$

and as diagonal terms:

$$\operatorname{Var}\left(\frac{g(x+he_j)-g(x)}{h} \right) = 2 \frac{\rho(0) - \rho(h^2)}{h^2}. \tag{83}$$

Taking the limit of Equations (82) and (83) using the first order Taylor expansion of ρ (which the Proposition assumes is $\mathcal{C}^2$), we get that:

$$\Sigma_{\nabla g(x)} = -2 \frac{\mathrm{d}\rho}{\mathrm{d}x}(0) \mathrm{I}_d = \Sigma_{\nabla f(x)}, \tag{84}$$

It then follows from Lemma 8 that the limit $\nabla g(x)$ of

$$\left(\frac{g(x+he_1)-g(x)}{h}, \dots, \frac{g(x+he_d)-g(x)}{h} \right)$$

is also a Gaussian vector, which proves that $H(\nabla f(x)) = H(\nabla g(x))$.

Appendix E.2 Proof of Proposition 4 2)

We start by stating and proving another lemma we will later use.

Lemma 9 *Let A and B be two d-dimensional jointly Gaussian vectors with diagonal covariance matrices Σ_A and Σ_B respectively. Let $\Sigma_{A,B}$ be the cross-covariance matrix between A and B, and let $diag(\Sigma_{A,B})$ be the diagonal matrix whose diagonal is that of $\Sigma_{A,B}$. Then:*

$$det\left(\begin{bmatrix}\Sigma_A & diag(\Sigma_{A,B})\\ diag(\Sigma_{A,B}) & \Sigma_B\end{bmatrix}\right) \geq det\left(\begin{bmatrix}\Sigma_A & \Sigma_{A,B}\\ \Sigma_{A,B}^T & \Sigma_B\end{bmatrix}\right).$$

Proof Firstly we note that

$$\det\left(\begin{bmatrix}\Sigma_A & \mathrm{diag}(\Sigma_{A,B})\\ \mathrm{diag}(\Sigma_{A,B}) & \Sigma_B\end{bmatrix}\right) = \det(\Sigma_A)\det\left(\Sigma_B - \mathrm{diag}(\Sigma_{A,B})\Sigma_A^{-1}\mathrm{diag}(\Sigma_{A,B})\right)$$

and

$$\det\left(\begin{bmatrix}\Sigma_A & \Sigma_{A,B}\\ \Sigma_{A,B} & \Sigma_B\end{bmatrix}\right) = \det(\Sigma_A)\det\left(\Sigma_B - \Sigma_{A,B}^T\Sigma_A^{-1}\Sigma_{A,B}\right).$$

As the matrix Σ_A is positive semi-definite, $\det(\Sigma_A) \geq 0$. The case $\det(\Sigma_A) = 0$ is straight-forward. Thus we assume that $\det(\Sigma_A) > 0$, so that all we need to prove is that

$$\det\left(\Sigma_B - \mathrm{diag}(\Sigma_{A,B})\Sigma_A^{-1}\mathrm{diag}(\Sigma_{A,B})\right) \geq \det\left(\Sigma_B - \Sigma_{A,B}^T\Sigma_A^{-1}\Sigma_{A,B}\right).$$

Secondly, the matrix $\Sigma_{B|A}^{\text{diag}} := \Sigma_B - \mathrm{diag}(\Sigma_{A,B})\Sigma_A^{-1}\mathrm{diag}(\Sigma_{A,B})$ being diagonal, its determinant is the product of its diagonal terms:

$$\det(\Sigma_{B|A}^{\text{diag}}) = \prod_{i=1}^{d}\Sigma_{B|A}^{\text{diag}}[i,i] = \prod_{i=1}^{d}\left(\Sigma_B[i,i] - \frac{\Sigma_{A,B}[i,i]^2}{\Sigma_A[i,i]}\right).$$

As for the matrix $\Sigma_{B|A} := \Sigma_B - \Sigma_{A,B}^T\Sigma_A^{-1}\Sigma_{A,B}$, we note that it happens to be the covariance matrix of the (Gaussian) distribution of B given A, and thus is positive semi-definite and admits a Cholesky decomposition $\Sigma_{B|A} = LL^T$. It follows that

$$\begin{aligned}\det(\Sigma_{B|A}) = \prod_{i=1}^{d}L[i,i]^2 \leq \prod_{i=1}^{d}\Sigma_{B|A}[i,i] &= \prod_{i=1}^{d}\left(\Sigma_B[i,i] - \sum_{j=1}^{d}\frac{\Sigma_{A,B}[j,i]^2}{\Sigma_A[j,j]}\right)\\ &\leq \prod_{i=1}^{d}\left(\Sigma_B[i,i] - \frac{\Sigma_{A,B}[i,i]^2}{\Sigma_A[i,i]}\right) = \det(\Sigma_{B|A}^{\text{diag}}),\end{aligned} \tag{85}$$

where the first inequality results from the fact that $\Sigma_{B|A}[i,i] = \sum_{j=1}^{i}L[j,i]^2$ by definition of the Cholesky decomposition. This proves that

$$\det\left(\Sigma_B - \mathrm{diag}(\Sigma_{A,B})\Sigma_A^{-1}\mathrm{diag}(\Sigma_{A,B})\right) \geq \det\left(\Sigma_B - \Sigma_{A,B}^T\Sigma_A^{-1}\Sigma_{A,B}\right),$$

which as previously discussed concludes the proof of the lemma. ■

Proof of Proposition 4 2): Let $x = (t_1^x, \dots, t_d^x),\ y = (t_1^y, \dots, t_d^y) \in I^1 \times \cdots \times I^d,\ x \neq y$. We want to show that $I(\nabla f(x); \nabla f(y)) \leq I(\nabla g(x); \nabla g(y))$ where f and g are as per Proposition 4, and

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

is the mutual information between X and Y. As we have proved that $\forall\, x,\ H(\nabla f(x)) = H(\nabla g(x))$, all we need to prove now is that

$$H(\nabla f(x), \nabla f(y)) \geq H(\nabla g(x), \nabla g(y)).$$

Firstly, it follows from Equation (79) and the fact that the *derivative Gaussian processes* $(z_t^j, z_t^{j\prime})_{t \in I^j}$ are independent that $(\nabla f(x), \nabla f(y))$ is a jointly Gaussian vector. Moreover, the cross-covariance matrix $\Sigma_{\nabla f(x), \nabla f(y)}$ is diagonal with diagonal terms:

$$\Sigma_{\nabla f(x), \nabla f(y)}[i,i] = -2\left[\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(||x-y||_{L^2}^2\right) + 2(t_i^x - t_i^y)^2 \frac{\mathrm{d}^2\rho}{\mathrm{d}x^2}\left(||x-y||_{L^2}^2\right)\right]. \tag{86}$$

Secondly, it follows from a similar argument to the previous proof that $(\nabla g(x), \nabla g(y))$ is also a jointly Gaussian vector, and the terms $\Sigma_{\nabla g(x), \nabla g(y)}[i,j]$ are evaluated as limit of the cross-covariance terms $\mathrm{cov}\left(\frac{g(x+he_i)-g(x)}{h}, \frac{g(y+he_j)-g(y)}{h}\right)$ as $h \to 0$. For $i = j$,

$$\mathrm{cov}\left(\frac{g(x+he_i)-g(x)}{h}, \frac{g(y+he_i)-g(y)}{h}\right) = \frac{1}{h^2}\Bigg\{2\rho\left(\sum_k (t_k^x - t_k^y)^2\right)$$
$$- \rho\left(\sum_{k\neq i}(t_k^x - t_k^y)^2 + (t_i^x + h - t_i^y)^2\right) - \rho\left(\sum_{k\neq i}(t_k^x - t_k^y)^2 + (t_i^x - h - t_i^y)^2\right)\Bigg\}, \tag{87}$$

As ρ is assumed to be $\mathcal{C}^2$, the below Taylor expansions around $h = 0$ hold true:

$$\rho\left(\sum_k (t_k^x - t_k^y)^2\right) - \rho\left(\sum_{k\neq i}(t_k^x - t_k^y)^2 + (t_i^x - h - t_i^y)^2\right) = 2(t_i^x - t_i^y)h\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(\sum_k (t_k^x - t_k^y)^2\right) \tag{88}$$
$$- \left[\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(\sum_k (t_k^x - t_k^y)^2\right) + 2(t_i^x - t_i^y)^2\frac{\mathrm{d}^2\rho}{\mathrm{d}x^2}\left(\sum_k (t_k^x - t_k^y)^2\right)\right]h^2 + o(h^2)$$

$$\rho\left(\sum_k (t_k^x - t_k^y)^2\right) - \rho\left(\sum_{k\neq i}(t_k^x - t_k^y)^2 + (t_i^x + h - t_i^y)^2\right) = -2(t_i^x - t_i^y)h\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(\sum_k (t_k^x - t_k^y)^2\right) \tag{89}$$
$$- \left[\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(\sum_k (t_k^x - y_k)^2\right) + 2(t_i^x - t_i^y)^2\frac{\mathrm{d}^2\rho}{\mathrm{d}x^2}\left(\sum_k (t_k^x - t_k^y)^2\right)\right]h^2 + o(h^2)$$

Plugging Equations (88) and (89) into Equation (87) and taking the limit we obtain:

$$\begin{aligned}\Sigma_{\nabla g(x),\nabla g(y)}[i,i] &= -2\left[\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(||x-y||_{L^2}^2\right) + 2(t_i^x - t_i^y)^2\frac{\mathrm{d}^2\rho}{\mathrm{d}x^2}\left(||x-y||_{L^2}^2\right)\right] \\ &= \Sigma_{\nabla f(x),\nabla f(y)}[i,i]. \end{aligned} \tag{90}$$

Similarly for $i \neq j$,

$$\begin{aligned}&\mathrm{cov}\left(\frac{g(x+he_i)-g(x)}{h}, \frac{g(y+he_j)-g(y)}{h}\right) \\ &= \frac{1}{h^2}\Bigg\{\rho\left(\sum_{k\neq i,j}(t_k^x - t_k^y)^2 + (t_i^x + h - t_i^y)^2 + (t_j^x - h - t_j^y)^2\right) \\ &- \rho\left(\sum_{k\neq i}(t_k^x - t_k^y)^2 + (t_i^x + h - t_i^y)^2\right) - \rho\left(\sum_{k\neq j}(t_k^x - t_k^y)^2 + (t_i^x - h - t_j^y)^2\right) \\ &+ \rho\left(\sum_k (t_k^x - t_k^y)^2\right)\Bigg\}, \end{aligned} \tag{91}$$

and

$$\begin{aligned}&\rho\left(\sum_{k\neq i,j}(t_k^x - t_k^y)^2 + (t_i^x + h - t_i^y)^2 + (t_j^x - h - t_j^y)^2\right) - \rho\left(\sum_k (t_k^x - t_k^y)^2\right) \\ &= \left[2\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(\sum_k (t_k^x - t_k^y)^2\right) + 2\left((t_i^x - t_i^y) - (t_j^x - t_j^y)\right)^2 \times \frac{\mathrm{d}^2\rho}{\mathrm{d}x^2}\left(\sum_k (t_k^x - t_k^y)^2\right)\right]h^2 \\ &+ 2\left(t_i^x - t_i^y - t_j^x + t_j^y\right)\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(\sum_k (t_k^x - t_k^y)^2\right)h + o(h^2). \end{aligned} \tag{92}$$

Plugging Equations (88), (89) and (92) in Equation (91) and taking the limit we obtain:

$$\Sigma_{\nabla g(x),\nabla g(y)}[i,j] = -4(t_i^x - t_i^y)(t_j^x - t_j^y)\frac{\mathrm{d}^2\rho}{\mathrm{d}x^2}\left(||x-y||_{L^2}^2\right). \tag{93}$$

To summarize, $(\nabla f(x), \nabla f(y))$ and $(\nabla g(x), \nabla g(y))$ are both jointly Gaussian vectors; $\nabla f(x)$, $\nabla g(x)$, $\nabla f(y)$, and $\nabla g(y)$ are (Gaussian) identically distributed with a diagonal covariance matrix; $\Sigma_{\nabla f(x),\nabla f(y)}$ is diagonal; $\Sigma_{\nabla g(x),\nabla g(y)}$ has the same diagonal as $\Sigma_{\nabla f(x),\nabla f(y)}$ but has possibly non-zero off-diagonal terms. Hence, it follows from Lemma 9 that the determinant of the auto-covariance matrix of $(\nabla f(x), \nabla f(y))$ is higher than that of the auto-covariance matrix of $(\nabla g(x), \nabla g(y))$; or equivalently the entropy of $(\nabla f(x), \nabla f(y))$ is higher than that of $(\nabla g(x), \nabla g(y))$ (as both are Gaussian vectors), which as previously discussed is sufficient to conclude that the mutual information between $\nabla f(x)$ and $\nabla f(y)$ is smaller than that between $\nabla g(x)$ and $\nabla g(y)$.

Appendix F. Proof of Proposition 6

In this section, we prove Proposition 6, which we recall below.

Proposition 6 (Extension of the *standard GP paradigm*)
Let $K \in \mathbb{N}^*$, let $I = [a_0, a_K]$ and $I_k = [a_{k-1}, a_k]$ be intervals with $a_0 < \cdots < a_K$. Furthermore, let $m : I \to \mathbb{R}$ be a $\mathcal{C}^1$ function, m_k the restriction of m to I_k, $h : I \times I \to \mathbb{R}$ a $\mathcal{C}^3$ symmetric positive semi-definite function, and h_k the restriction of h to $I_k \times I_k$. If

$$(z_t)_{t\in I} \sim \mathcal{SGP}(\{a_k\}, \{m_k\}, \{h_k\}),$$

then

$$\forall\, k \in [1..K],\ (z_t)_{t\in I_k} \sim \mathcal{GP}(m, h).$$

Proof
To prove Proposition 6, we consider the *string derivative Gaussian process* (Theorem 2) $(SD_t)_{t\in I}$, $SD_t = (z_t, z'_t)$ with unconditional string mean and covariance functions as per Proposition 6 and prove that its restrictions on the intervals $I_k = [a_{k-1}, a_k]$ are *derivative Gaussian processes* with the same mean function m and covariance function h. Proposition 1(B) will then allow us to conclude that $(z_t)_{t\in I_k}$ are GPs with mean m and covariance function h.

Let $t_1, \ldots, t_n \in]a_{k-1}, a_k[$ and let $p_D(x_{a_{k-1}})$ (respectively $p_D(x_{a_k}|x_{a_{k-1}})$ and $p_D(x_{t_1}, \ldots, x_{t_n}|x_{a_{k-1}}, x_{a_k})$) denote the pdf of the value of the *derivative Gaussian process* with mean function m and covariance function h at a_{k-1} (respectively its value at a_k conditional on its value at a_{k-1}, and its values at $t_1, \ldots, t_n$ conditional on its values at a_{k-1} and a_k). Saying that the restriction of the *string derivative Gaussian process* (SD_t) on $[a_{k-1}, a_k]$ is the *derivative Gaussian process* with mean m and covariance h is equivalent to saying that all finite dimensional marginals of the *string derivative Gaussian process* $p_{SD}(x_{a_{k-1}}, x_{t_1}, \ldots, x_{t_n}, x_{a_k}),\ t_i \in [a_{k-1}, a_k]$, factorise as[20]:

$$p_{SD}(x_{a_{k-1}}, x_{t_1}, \ldots, x_{t_n}, x_{a_k}) = p_D(x_{a_{k-1}}) p_D(x_{a_k}|x_{a_{k-1}}) p_D(x_{t_1}, \ldots, x_{t_n}|x_{a_{k-1}}, x_{a_k}).$$

Moreover, we know from Theorem 2 that by design, $p_{SD}(x_{a_{k-1}}, x_{t_1}, \ldots, x_{t_n}, x_{a_k})$ factorises as

$$p_{SD}(x_{a_{k-1}}, x_{t_1}, \ldots, x_{t_n}, x_{a_k}) = p_{SD}(x_{a_{k-1}}) p_D(x_{a_k}|x_{a_{k-1}}) p_D(x_{t_1}, \ldots, x_{t_n}|x_{a_{k-1}}, x_{a_k}).$$

In other words, all we need to prove is that

$$p_{SD}(x_{a_k}) = p_D(x_{a_k})$$

for every boundary time, which we will do by induction. We note by integrating out every boundary condition but the first in p_b (as per Theorem 2 (a-1)) that

$$p_{SD}(x_{a_0}) = p_D(x_{a_0}).$$

If we assume that $p_{SD}(x_{a_{k-1}}) = p_D(x_{a_{k-1}})$ for some $k > 0$, then as previously discussed the restriction of the *string derivative Gaussian process* on $[a_{k-1}, a_k]$ will be the *derivative Gaussian process* with the same mean and covariance functions, which will imply that $p_{SD}(x_{a_k}) = p_D(x_{a_k})$. This concludes the proof. ■

20. We emphasize that the terms on the right hand-side of this equation involve p_D not p_{SD}.

Appendix G. Proof of Lemma 10

In this section, we will prove Lemma 10 that we recall below.

Lemma 10 Let X be a multivariate Gaussian with mean μ_X and covariance matrix Σ_X. If conditional on X, Y is a multivariate Gaussian with mean $MX + A$ and covariance matrix Σ_Y^c where M, A and Σ_Y^c do not depend on X, then (X, Y) is a jointly Gaussian vector with mean

$$\mu_{X;Y} = \begin{bmatrix} \mu_X \\ M\mu_X + A \end{bmatrix},$$

and covariance matrix

$$\Sigma_{X;Y} = \begin{bmatrix} \Sigma_X & \Sigma_X M^T \\ M\Sigma_X & \Sigma_Y^c + M\Sigma_X M^T \end{bmatrix}.$$

Proof To prove this lemma we introduce two vectors $\tilde{X}$ and $\tilde{Y}$ whose lengths are the same as those of X and Y respectively, and such that $(\tilde{X}, \tilde{Y})$ is jointly Gaussian with mean $\mu_{X;Y}$ and covariance matrix $\Sigma_{X;Y}$. We then prove that the (marginal) distribution of $\tilde{X}$ is the same as the distribution of X and that the distribution of $\tilde{Y}|\tilde{X} = x$ is the same as $Y|X = x$ for any x, which is sufficient to conclude that (X, Y) and $(\tilde{X}, \tilde{Y})$ have the same distribution.

It is obvious from the joint $(\tilde{X}, \tilde{Y})$ that $\tilde{X}$ is Gaussian distribution with mean μ_X and covariance matrix Σ_X. As for the distribution of $\tilde{Y}$ conditional on $\tilde{X} = x$, it follows from the usual Gaussian identities that it is Gaussian with mean

$$M\mu_X + c + M\Sigma_X\Sigma_X^{-1}(x - \mu_X) = Mx + c,$$

and covariance matrix

$$\Sigma_Y^c + M\Sigma_X M^T - M\Sigma_X\Sigma_X^{-1}\Sigma_X^T M^T = \Sigma_Y^c,$$

which is the same distribution as that of $Y|X = x$ since the covariance matrix Σ_X is symmetric. This concludes our proof. ■

Appendix H. Proof of Proposition 5

In this section we will prove that *string GPs* with link function ϕ_s are GPs, or in other words that if f is a *string GP* indexed on $\mathbb{R}^d$, $d > 0$ with link function $\phi_s(x_1, \ldots, x_d) = \sum_{j=1}^d x_j$, then $(f(x_1), \ldots, f(x_n))$ has a multivariate Gaussian distribution for every set of distinct points $x_1, \ldots, x_n \in \mathbb{R}^d$.

Proof As the sum of independent Gaussian processes is a Gaussian process, a sufficient condition for additively separable *string GPs* to be GPs in dimensions $d > 1$ is that *string GPs* be GPs in dimension 1. Hence, all we need to do is to prove that *string GPs* are GPs in dimension 1.

Let $(z_t^j, z_t^{j\prime})_{t \in I^j}$ be a string derivative GP in dimension 1, with boundary times $a_0^j, \ldots, a_{K^j}^j$, and unconditional string mean and covariance functions m_k^j and k_k^j respectively. We want to prove that $(z_{t_1}^j, \ldots, z_{t_n}^j)$ is jointly Gaussian for any $t_1, \ldots, t_n \in I^j$.

APPENDIX H.0.1 STEP 1 $\left(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{K^j}}, z^{j\prime}_{a_{K^j}}\right)$ IS JOINTLY GAUSSIAN

We first prove recursively that the vector $\left(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{K^j}}, z^{j\prime}_{a_{K^j}}\right)$ is jointly Gaussian. We note from Theorem 2 that $(z^j_t, z^{j\prime}_t)_{t\in[a_0,a_1]}$ is the *derivative Gaussian process* with mean m^j_1 and covariance function k^j_1. Hence, $(z^j_{a_0}, z^{j\prime}_{a_0}, z^j_{a_1}, z^{j\prime}_{a_1})$ is jointly Gaussian. Moreover, let us assume that $\mathcal{B}_{k-1} := (z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{k-1}}, z^{j\prime}_{a_{k-1}})$ is jointly Gaussian for some $k > 1$. Conditional on $\mathcal{B}_{k-1}$, $(z^j_{a_k}, z^{j\prime}_{a_k})$ is Gaussian with covariance matrix independent of $\mathcal{B}_{k-1}$, and with mean

$$\begin{bmatrix} m^j_k(a^j_k) \\ \frac{dm^j_k}{dt}(a^j_k) \end{bmatrix} + {}^j_k\mathbf{K}_{a^j_k;a^j_{k-1}} \, {}^j_k\mathbf{K}^{-1}_{a_{k-1};a^j_{k-1}} \begin{bmatrix} z^j_{a^j_{k-1}} - m^j_k(a^j_{k-1}) \\ z^{j\prime}_{a^j_{k-1}} - \frac{dm^j_k}{dt}(a^j_{k-1}) \end{bmatrix},$$

which depends linearly on $(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{k-1}}, z^{j\prime}_{a_{k-1}})$. Hence by Lemma 10,

$$(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_k}, z^{j\prime}_{a_k})$$

is jointly Gaussian.

APPENDIX H.0.2 STEP 2 $(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{K^j}}, z^{j\prime}_{a_{K^j}}, \ldots, z^j_{t^k_i}, z^{j\prime}_{t^k_i}, \ldots)$ IS JOINTLY GAUSSIAN

Let $t^k_1, \ldots, t^k_{n^k} \in]a^j_{k-1}, a^j_k[$, $k \leq K^j$ be distinct string times. We want to prove that the vector $(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{K^j}}, z^{j\prime}_{a_{K^j}}, \ldots, z^j_{t^k_i}, z^{j\prime}_{t^k_i}, \ldots)$ where all boundary times are represented, and for any finite number of string times is jointly Gaussian. Firstly, we have already proved that $(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{K^j}}, z^{j\prime}_{a_{K^j}})$ is jointly Gaussian. Secondly, we note from Theorem 2 that conditional on $(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{K^j}}, z^{j\prime}_{a_{K^j}})$, $(\ldots, z^j_{t^k_i}, z^{j\prime}_{t^k_i}, \ldots)$ is a Gaussian vector whose covariance matrix does not depend on $(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{K^j}}, z^{j\prime}_{a_{K^j}})$, and whose mean depends linearly on

$$\left(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{K^j}}, z^{j\prime}_{a_{K^j}}\right).$$

Hence,

$$\left(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{K^j}}, z^{j\prime}_{a_{K^j}}, \ldots, z^j_{t^k_i}, z^{j\prime}_{t^k_i}, \ldots\right)$$

is jointly Gaussian (by Lemma 10).

APPENDIX H.0.3 STEP 3 $(z^j_{t_1}, \ldots, z^j_{t_n})$ IS JOINTLY GAUSSIAN

$(z^j_{t_1}, z^{j\prime}_{t_1}, \ldots, z^j_{t_n}, z^{j\prime}_{t_n})$ is jointly Gaussian as it can be regarded as the marginal of some joint distribution of the form $(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{K^j}}, z^{j\prime}_{a_{K^j}}, \ldots, z^j_{t^k_i}, z^{j\prime}_{t^k_i}, \ldots)$. Hence, its marginal $(z^j_{t_1}, \ldots, z^j_{t_n})$ is also jointly Gaussian, which concludes our proof. ■

Appendix I. Derivation of Global String GP Mean and Covariance Functions

We begin with *derivative string GPs* indexed on $\mathbb{R}$. Extensions to *membrane GPs* are easily achieved for a broad range of link functions. In our exposition, we focus on the class of *elementary symmetric*

polynomials (Macdonald (1995)). In addition to containing the link function ϕ_s previously introduced, this family of polynomials yields global covariance structures that have many similarities with existing kernel approaches, which we discuss in Section 4.3.

For $n \leq d$, the n-th order elementary symmetric polynomial is given by

$$e_0(x_1, \ldots, x_d) := 1, \quad \forall 1 \leq n \leq d \;\; e_n(x_1, \ldots, x_d) = \sum_{1 \leq j_1 < j_2 < \cdots < j_n \leq d} \prod_{k=1}^{n} x_{j_k}. \tag{94}$$

As an illustration,

$$e_1(x_1, \ldots, x_d) = \sum_{j=1}^{d} x_j = \phi_s(x_1, \ldots, x_d),$$

$$e_2(x_1, \ldots, x_d) = x_1x_2 + x_1x_3 + \cdots + x_1x_d + \cdots + x_{d-1}x_d,$$

$$\ldots$$

$$e_d(x_1, \ldots, x_d) = \prod_{j=1}^{d} x_j = \phi_p(x_1, \ldots, x_d).$$

Let f denote a *membrane GP* indexed on $\mathbb{R}^d$ with link function e_n and by $(z_t^1), \ldots, (z_t^d)$ its independent building block *string GPs*. Furthermore, let m_k^j and k_k^j denote the unconditional mean and covariance functions corresponding to the k-th string of (z_t^j) defined on $[a_{k-1}^j, a_k^j]$. Finally, let us define

$$\boxed{\bar{m}^j(t) := \mathrm{E}(z_t^j), \quad \bar{m}^{j\prime}(t) := \mathrm{E}(z_t^{j\prime}),}$$

the global mean functions of the j-th building block *string GP* and of its derivative, where $\forall t \in I^j$. It follows from the independence of the building block *string GPs* (z_t^j) that:

$$\boxed{\bar{m}^f(t_1, \ldots, t_d) := \mathrm{E}(f(t_1, \ldots, t_d)) = e_n(\bar{m}^1(t_1), \ldots, \bar{m}^d(t_d)).}$$

Moreover, noting that

$$\frac{\partial e_n}{\partial x_j} = e_{n-1}(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d),$$

it follows that:

$$\boxed{\bar{m}^{\nabla f}(t_1, \ldots, t_d) := \mathrm{E}\left(\nabla f(t_1, \ldots, t_d)\right) = \begin{bmatrix} \bar{m}^{1\prime}(t_1) e_{n-1}\left(\bar{m}^2(t_2), \ldots, \bar{m}^d(t_d)\right) \\ \cdots \\ \bar{m}^{d\prime}(t_d) e_{n-1}\left(\bar{m}^1(t_1), \ldots, \bar{m}^{d-1}(t_{d-1})\right) \end{bmatrix}.}$$

Furthermore, for any $u_j, v_j \in I^j$ we also have that

$$\boxed{\mathrm{cov}\left(f(u_1, \ldots, u_d), f(v_1, \ldots, v_d)\right) = e_n\left(\mathrm{cov}(z_{u_1}^1, z_{v_1}^1), \ldots, \mathrm{cov}(z_{u_d}^d, z_{v_d}^d)\right),}$$

$$\boxed{\mathrm{cov}\left(\frac{\partial f}{\partial x_i}(u_1, \ldots, u_d), f(v_1, \ldots, v_d)\right) = e_n\left(\mathrm{cov}(z_{u_1}^1, z_{v_1}^1), \ldots, \mathrm{cov}(z_{u_i}^{i\prime}, z_{v_i}^i), \ldots, \mathrm{cov}(z_{u_d}^d, z_{v_d}^d)\right),}$$

and for $i \leq j$

$$\boxed{\begin{aligned}&\operatorname{cov}\left(\frac{\partial f}{\partial x_i}(u_1,\ldots,u_d), \frac{\partial f}{\partial x_j}(v_1,\ldots,v_d)\right)\\ &= \begin{cases} e_n\left(\operatorname{cov}(z^1_{u_1}, z^1_{v_1}), \ldots, \operatorname{cov}(z^{i\prime}_{u_i}, z^i_{v_i}), \ldots, \operatorname{cov}(z^{j\prime}_{u_j}, z^j_{v_j}), \ldots, \operatorname{cov}(z^d_{u_d}, z^d_{v_d})\right), & \text{if } i < j \\ e_n\left(\operatorname{cov}(z^1_{u_1}, z^1_{v_1}), \ldots, \operatorname{cov}(z^{i\prime}_{u_i}, z^{i\prime}_{v_i}), \ldots, \operatorname{cov}(z^d_{u_d}, z^d_{v_d})\right), & \text{if } i = j \end{cases}.\end{aligned}}$$

Overall, for any elementary symmetric polynomial link function, multivariate mean and covariance functions are easily deduced from the previously boxed equations and the univariate quantities

$$\bar{m}^j(u),\ \bar{m}^{j\prime}(u),\ \text{and } {}^j\bar{\mathbf{K}}_{u;v} := \begin{bmatrix} \operatorname{cov}(z^j_u, z^j_v) & \operatorname{cov}(z^j_u, z^{j\prime}_v) \\ \operatorname{cov}(z^{j\prime}_u, z^j_v) & \operatorname{cov}(z^{j\prime}_u, z^{j\prime}_v) \end{bmatrix} = {}^j\bar{\mathbf{K}}^T_{v;u},$$

which we now derive. In this regards, we will need the following lemma.

Lemma 10 *Let X be a multivariate Gaussian with mean μ_X and covariance matrix Σ_X. If conditional on X, Y is a multivariate Gaussian with mean $MX + A$ and covariance matrix Σ^c_Y where M, A and Σ^c_Y do not depend on X, then (X, Y) is a jointly Gaussian vector with mean*

$$\mu_{X;Y} = \begin{bmatrix} \mu_X \\ M\mu_X + A \end{bmatrix},$$

and covariance matrix

$$\Sigma_{X;Y} = \begin{bmatrix} \Sigma_X & \Sigma_X M^T \\ M\Sigma_X & \Sigma^c_Y + M\Sigma_X M^T \end{bmatrix}.$$

Proof See Appendix G. ■

APPENDIX I.0.1 GLOBAL STRING GP MEAN FUNCTIONS

We now turn to evaluating the univariate global mean functions $\bar{m}^j$ and $\bar{m}^{j\prime}$. We start with boundary times and then generalise to other times.

Boundary times: We note from Theorem 2 that the restriction $\left(z^j_t, z^{j\prime}_t\right)_{t\in[a^j_0, a^j_1]}$ is the *derivative Gaussian process* with mean and covariance functions m^j_1 and k^j_1. Thus,

$$\boxed{\begin{bmatrix} \bar{m}^j(a^j_0) \\ \bar{m}^{j\prime}(a^j_0) \end{bmatrix} = \begin{bmatrix} m^j_1(a^j_0) \\ \frac{dm^j_1}{dt}(a^j_0) \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} \bar{m}^j(a^j_1) \\ \bar{m}^{j\prime}(a^j_1) \end{bmatrix} = \begin{bmatrix} m^j_1(a^j_1) \\ \frac{dm^j_1}{dt}(a^j_1) \end{bmatrix}.}$$

For $k > 1$, we recall that conditional on $\left(z^j_{a^j_{k-1}}, z^{j\prime}_{a^j_{k-1}}\right)$, $\left(z^j_{a^j_k}, z^{j\prime}_{a^j_k}\right)$ is Gaussian with mean

$$\begin{bmatrix} m^j_k(a^j_k) \\ \frac{dm^j_k}{dt}(a^j_k) \end{bmatrix} + {}^j_k\mathbf{K}_{a^j_k;a^j_{k-1}}\ {}^j_k\mathbf{K}^{-1}_{a^j_{k-1};a^j_{k-1}} \begin{bmatrix} z^j_{a^j_{k-1}} - m^j_k(a^j_{k-1}) \\ z^{j\prime}_{a^j_{k-1}} - \frac{dm^j_k}{dt}(a^j_{k-1}) \end{bmatrix},$$

$$\text{with } {}^{j}_{k}\mathbf{K}_{u;v} = \begin{bmatrix} k^j_k(u,v) & \frac{\partial k^j_k}{\partial y}(u,v) \\ \frac{\partial k^j_k}{\partial x}(u,v) & \frac{\partial^2 k^j_k}{\partial x \partial y}(u,v) \end{bmatrix}.$$

It then follows from the law of total expectations that for all $k > 1$

$$\boxed{\begin{bmatrix} \bar{m}^j(a^j_k) \\ \bar{m}^{j\prime}(a^j_k) \end{bmatrix} = \begin{bmatrix} m^j_k(a^j_k) \\ \frac{dm^j_k}{dt}(a^j_k) \end{bmatrix} + {}^{j}_{k}\mathbf{K}_{a^j_k;a^j_{k-1}} \, {}^{j}_{k}\mathbf{K}^{-1}_{a^j_{k-1};a^j_{k-1}} \begin{bmatrix} \bar{m}^j(a^j_{k-1}) - m^j_k(a^j_{k-1}) \\ \bar{m}^{j\prime}(a^j_{k-1}) - \frac{dm^j_k}{dt}(a^j_{k-1}) \end{bmatrix}.}$$

String times: As for non-boundary times $t \in]a^j_{k-1}, a^j_k[$, conditional on $\left(z^j_{a_{k-1}}, z^{j\prime}_{a_{k-1}}\right)$ and $\left(z^j_{a_k}, z^{j\prime}_{a_k}\right)$, $\left(z^j_t, z^{j\prime}_t\right)$ is Gaussian with mean

$$\begin{bmatrix} m^j_k(t) \\ \frac{dm^j_k}{dt}(t) \end{bmatrix} + {}^{j}_{k}\mathbf{K}_{t;(a^j_{k-1},a^j_k)} \, {}^{j}_{k}\mathbf{K}^{-1}_{(a^j_{k-1},a^j_k);(a^j_{k-1},a^j_k)} \begin{bmatrix} z^j_{a^j_{k-1}} - m^j_k(a^j_{k-1}) \\ z^{j\prime}_{a^j_{k-1}} - \frac{dm^j_k}{dt}(a^j_{k-1}) \\ z^j_{a^j_k} - m^j_k(a^j_k) \\ z^{j\prime}_{a^j_k} - \frac{dm^j_k}{dt}(a^j_k) \end{bmatrix},$$

with

$${}^{j}_{k}\mathbf{K}_{(a^j_{k-1},a^j_k);(a^j_{k-1},a^j_k)} = \begin{bmatrix} {}^{j}_{k}\mathbf{K}_{a^j_{k-1};a^j_{k-1}} & {}^{j}_{k}\mathbf{K}_{a^j_{k-1};a^j_k} \\ {}^{j}_{k}\mathbf{K}_{a^j_k;a^j_{k-1}} & {}^{j}_{k}\mathbf{K}_{a^j_k;a^j_k} \end{bmatrix}$$

and

$${}^{j}_{k}\mathbf{K}_{t;(a^j_{k-1},a^j_k)} = \begin{bmatrix} {}^{j}_{k}\mathbf{K}_{t;a^j_{k-1}} & {}^{j}_{k}\mathbf{K}_{t;a^j_k} \end{bmatrix}.$$

Hence, using once again the law of total expectation, it follows that for any $t \in]a^j_{k-1}, a^j_k[$,

$$\boxed{\begin{bmatrix} \bar{m}^j(t) \\ \bar{m}^{j\prime}(t) \end{bmatrix} = \begin{bmatrix} m^j_k(t) \\ \frac{dm^j_k}{dt}(t) \end{bmatrix} + {}^{j}_{k}\mathbf{K}_{t;(a^j_{k-1},a^j_k)} \, {}^{j}_{k}\mathbf{K}^{-1}_{(a^j_{k-1},a^j_k);(a^j_{k-1},a^j_k)} \begin{bmatrix} \bar{m}^j(a^j_{k-1}) - m^j_k(a^j_{k-1}) \\ \bar{m}^{j\prime}(a^j_{k-1}) - \frac{dm^j_k}{dt}(a^j_{k-1}) \\ \bar{m}^j(a^j_k) - m^j_k(a^j_k) \\ \bar{m}^{j\prime}(a^j_k) - \frac{dm^j_k}{dt}(a^j_k) \end{bmatrix}.}$$

We note in particular that when $\forall\ j, k,\ m^j_k = 0$, it follows that $\bar{m}^j = 0, \bar{m}^f = 0, \bar{m}^{\nabla f} = 0$.

APPENDIX I.0.2 GLOBAL STRING GP COVARIANCE FUNCTIONS

As for the evaluation of ${}^{j}\bar{\mathbf{K}}_{u,v}$, we start by noting that the covariance function of a univariate *string GP* is the same as that of another *string GP* whose strings have the same unconditional kernels but unconditional mean functions $m^j_k = 0$, so that to evaluate univariate *string GP* kernels we may assume that $\forall\ j, k,\ m^j_k = 0$ without loss of generality. We start with the case where u and v are both boundary times, after which we will generalise to other times.

Boundary times: As previously discussed, the restriction $\left(z_t^j, z_t^{j\prime}\right)_{t\in[a_0^j,a_1^j]}$ is the *derivative Gaussian process* with mean 0 and covariance function k_1^j. Thus,

$$ {}^{j}\bar{\mathbf{K}}_{a_0^j;a_0^j} = {}^{j}_{1}\mathbf{K}_{a_0^j;a_0^j}, \quad {}^{j}\bar{\mathbf{K}}_{a_1^j;a_1^j} = {}^{j}_{1}\mathbf{K}_{a_1^j;a_1^j}, \quad {}^{j}\bar{\mathbf{K}}_{a_0^j;a_1^j} = {}^{j}_{1}\mathbf{K}_{a_0^j;a_1^j}. \tag{95} $$

We recall that conditional on the boundary conditions at or prior to a_{k-1}^j, $\begin{pmatrix} z^j_{a_k^j}, z^{j\prime}_{a_k^j} \end{pmatrix}$ is Gaussian with mean

$$ {}^{b}_{k}M \begin{bmatrix} z^j_{a_{k-1}^j} \\ z^{j\prime}_{a_{k-1}^j} \end{bmatrix} \quad \text{with} \quad {}^{b}_{k}M = {}^{j}_{k}\mathbf{K}_{a_k^j;a_{k-1}^j} \, {}^{j}_{k}\mathbf{K}^{-1}_{a_{k-1}^j;a_{k-1}^j}, $$

and covariance matrix

$$ {}^{b}_{k}\Sigma = {}^{j}_{k}\mathbf{K}_{a_k^j;a_k^j} - {}^{b}_{k}M \, {}^{j}_{k}\mathbf{K}_{a_{k-1}^j;a_k^j}. $$

Hence using Lemma 10 with $M = \begin{bmatrix} {}^{b}_{k}M & 0 & \dots & 0 \end{bmatrix}$ where there are $(k-1)$ null block 2×2 matrices, and noting that $\left(z^j_{a_0^j}, z^{j\prime}_{a_0^j}, \dots, z^j_{a_{k-1}^j}, z^{j\prime}_{a_{k-1}^j}\right)$ is jointly Gaussian, it follows that the vector $\left(z^j_{a_0^j}, z^{j\prime}_{a_0^j}, \dots, z^j_{a_k^j}, z^{j\prime}_{a_k^j}\right)$ is jointly Gaussian, that $\begin{pmatrix} z^j_{a_k^j}, z^{j\prime}_{a_k^j} \end{pmatrix}$ has covariance matrix

$$ \boxed{ {}^{j}\bar{\mathbf{K}}_{a_k^j;a_k^j} = {}^{b}_{k}\Sigma + {}^{b}_{k}M \, {}^{j}\bar{\mathbf{K}}_{a_{k-1}^j;a_{k-1}^j} \, {}^{b}_{k}M^T, } $$

and that the covariance matrix between the boundary conditions at a_k^j and at any earlier boundary time $a_l^j,\ l<k$ reads:

$$ \boxed{ {}^{j}\bar{\mathbf{K}}_{a_k^j;a_l^j} = {}^{b}_{k}M \, {}^{j}\bar{\mathbf{K}}_{a_{k-1}^j;a_l^j}. } $$

String times: Let $u\in[a_{p-1}^j, a_p^j],\ v\in[a_{q-1}^j, a_q^j]$. By the law of total expectation, we have that

$$ {}^{j}\bar{\mathbf{K}}_{u;v} := \mathrm{E}\left(\begin{bmatrix} z_u^j \\ z_u^{j\prime} \end{bmatrix} \begin{bmatrix} z_v^j & z_v^{j\prime} \end{bmatrix} \right) = \mathrm{E}\left(\mathrm{E}\left(\begin{bmatrix} z_u^j \\ z_u^{j\prime} \end{bmatrix} \begin{bmatrix} z_v^j & z_v^{j\prime} \end{bmatrix} \middle| \mathcal{B}(p,q) \right) \right), $$

where $\mathcal{B}(p,q)$ refers to the boundary condtions at the boundaries of the p-th and q-th strings, in other words $\left\{ z_x^j, z_x^{j\prime},\ x\in\left\{a_{p-1}^j, a_p^j, a_{q-1}^j, a_q^j\right\} \right\}$. Furthermore, using the definition of the covariance matrix under the conditional law, it follows that

$$ \mathrm{E}\left(\begin{bmatrix} z_u^j \\ z_u^{j\prime} \end{bmatrix} \begin{bmatrix} z_v^j & z_v^{j\prime} \end{bmatrix} \middle| \mathcal{B}(p,q) \right) = {}^{j}_{c}\bar{\mathbf{K}}_{u;v} + \mathrm{E}\left(\begin{bmatrix} z_u^j \\ z_u^{j\prime} \end{bmatrix} \middle| \mathcal{B}(p,q) \right) \mathrm{E}\left(\begin{bmatrix} z_v^j & z_v^{j\prime} \end{bmatrix} \middle| \mathcal{B}(p,q) \right), \tag{96} $$

where ${}^{j}_{c}\bar{\mathbf{K}}_{u;v}$ refers to the covariance matrix between $(z_u^j, z_u^{j\prime})$ and $(z_v^j, z_v^{j\prime})$ conditional on the boundary conditions $\mathcal{B}(p,q)$, and can be easily evaluated from Theorem 2. In particular,

$$ \text{if } p\neq q,\ {}^{j}_{c}\bar{\mathbf{K}}_{u;v} = 0, \text{ and if } p=q,\ {}^{j}_{c}\bar{\mathbf{K}}_{u;v} = {}^{j}_{p}\mathbf{K}_{u;v} - {}^{j}_{p}\Lambda_u \begin{bmatrix} {}^{j}_{p}\mathbf{K}^T_{v;a_{p-1}^j} \\ {}^{j}_{p}\mathbf{K}^T_{v;a_p^j} \end{bmatrix}, \tag{97} $$

where

$$\forall\, x, l,\ {}^{j}_{l}\Lambda_x = \begin{bmatrix} {}^{j}_{l}\mathbf{K}_{x;a^j_{l-1}} & {}^{j}_{l}\mathbf{K}_{x;a^j_l} \end{bmatrix} \begin{bmatrix} {}^{j}_{l}\mathbf{K}_{a^j_{l-1};a^j_{l-1}} & {}^{j}_{l}\mathbf{K}_{a^j_{l-1};a^j_l} \\ {}^{j}_{l}\mathbf{K}_{a^j_l;a^j_{l-1}} & {}^{j}_{l}\mathbf{K}_{a^j_l;a^j_l} \end{bmatrix}^{-1}.$$

We also note that

$$\mathrm{E}\left(\begin{bmatrix} z^j_u \\ z^{j\prime}_u \end{bmatrix} \middle| \mathcal{B}(p,q) \right) = {}^{j}_{p}\Lambda_u \begin{bmatrix} z^j_{a^j_{p-1}} \\ z^{j\prime}_{a^j_{p-1}} \\ z^j_{a^j_p} \\ z^{j\prime}_{a^j_p} \end{bmatrix} \text{ and } \mathrm{E}\left(\begin{bmatrix} z^j_v & z^{j\prime}_v \end{bmatrix} \middle| \mathcal{B}(p,q) \right) = \begin{bmatrix} z^j_{a^j_{q-1}} & z^{j\prime}_{a^j_{q-1}} & z^j_{a^j_q} & z^{j\prime}_{a^j_q} \end{bmatrix} {}^{j}_{q}\Lambda^T_v.$$

Hence, taking the expectation with respect to the boundary conditions on both sides of Equation (96), we obtain:

$$\forall\, u \in [a^j_{p-1}, a^j_p],\ v \in [a^j_{q-1}, a^j_q],\ \ {}^{j}\bar{\mathbf{K}}_{u;v} = {}^{j}_{c}\bar{\mathbf{K}}_{u;v} + {}^{j}_{p}\Lambda_u \begin{bmatrix} {}^{j}\bar{\mathbf{K}}_{a^j_{p-1};a^j_{q-1}} & {}^{j}\bar{\mathbf{K}}_{a^j_{p-1};a^j_q} \\ {}^{j}\bar{\mathbf{K}}_{a^j_p;a^j_{q-1}} & {}^{j}\bar{\mathbf{K}}_{a^j_p;a^j_q} \end{bmatrix} {}^{j}_{q}\Lambda^T_v,$$

where ${}^{j}_{c}\bar{\mathbf{K}}_{u;v}$ is provided in Equation (97).

References

R. P. Adams and O. Stegle. Gaussian process product models for nonparametric nonstationarity. In *International Conference on Machine Learning (ICML)*, pages 1–8, 2008.

R. J. Adler and J. E. Taylor. *Topological Complexity of Smooth Random Functions: École D'Été de Probabilités de Saint-Flour XXXIX-2009*. Lecture Notes in Mathematics / École d'Été de Probabilités de Saint-Flour. Springer, 2011.

P. Alquier, N. Friel, R. Everitt, and A. Boland. Noisy monte carlo: Convergence of Markov chains with approximate transition kernels. *Statistics and Computing*, 26(1-2):29–47, 2016.

R. Bardenet, A. Doucet, and C. Holmes. Towards scaling up Markov chain monte carlo: an adaptive subsampling approach. In *International Conference on Machine Learning (ICML)*, pages 405–413, 2014.

S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold Gaussian processes for regression. *arXiv preprint arXiv:1402.5876*, 2014.

Y. Cao and D. J. Fleet. Generalized product of experts for automatic and principled fusion of Gaussian process predictions. *arXiv preprint arXiv:1410.7827*, 2014.

D. J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes*. Springer-Verlag, 2008.

M. Deisenroth and J. W. Ng. Distributed Gaussian processes. In *International Conference on Machine Learning (ICML)*, pages 1481–1490, 2015.

J. L. Doob. The elementary Gaussian processes. *The Annals of Mathematical Statistics*, 15(3): 229–282, 1944.

N. Durrande, D. Ginsbourger, and O. Roustant. Additive covariance kernels for high-dimensional Gaussian process modeling. *Annales de la Faculté de Sciences de Toulouse*, 21(3), 2012.

D. Duvenaud, H. Nickisch, and C. E. Rasmussen. Additive Gaussian processes. In *Advances in Neural Information Processing Systems (NIPS)*, pages 226–234, 2011.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, pages 209–230, 1973.

N. J. Foti and S. Williamson. A survey of non-exchangeable priors for Bayesian nonparametric models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):359–371, 2015.

R. B. Gramacy and H. K. H. Lee. Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483), 2008.

P. J. Green. Reversible jump Markov chain monte carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

P. J. Green and D. I. Hastie. Reversible jump MCMC. *Genetics*, 155(3):1391–1403, 2009.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intellegence (UAI)*, pages 282–290, 2013.

J. Hensman, A. Matthews, and Z. Ghahramani. Scalable variational Gaussian process classification. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 351–360, 2015.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

I. Karatzas and R. Fernholz. Stochastic Portfolio Ttheory: An Overview. *Handbook of Numerical Analysis*, 15:89–167, 2009.

H. Kim, Mallick B. K., and Holmes C. C. Analyzing nonstationary spatial data using piecewise Gaussian processes. *Journal of the American Statistical Association*, 100(470):653–668, 2005.

J. Kingman. Completely random measures. *Pacific Journal of Mathematics*, 21(1):59–78, 1967.

Y.-L. Kom Samo and A. Vervuurt. Stochastic portfolio theory : a machine learning perspective. In *Uncertainty in Artificial Intelligence (UAI)*, pages 657– 665, 2016.

N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: the informative vector machine. In *Advances in Neural Information Processing Systems (NIPS)*, pages 625–632, 2003.

M. Lazaro-Gredilla, J. Quinonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vida. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11:1866–1881, 2010.

Q. Le, T. Sarlós, and A. Smola. Fastfood-approximating kernel expansions in loglinear time. In *International Conference on Machine Learning (ICML)*, 2013.

I. G. Macdonald. *Symmetric Functions and Hall Polynomials*. Oxford University Press, 1995.

D. J. C. MacKay. Introduction to Gaussian processes. In *NATO ASI Series F: Computer and Systems Sciences*, pages 133–166. Springer, Berlin, 1998.

E. Meeds and S. Osindero. An alternative infinite mixture of Gaussian process experts. In *Advances In Neural Information Processing Systems (NIPS)*, 2006.

I. Murray, R. P. Adams, and D. J. C. MacKay. Elliptical slice sampling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 9–16, 2010.

R. Neal. *Bayesian learning for neural networks*. Lecture notes in Statistics. Springer, 1996.

T. Nguyen and E. Bonilla. Fast allocation of Gaussian process experts. In *International Conference on Machine Learning (ICML)*, pages 145–153, 2014.

B. Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Hochschultext / Universitext. Springer, 2003.

C. Paciorek and M. Schervish. Nonstationary covariance functions for Gaussian process regression. In *Advances in Neural Information Processing Systems (NIPS)*, pages 273–280, 2004.

J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, pages 855–900, 1997.

C. Plagemann, K. Kersting, and W. Burgard. Nonstationary Gaussian process regression using point estimate of local smoothness. In *European Conference on Machine Learning (ECML)*, pages 204–219, 2008.

J. Quinonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1177–1184, 2007.

C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems (NIPS)*, pages 881–888, 2001.

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

J. Ross and J. Dy. Nonparametric mixture of Gaussian processes with constraints. In *International Conference on Machine Learning (ICML)*, pages 1346–1354, 2013.

Y. Saatchi. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2011.

A. M. Schmidt and A. O'Hagan. Bayesian inference for nonstationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(3):743–758, 2003.

M. Seeger. Bayesian Gaussian process models: Pac-Bayesian generalisation error bounds and sparse approximations. Technical report, 2003a.

M. Seeger. PAC-Bayesian generalisation error bounds for Gaussian process classification. *Journal of Machine Learning Research*, 3:233–269, 2003b.

A. Shah, A. G. Wilson, and Z. Ghahramani. Student-t processes as alternatives to Gaussian processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 877–885, 2014.

B. W. Silverman. Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 47(1):1–52, 1985.

A. J. Smola and P. Bartlett. Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems (NIPS)*, pages 619–625. MIT Press, 2001.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1257–1264, 2006.

The GPy authors. GPy: A Gaussian process framework in python. http://github.com/SheffieldML/GPy, 2012–2016.

V. Tresp. A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741, 2000.

V. Tresp. Mixtures of Gaussian processes. In *Advances in Neural Information Processing Systems (NIPS)*, pages 654–660, 2001.

A. Vervuurt and I. Karatzas. Diversity-weighted portfolios with negative parameter. *Annals of Finance*, 11(3):411–432, 2015.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 682–688, 2001.

A. G. Wilson and R. P. Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International Conference on Machine Learning (ICML)*, pages 1067–1075, 2013.

A. G. Wilson and H. Nickisch. Kernel interpolation for scalable structured Gaussian processes. In *International Conference on Machine Learning (ICML)*, pages 1775–1784, 2015.

A. G. Wilson, E. Gilboa, and J. P. Nehorai, A.and Cunningham. Fast kernel learning for multidimensional pattern extrapolation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3626–3634. 2014.

Z. Yang, A. Smola, L. Song, and A. G. Wilson. A la carte – learning fast kernels. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1098–1106, 2015.