Nonparametric Gaussian Process Covariances via Multidimensional Convolutions

Thomas M. McDonald[†]
University of Manchester

Magnus Ross[†]
University of Manchester

Michael T. Smith
University of Sheffield

Mauricio A. Álvarez
University of Manchester

Abstract

A key challenge in the practical application of Gaussian processes (GPs) is selecting a proper covariance function. The process convolutions construction of GPs allows some additional flexibility, but still requires choosing a proper smoothing kernel, which is non-trivial. Previous approaches have built covariance functions by using GP priors over the smoothing kernel, and by extension the covariance, as a way to bypass the need to specify it in advance. However, these models have been limited in several ways: they are restricted to single dimensional inputs, e.g. time; they only allow modelling of single outputs and they do not scale to large datasets since inference is not straightforward. In this paper, we introduce a nonparametric process convolution formulation for GPs that alleviates these weaknesses. We achieve this using a functional sampling approach based on Matheron's rule to perform fast sampling using interdomain inducing variables. We test the performance of our model on benchmarks for single output, multi-output and large-scale GP regression, and find that our approach can provide improvements over standard GP models, particularly for larger datasets.

1 INTRODUCTION

Gaussian processes (GPs) are a widely used method for probabilistic machine learning, that have been applied successfully in many areas (Shafieloo et al., 2012; Kong et al., 2018; Richardson et al., 2018). A central problem when modelling data with GPs is the choice of a covariance function. The covariance function controls the properties of the functions that the GP places high probability over, therefore selection of an appropriate covariance is crucially important to achieving success when modelling with GPs. When working with a single dimensional input, most commonly in the time series setting, practitioners can inspect the data to determine patterns such as periodicity, long term trends and so on, and construct an appropriate covariance by combining simpler covariance functions that account for these patterns. However this procedure becomes very difficult in higher dimensions, where it is not easy to determine which covariances should be used by simply inspecting the data. Because of this, in high dimensions, practitioners typically revert to using simple covariances, most commonly the exponentiated quadratic (EQ) or Matérn class of kernels. The difficulty of covariance design in high dimensions means that the ability of GPs to represent rich structures present in the data via the covariance function is often not fully utilised. In this work, we present a model that can be applied to problems with both multiple inputs, multiple outputs (or tasks), and can infer the form of the covariance in a nonparametric fashion.

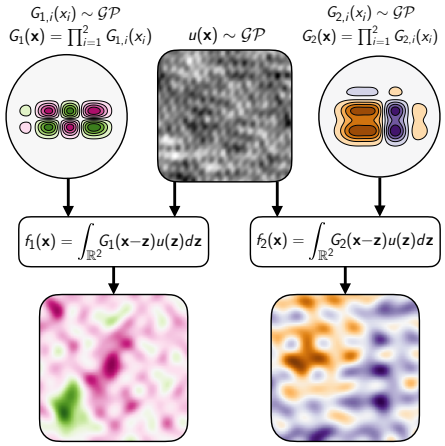


Figure 1: Sampling for the proposed model, showing separate convolutional kernels constructed using GPs (top left/right), being convolved with a shared latent GP (top center), to produce two distinct outputs with nonparametric covariances (bottom).

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

[†]Equal contribution.

In order to build such a model, we employ the framework of process convolutions (PCs) (Higdon, 2002; Álvarez et al., 2012), in which a base Gaussian process is convolved with a smoothing kernel to generate another Gaussian process with a modified covariance. The PC framework can be leveraged to infer covariances nonparametrically, by placing a GP prior over the smoothing kernel. Tobar et al. (2015) introduced the Gaussian process convolution model (GPCM), which uses this mechanism to construct a GP with a nonparametric covariance for data with a single input and output dimension. Recently, Ross et al. (2021) extended the GPCM to nonlinear process convolutions, and applied the model to problems in systems identification. In this work, we extend and generalise the GPCM to both multiple input and output dimensions, and provide a scalable inference scheme for the model.

Figure 1 illustrates the proposed generative model for the case of two input dimensions and two outputs. A shared latent function is sampled from a GP with a short lengthscale, which is then smoothed by two distinct convolutional kernels which are generated by a product of GPs, separable over the input dimension, to produce two correlated GP outputs. These distinct convolutional kernels induce different covariance properties in each output, whilst correlations between the outputs across the domain are captured by the shared latent input function. We refer to our method as the nonparametric convolved Gaussian process (NP-CGP). It can be applied with an arbitrary number of inputs, outputs and latent functions.

Many approaches have been proposed to allow for inference of expressive covariance functions. Besides the PC framework, a fruitful avenue has been the design of covariances in the spectral domain. Spectral mixture kernels (Wilson and Adams, 2013; Wilson et al., 2014; Jang et al., 2017) model the covariance as a mixture of Gaussians in frequency space, and have been extended to multiple outputs (Parra and Tobar, 2017), and nonstationary processes (Altamirano and Tobar, 2022; Shen et al., 2019). Additionally, nonparametric GP priors can also be placed on the spectral density (Benton et al., 2019). Expressive nonstationary covariances can also be constructed by warping the input space with deep neural networks (Wilson et al., 2016), by the application of stochastic differential equations Hegde et al. (2019), or, as in the case of deep GPs (Damianou and Lawrence, 2013; Blomqvist et al., 2019; Salimbeni and Deisenroth, 2017; McDonald and Alvarez, 2021), by warping the space with a composition of GPs.

In this work we present the following three contributions: **1)** A generalisation of the GPCM to the case of multidimensional inputs and outputs with a scalable inference scheme to allow for the use of large datasets. **2)** An extension to the functional sampling method of Wilson et al. (2020) to cases where the inducing points lie in a transformed space relative to the samples. **3)** A fast sampling procedure for the model in high dimensions based on the exact integration of the aforementioned approximate functional samples.

2 BACKGROUND

This section briefly reviews the theory behind process convolutions, the GPCM and sampling with pathwise updates.

2.1 Process convolutions

The PC framework allows expressive covariances for GPs to be constructed, and can be used to automatically learn the form of covariances from data. In the PC framework (Barry and Ver Hoef, 1996; Higdon, 2002; Álvarez et al., 2012), the function which we wish to model is assumed to have been generated by applying some linear convolution operator to a latent function u represented by a GP, which outputs a new GP (Rasmussen and Williams, 2005). The PC framework can be used to construct multiple output GPs (MOGPs), which allow for inference over vector functions $\mathbf{f} : \mathbb{R}^P \mapsto \mathbb{R}^D$, where P is the number of input dimensions and D is the number of output dimensions. This is useful when we have a set of outputs, represented by the elements of $\mathbf{f}$, which we know are correlated in some way, but also exhibit independent variation. We can construct MOGPs using PCs by assuming each output, f_d is generated by convolving an independent convolutional kernel G_d with a shared latent process u, so $f_d(\mathbf{x}) = \int_{\mathcal{X}} G_d(\mathbf{x} - \boldsymbol{\tau})u(\boldsymbol{\tau})d\boldsymbol{\tau}$, where $\mathcal{X}$ is the domain of integration. For many applications it is often overly restrictive to assume that the shared variations can be encapsulated by a single function, and so we instead can use a set of functions $\mathbf{u} : \mathbb{R}^P \mapsto \mathbb{R}^Q$, with each being transformed in a different way for each output. We can express this as $f_d(\mathbf{x}) = \sum_{q=1}^{Q} \int_{\mathcal{X}} G_{d,q}(\mathbf{x} - \boldsymbol{\tau})u_q(\boldsymbol{\tau})d\boldsymbol{\tau}$. This is the most general form of the model and can be written in the more succinct form,

$$\begin{aligned} \mathbf{f}(\mathbf{x}) &= \int_{\mathcal{X}} \mathbf{G}(\mathbf{x} - \mathbf{z})\mathbf{u}(\mathbf{z})d\mathbf{z} \implies \\ \mathbf{k}_{\mathbf{f}}(\mathbf{x}, \mathbf{x}') &= \int_{\mathcal{X}} \mathbf{G}(\mathbf{x} - \boldsymbol{\tau})\mathbf{k}_{\mathbf{u}}(\boldsymbol{\tau}, \boldsymbol{\tau}')\mathbf{G}^{\top}(\mathbf{x}' - \boldsymbol{\tau}')d\boldsymbol{\tau}d\boldsymbol{\tau}', \end{aligned} \tag{1}$$

where $\mathbf{G} : \mathbb{R}^P \mapsto \mathbb{R}^{D\times Q}$ consists of square integrable elements, $\mathbf{k}_{\mathbf{f}} : \mathbb{R}^P \times \mathbb{R}^P \mapsto \mathbb{R}^{D\times D}$ is the matrix-valued covariance for the output, and $\mathbf{k}_{\mathbf{u}} : \mathbb{R}^P \times \mathbb{R}^P \mapsto \mathbb{R}^{Q\times Q}$ is the matrix-valued covariance for the shared inputs, which is diagonal due to the assumed independence of the functions representing the elements of $\mathbf{u}$. Eq. (1) produces functions with stationary covariance; a nonstationary covariance can be obtained by using a convolutional kernel that varies over the input domain, but this will not be considered in the present work. Various properties of interest can be embedded in $\mathbf{f}$ via $\mathbf{G}$, for example the properties of differ-

ent physical systems, by using the Green's function of a differential operator Alvarez et al. (2009).

2.2 Gaussian process convolution models

The GPCM uses the form of Eq. (1) in the case $P, D, Q = 1$, and places a GP prior over the convolutional kernel, which in turn induces a prior over the covariance function of the output. To ensure the output is finite, the authors introduce the decaying square exponential (DSE) covariance, which consists of a regular EQ covariance with an additional window, which ensures that the samples decay to zero away from the origin. For the input process, a white noise covariance is used, with the process being summarised by a set of interdomain inducing points, where the interdomain transform is a Gaussian convolution. The use of white noise for the input process is motivated by the fact that the lengthscale of the output process is bounded from below by the lengthscale of the input process. The white noise input process informally has a lengthscale of zero, therefore by using it, no restriction is placed on the lengthscale of the output. Since the white noise has zero lengthscale, it cannot be summarised by a finite number of inducing points, which necessitates the interdomain transform. For inference, the GPCM uses a classical mean field variational inference scheme. Bruinsma et al. (2022) introduce a number of improvements to both the model structure and inference in the GPCM, particularly extending the model to non-smooth time series using a causal convolution operator, as well as forming a structured variational inference scheme which drastically improves the accuracy and speed of inference. A generalisation of the GPCM to multiple inputs and outputs has previously been discussed by Bruinsma (2016), who coined the name generalised GPCM (GGPCM) to refer to the model. However inference in the model was never implemented, and as such the model was not applied to any data. Ross et al. (2021) propose the nonparametric Volterra kernels model (NVKM), an extension of the GPCM to nonlinear convolution operators and multiple outputs, which employs doubly stochastic variational inference (DSVI) for approximate inference, and can be used for systems identification, but does not use the interdomain transform for the input process.

2.3 Sampling GP functions

Wilson et al. (2020) present a method based on Matheron's rule (Journel and Huijbregts, 1976), which allows for the efficient sampling of approximate functions from the posterior of a GP with a stationary covariance. Sampling functions enables samples from the GP at N locations to be evaluated in $O(N)$ time, as opposed to the $O(N^3)$ of standard GP sampling, a significant improvement for applications which require the evaluation of samples at many locations. An additional benefit of sampling *functions* from GPs is that different operators, including integral and differential operators, can be applied to the samples themselves, allowing for the generation of samples from (possibly) highly complex non-Gaussian processes to be obtained efficiently. This idea was used by Ross et al. (2021) in the context of the NVKM to generate samples from the output of a nonlinear process convolution. Wilson et al. (2020) present Matheron's rule in the context of samples from a GP posterior given inducing variables (or data) $\mathbf{u}$ as

$$\underbrace{(f \mid \mathbf{u})(\cdot)}_{\text{posterior}} \overset{\text{d}}{=} \underbrace{f(\cdot)}_{\text{prior}} + \underbrace{k(\cdot, \mathbf{Z})\mathbf{K}^{-1}(\mathbf{u} - \mathbf{f})}_{\text{update}}, \tag{2}$$

where $\mathbf{K}$ is the covariance matrix of the inducing variables with inputs $\mathbf{Z}$, and $\mathbf{f} = f(\mathbf{Z})$. This expression shows that a functional sample from the posterior, conditioned on data (or inducing outputs) $\mathbf{u}$, can be decomposed into functional samples from the prior, and an update term which accounts for the residual between the prior sample and the data. The key innovation introduced by Wilson et al. (2020) is that we can represent $f(\cdot)$ using random Fourier features (RFFs) (Rahimi and Recht, 2007). Since only the prior, which uses a stationary covariance, uses RFFs, the pathologies associated with the use of RFFs in the nonstationary posterior can be avoided, while still retaining the computational benefits they provide.

3 NONPARAMETRIC CONVOLUTIONS FOR GAUSSIAN PROCESSES

In this section, we present a generalised PC model of the form shown in Eq. (1) which jointly infers vector-valued functions $\mathbf{f} : \mathbb{R}^P \mapsto \mathbb{R}^D$, and their corresponding nonparametric convolutional kernel $\mathbf{G}$. This induces a nonparametric matrix-valued covariance, $\mathbf{k}_{\mathbf{f}}$, for $\mathbf{f}$.

3.1 Single-dimensional inputs

Before presenting the multi-dimensional version of our model, we start with the single-dimensional input case ($P = 1$) in Eq. (1), given as $\mathbf{f}(x) = \int_{\mathbb{R}} \mathbf{G}(x - z)\mathbf{u}(z)dz$, where each entry in the vector $\mathbf{u}(x)$ follows a GP, i.e. $u_q(x) \sim \mathcal{GP}[0, k_{u_q}(x, x')], q = 1, \ldots, Q$; and each entry in the matrix $\mathbf{G}(x)$ also follows a GP, i.e. $G_{d,q}(x) \sim \mathcal{GP}[0, k_{G_{d,q}}(x, x')], q = 1, \ldots, Q$ and $d = 1, \ldots, D$. Throughout this work, we use the DSE covariance, described by Tobar et al. (2015), for the elements of $\mathbf{G}$ to ensure that they are square integrable. The DSE covariance is given by $k_{DSE}(x, x') = \sigma^2 \exp(-\alpha(x^2 + x'^2) - \gamma(x - x')^2)$. Around the origin the DSE behaves similarly to the SE, however as the input grows the covariance, and therefore sample functions, decay to 0. This model can be seen as a generalisation of the GPCM, which can only represent functions from $\mathbb{R} \mapsto \mathbb{R}$, to the multi-output case. Both exact sampling and exact inference in the NP-CGP model above are intractable as the integral cannot be computed when $G_{d,q}(\cdot)$ and $u_q(\cdot)$ are infinite dimensional stochastic

processes. In order to draw samples from the model, we must first summarise the GPs representing the convolutional kernels and the input processes with finite collections of inducing points. These inducing points can then be used to sample approximate functions from the convolutional kernels and input processes, which can be integrated exactly to produce samples from the output. Fast and accurate sampling allows a doubly stochastic variational inference scheme to be constructed for the model, which is discussed later in this section.

3.2 Multi-dimensional inputs

To extend the model above to the multi-dimensional input case, we need to perform inference on the model $\mathbf{f}(\mathbf{x}) = \int_{\mathbb{R}^P} \mathbf{G}(\mathbf{x}-\mathbf{z})\mathbf{u}(\mathbf{z})d\mathbf{z}$. Following the same construction used for the single-dimensional input case, we place GP priors over the inputs of $\mathbf{u}(\mathbf{x})$, i.e. $u_q(\mathbf{x}) \sim \mathcal{GP}[0, k_{u_q}(\mathbf{x}, \mathbf{x}')], q = 1, \ldots, Q$. In a similar way, one could place GP priors over the individual elements of $\mathbf{G}$, i.e. $G_{d,q}(\mathbf{x}) \sim \mathcal{GP}[0, k_{G_{d,q}}(\mathbf{x}, \mathbf{x}')]$. Given that our inference approach is based on inducing points, this option of GP priors for $G_{d,q}(\mathbf{x})$ is computationally intractable in high input dimensions. The number of inducing points required to characterise $G_{d,q}$ increases exponentially with the number of input dimensions, as the inducing points become increasingly sparsely distributed across the input space in higher dimensions. This is a problem for the convolutional kernels in particular because they operate over the whole domain, so any uncertainty in their value translates across the entire output function.

We can address this problem by modelling $G_{d,q}$ as product separable, such that $G_{d,q}(\mathbf{x}) = \prod_{p=1}^{P} G_{d,q}^{(p)}(x_p)$, where each $G_{d,q}^{(p)}$ is an independent GP with its own set of inducing points, and x_p is the p-th dimension of the input. This assumption allows us to characterise $G_{d,q}$ using a set of inducing points whose size scales linearly with the number of input dimensions, which is a considerable improvement. Furthermore, if we were to assume that each degree of freedom for $G_{d,q}^{(p)}(x_p)$ were to be modelled as a GP, we would need DQP independent GPs to model all the elements in $\mathbf{G}$. Therefore, to reduce the number of GPs used, we further assume that each smoothing kernel $G_{d,q}(\mathbf{x})$ can be expressed as $G_{d,q}(\mathbf{x}) = a_q G_d(\mathbf{x})$, where $a_q \in \mathbb{R}$, reducing the number of GPs to model $\mathbf{G}$ to DP. The generative model is given as

$$
\begin{aligned}
u_q(\mathbf{x}) &\sim \mathcal{GP}[0, k_{u_q}(\mathbf{x}, \mathbf{x}')], \\
G_d^{(p)}(x_p) &\sim \mathcal{GP}[0, k_{G_d^{(p)}}(x_p, x_p')], \\
G_{d,q}(\mathbf{x}) = a_q G_d(\mathbf{x}), &\quad G_d(\mathbf{x}) = \prod_{p=1}^{P} G_d^{(p)}(x_p), \\
\mathbf{f}(\mathbf{x}) &= \int_{\mathbb{R}^P} \mathbf{G}(\mathbf{x}-\mathbf{z})\mathbf{u}(\mathbf{z})d\mathbf{z},
\end{aligned}
\tag{3}
$$

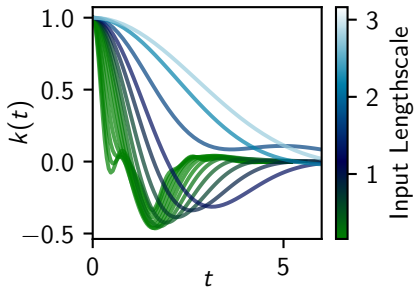


Figure 2: Samples of output covariance, shaded by input process lengthscale.

with $q = 1, \ldots, Q$, $p = 1, \ldots, P$ and $d = 1, \ldots, D$. We refer to the model in Eq. (3) as the nonparametric convolved Gaussian process (NP-CGP). Notice that $G_{d,q}(\mathbf{x})$ is not a GP, though the output process conditioned on $G_{d,q}(\mathbf{x})$ will still be a GP as the convolution is a linear operator on the input process. The separable restriction on $G_{d,q}(\mathbf{x})$ corresponds to the restriction that the covariance function is a product of covariances for each input, which is true for most popular multi-dimensional covariances, such as the *automatic relevance determination* (ARD) kernel. Whilst specifying a product separable convolutional kernel results in a computationally feasible form of the model, we also present a more efficient variant, which we refer to as the *Fast NP-CGP* (FNP-CGP). In this case, rather than using separate convolutional kernels $G_d^{(p)}$ per input and output dimension (resulting in a total of DP kernels), we share a single convolutional kernel across $d = 1, ..., D$, with each being convolved with a different linear combination of input functions, i.e. $G_{d,q}(\mathbf{x}) = a_{d,q} \prod_{p=1}^{P} G^{(p)}(x_p)$.

3.3 Interdomain input processes

When considering this model, it may seem as though we are free to choose any covariance we please for the input process, and place inducing points in the same space. Indeed, Ross et al. (2021) take this approach for a model with a similar structure. However, for the case of high dimensional inputs, this approach is no longer suitable. As mentioned previously, in high dimensions it is not computationally feasible to use a number of inducing points that will densely cover the space, and as such the lengthscale of the process the inducing points are placed upon must become large. In PC models that use a smoothing transform, such as the GPCM, the lengthscale of the output is, in a sense, bounded from below by the lengthscale of the input process, as the smoothing transform can never increase its lengthscale. This presents an issue for the NP-CGP, because the necessity of a long lengthscale input process in high dimensions would make it difficult to learn expressive covariances. Figure 2 shows a covariance sample from the output smoothly inter-

polated between a variety of input lengthscales, with all else remaining the same. We can see that as the input lengthscale increases, the complex structures in the output process covariance disappear, and the covariance tends to that of the input. One way to address this problem is to use the framework of interdomain inducing points (Lázaro-Gredilla and Figueiras-Vidal, 2009; Álvarez et al., 2010), using a short lengthscale for the input process to allow the covariances of the output to remain expressive, and placing inducing points within a smoothed domain. This allows us to retain the ability to summarise the variation of the process over long lengthscales, as is necessary in high dimensions.

3.4 Sampling from the outputs

Inference in the NP-CGP is predicated upon the ability to generate samples from the model outputs efficiently. To achieve this, we utilise the method of Wilson et al. (2020) to produce function samples from the input process, $\mathbf{u}^{(s)}(\mathbf{x})$ and convolutional kernel process $\mathbf{G}^{(s)}(\mathbf{x})$, which we then analytically integrate through the convolution integral, to obtain the output as $\mathbf{f}^{(s)}(\mathbf{x}) = \int_{\mathbb{R}^P} \mathbf{G}^{(s)}(\mathbf{x}-\mathbf{z})\mathbf{u}^{(s)}(\mathbf{z})d\mathbf{z}$. Although the computation is somewhat involved, closed form solutions to this integral can be obtained. Further details regarding this computation are available in the supplemental material.

3.5 Sampling interdomain functions

As discussed above, it is necessary to use the framework of interdomain inducing points for the functions u_q, to maintain the ability to represent expressive covariances in high dimensions. Sampling and inference in our model also requires access to functional samples for the functions u_q, and as such we must combine the two methods. We represent the elements of our input process using an EQ covariance with a short lengthscale, with the interdomain process $\tilde{\mathbf{u}}$ being generated by a smoothing transformation with a Gaussian window, $\tilde{u}_q(\mathbf{x}) = \int_{\mathbb{R}^P} g_q(\mathbf{z},\mathbf{x})u_q(\mathbf{z})d\mathbf{z}$, with $g_q(\mathbf{z},\mathbf{x}) = a_q e^{-\sum_p \alpha_{q,p}(x_p - z_p)^2}$, where $\alpha_{q,p}$ is related to the lengthscale of the transformation. We can consider the process $\tilde{\mathbf{u}}$ as a version of the process $\mathbf{u}$ with the high frequencies removed. Because the interdomain process is smoothed, inducing points can be used to summarise it, since the process now has correlations over the domain.

Matheron's rule, which the method of Wilson et al. (2020) relies upon, applies for collections of jointly distributed Gaussian variables, and as such, can be readily adapted to the interdomain case, because the processes u_q and $\tilde{u}_q$ are jointly Gaussian. The expression for a functional sample from u_q in the interdomain case now becomes

$$u_q^{(s)}(\cdot) = \sum_{i=1}^{B} w_i \phi_i(\cdot) + k_{u_q,\tilde{u}_q}(\cdot, \mathbf{z}^{\tilde{u}_q})\mathbf{K}_{\tilde{u}_q,\tilde{u}_q}^{-1}\left(\mathbf{v}^{\tilde{u}_q} - \tilde{\mathbf{\Phi}}\boldsymbol{w}\right), \tag{4}$$

where ϕ_i is one of B RFF basis functions with random weights $w_i \sim \mathcal{N}(0,1)$, $k_{u_q,\tilde{u}_q}$ represents the cross-covariance between domains, $\mathbf{z}^{\tilde{u}_q} \in \mathbb{R}^{M_u \times P}$ is the set of M_u inducing inputs with corresponding outputs $\mathbf{v}^{\tilde{u}_q} \in \mathbb{R}^{M_u \times 1}$, $\mathbf{K}_{\tilde{u}_q,\tilde{u}_q}$ is the covariance of the inducing points in the transformed domain, and $\tilde{\mathbf{\Phi}} \in \mathbb{R}^{M_u \times B}$ is a matrix with each of the basis functions evaluated in the transformed domain for each inducing input. These transformed basis functions can be computed by applying the interdomain convolution, such that $\tilde{\phi}_i(\mathbf{x}) = \int_{\mathbb{R}^P} g_q(\mathbf{z},\mathbf{x})\phi_i(\mathbf{z})d\mathbf{z}$. For the Gaussian transform discussed above, we can obtain expressions for transformed basis functions in closed form, with details of the computation included in the supplemental material, alongside further information regarding the RFF basis, the derivation of (4) and the computation of the various interdomain and cross-covariances. To the best of our knowledge, this method for fast sampling of interdomain GPs is yet to appear in the literature.

3.6 Doubly stochastic variational inference

Following the approach of Salimbeni and Deisenroth (2017), we employ DSVI to perform approximate inference in the the NP-CGP and FNP-CGP. Firstly, we must introduce the inducing points for the convolutional kernel processes, $\mathbf{v}^{G_{d,q}} \in \mathbb{R}^{M_G \times 1}$, with entries $v_i^{G_{d,q}} = G_{d,q}(z_i^{G_{d,q}}), i = 1,\ldots,M_G$, where M_G is the number of inducing points used. The associated inducing inputs are denoted as $z_i^{G_{d,q}}, i = 1,\ldots,M_G$, which we collect into $\mathbf{z}^{G_{d,q}} \in \mathbb{R}^{M_G \times 1}$. Additionally, to simplify the notation, we collect all of the inducing points for the convolutional kernel and input processes into $\mathbf{V}^G = \{\mathbf{v}^{G_{d,q}}\}_{d,q=1}^{D,Q}$ and $\mathbf{V}^{\tilde{u}} = \{\mathbf{v}^{\tilde{u}_q}\}_{q=1}^{Q}$ respectively. If we consider some input data $\mathbf{X} \in \mathbb{R}^{N \times P}$ with corresponding outputs $\mathbf{Y} \in \mathbb{R}^{N \times D}$, we can express the joint distribution of the NP-CGP as,

$$p(\mathbf{Y},\mathbf{G},\mathbf{V}^{\mathbf{G}},\mathbf{u},\mathbf{V}^{\tilde{\mathbf{u}}}) = \prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{F}_i) \times p(\mathbf{G}|\mathbf{V}^{\mathbf{G}})p(\mathbf{V}^{\mathbf{G}})p(\mathbf{u}|\mathbf{V}^{\tilde{\mathbf{u}}})p(\mathbf{V}^{\tilde{\mathbf{u}}}), \tag{5}$$

where the likelihood is given by $p(\mathbf{y}_i|\mathbf{F}_i) = \mathcal{N}(\mathbf{y}_i; \mathbf{F}_i, \sigma_Y^2)$ and $\mathbf{F}_i = \mathbf{f}(\mathbf{X}_i)$ represents the output of the model for the i-th input. As all of the convolutional kernel and input GPs are independent, we have $p(\mathbf{G}|\mathbf{V}^{\mathbf{G}}) = \prod_{d=1}^{D} p(G_d|\mathbf{v}^{G_d})$ where $p(G_d|\mathbf{v}^{G_d})$ is the GP posterior distribution given the inducing points, and likewise $p(\mathbf{u}|\mathbf{V}^{\tilde{\mathbf{u}}}) = \prod_{q=1}^{Q} p(u_q|\mathbf{v}^{\tilde{u}_q})$, where again $p(u_q|\mathbf{v}^{\tilde{u}_q})$ are GP posteriors. $p(\mathbf{V}^{\mathbf{G}})$ and

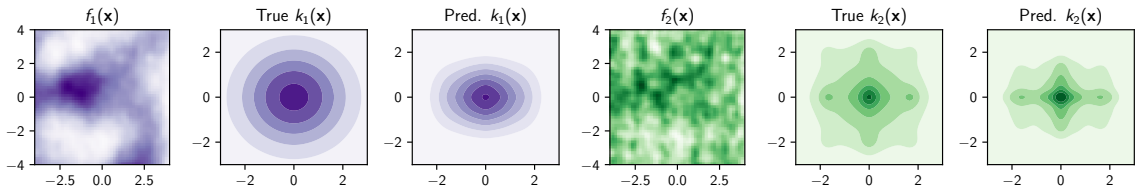


Figure 3: Inferred covariance for the toy experiment for each output, showing true target function over the training region (left plot), true data generating covariance (central plot) and inferred covariances (right plot). For the predictions, we show the mean function, with the uncertainty here being small.

$p(\mathbf{V}^{\tilde{\mathbf{u}}})$ represent the priors over the inducing points. Following the approach of Tobar et al. (2015), we employ a mean-field variational posterior, which takes the form,

$$q(\mathbf{G}, \mathbf{V}^{\mathbf{G}}, \mathbf{u}, \mathbf{V}^{\tilde{\mathbf{u}}}) = p(\mathbf{G}|\mathbf{V}^{\mathbf{G}})q(\mathbf{V}^{\mathbf{G}})p(\mathbf{u}|\mathbf{V}^{\tilde{\mathbf{u}}})q(\mathbf{V}^{\tilde{\mathbf{u}}}), \tag{6}$$

where $q(\mathbf{V}^{\mathbf{G}}) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{v}^{G_d}; \boldsymbol{\mu}^{G_d}, \boldsymbol{\Sigma}^{G_d})$ and $q(\mathbf{V}^{\tilde{\mathbf{u}}}) = \prod_{q=1}^{Q} \mathcal{N}(\mathbf{v}^{\tilde{u}_q}; \boldsymbol{\mu}^{\tilde{u}_q}, \boldsymbol{\Sigma}^{\tilde{u}_q})$ are variational distributions, whose means and covariance matrices are variational parameters. We use samples from both of these variational distributions in the process of analytically computing the functional samples from our model. For ease of exposition, we have omitted the factorisation of the posterior over the input dimensionality. Using $s = 1, ..., S$ samples from the model, denoted by $\mathbf{F}_i^{(s)} = \mathbf{f}^{(s)}(\mathbf{X}_i)$, and representing the KL divergence as $\mathrm{KL}[\cdot\|\cdot]$, we can approximate the variational lower bound as,

$$\begin{aligned}\mathcal{L} =& \frac{1}{S}\sum_{s=1}^{S} \log p(\mathbf{y}_i|\mathbf{F}_i^{(s)}) \\ &- \mathrm{KL}[q(\mathbf{V}^{\tilde{\mathbf{u}}})\|p(\mathbf{V}^{\tilde{\mathbf{u}}})] - \mathrm{KL}[q(\mathbf{V}^{\mathbf{G}})\|p(\mathbf{V}^{\mathbf{G}})].\end{aligned} \tag{7}$$

A discussion of the complexities and practical run times of the models is given in the supplemental material, along with an extended derivation of the bound above.

4 RELATED WORK

Ross et al. (2021) propose the *nonparametric Volterra kernels model* (NVKM), an extension of the GPCM to nonlinear systems and multiple outputs, which employs DSVI for approximate inference. The NP-CGP can be seen as extension of this model to multiple input dimensions, however unlike this work and that of Tobar et al. (2015), the authors do not use interdomain inducing points for the u process. Additionally, unlike Tobar et al. (2015), the authors use an EQ covariance for the u process, rather than a Dirac delta covariance. The NVKM does exploit the same efficient sampling scheme as our model, based on pathwise updates, as discussed in Section 2.3.

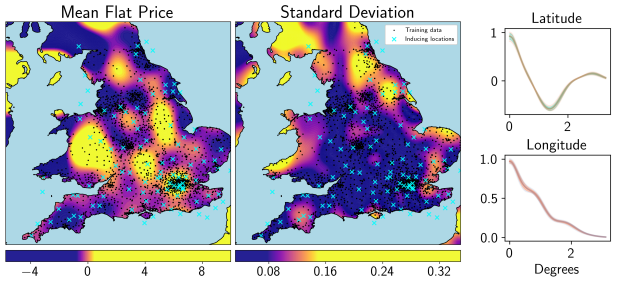
Spectral mixture kernels are an alternative approach to automatic learning of covariance functions from data, which involves modelling the power spectral density (PSD) of a kernel with a Gaussian mixture and taking the inverse Fourier transform of the PSD to obtain the covariance (Wilson and Adams, 2013). Building on this work, Parra and Tobar (2017) present the *multi-output spectral mixture* (MOSM), which involves using a multivariate extension to Bochner's theorem in order to extend this approach to MOGPs. A later work by Altamirano and Tobar (2022) further extends the MOSM to the case of nonstationary data using harmonizable kernels which automatically identify nonstationary behaviour. Benton et al. (2019) present another means of nonparametric learning of covariances which involves representing the log of the PSD with a GP and applying Bochner's theorem to yield a covariance function. This approach allows for exact GP inference after the covariance has been approximated, however as a result the model is not as scalable as the NP-CGP.

5 EXPERIMENTS

In this section, we provide a set of experiments that illustrate the performance of our model, and compare the model to relevant prior work. The numerical results used to generate the figures in this section are provided in the supplemental material, alongside further details regarding experimental setups and the data used. Our implementation of the NP-CGP is publicly available at `https://github.com/magnusross/npcgp`.

5.1 Toy experiment

Firstly, we present a toy experiment that shows the ability of our model to learn known covariances in multiple dimensions. We use a two dimensional input $\mathbf{x} = [x_1, x_2]^\top$ and generate the ground truth outputs $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ by sampling from two different linear combinations of $u_{EQ} \sim \mathcal{GP}(0, k_{EQ})$ and $u_P \sim \mathcal{GP}(0, k_P)$, which are GP priors with an EQ and weakly periodic kernel respectively. Figure 3 shows the true and predicted covariance, as well as the true target functions. The training data is uniformly sampled from the region shown in the plot, with a small amount of noise ($\sigma^2 = 0.01$) being added. From the results shown in Figure 3, we see that the form of the predicted



(a) (b)

Figure 4: (a) shows the predictive mean and variance of the NP-CGP evaluated on the UK apartment price dataset, whilst (b) is a visualisation of the covariances obtained for each spatial dimension.

covariances from the NP-CGP agree qualitatively with the true covariances, although they do not match exactly. These results illustrate the ability of the model to extract the key features of the covariances from a relatively small region of training data. The fact that the model is confident about the covariances when they do not match exactly can be attributed to the mean field variational approximation, discussed in Section 6.

5.2 Regression on a spatial domain

We also include a second illustrative experiment which displays the ability of our model to capture inhomogenous structure on a spatial domain. Specifically, we model the variation of apartment prices across the United Kingdom, using a dataset consisting of 149,659 observations. The results of this experiment are shown in Figure 4, which includes a visualisation of the predictive mean and variance obtained, alongside the covariance associated with each spatial dimension. The covariance deviates considerably from an EQ-like form in both dimensions, but particularly so for latitude.

5.3 UCI regression

We evaluate our model on four multi-input, single output UCI regression benchmark datasets (Dua and Graff, 2017). We specifically chose datasets which have been shown to benefit from additional model complexity in previous work (Salimbeni and Deisenroth, 2017) due to either their size, complex structure or a combination of the these factors. The results are presented in Figure 5, where we compare our model with 100 (NP-CGP) and 300 (NP-CGP-300) inducing points respectively, to stochastic variational GPs with ARD EQ kernels which also use 100 (SGP) and 300 (SGP-300) inducing points. Additionally, we compare to a SGP with 100 inducing points and the Matérn 3/2 ARD kernel (SGP-M32). The results show that the NP-CGP performs similarly to the SGP for the *energy* and *kin8nm* datasets when comparing to models with 100 inducing points, and slightly worse when the number of inducing points is raised to 300. All models provide similar results for *power*, and for *protein* the NP-CGP with 300 inducing points performs significantly better than the other models. We can see that the NP-CGP performs better, relatively, for datasets with more examples, an observation that is additionally supported by the results in Section 5.5, and which will be discussed in further detail in Section 6. Overall, these results show that although the NP-CGP does not always provide the best performance for all datasets, it can provide significant improvements for specific problems.

In order to test the scalability of our approach with respect to the number of inducing points, we also evaluated the NP-CGP with 1000 inducing points on the largest UCI dataset, *protein*. The resulting RMSE was 3.84 (0.01), whilst the MNLL was 2.79 (0.01). This is roughly comparable to the results obtained using an SGP with 1000 inducing points, an RMSE of 3.82 (0.02) and MNLL of 2.75 (0.004).

5.4 Regression with multiple inputs and outputs

Additionally, to demonstrate the utility of our approach for general regression with multiple inputs and outputs we fit the model on three datasets of that type: *energy* and *naval*, which are medium size datasets with two outputs, and *polymer*, a small dataset with four outputs. In Figure 6, we present results for the full (NP-CGP) and fast (FNP-CGP) variants of our model, alongside a stochastic variational MOGP, which uses the *linear model of coregionalization* (SLMC) (Álvarez et al., 2012) and an exact convolved

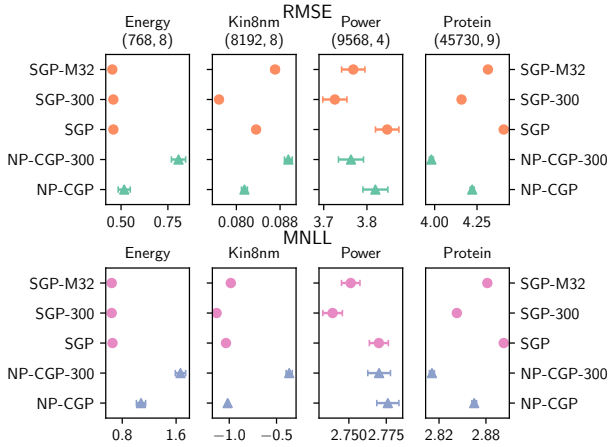


Figure 5: UCI regression results, showing mean and standard error over 20 train/test splits of RMSE and MNLL for our models (▲ and ▲) and of the baselines (● and ●). Lower (i.e. further to the left) is better. Plot titles additionally show (N, P) for each dataset.

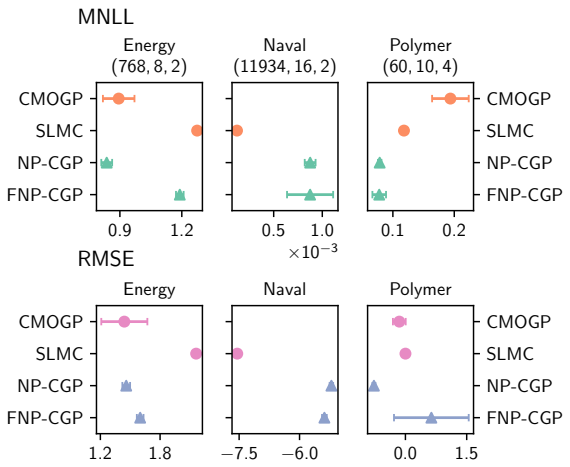


Figure 6: Results over 20 train/test splits for the experiments with multiple inputs and outputs. Plot titles additionally show (N, P, D) for each dataset.

MOGP (CMOGP)[1](Álvarez and Lawrence, 2011), which is comparable to our model, but with parametric convolutions instead. We find that our NP-CGP either matches or exceeds the performance of the other models tested for the *energy* and *polymer* datasets. The SLMC exhibits improved performance for *naval*, however this dataset is known to be easy for GPs, and even linear models, to fit well (Salimbeni and Deisenroth, 2017). The FNP-CGP exhibits similar performance to the full model for *energy* and *polymer*, whilst for the *energy* dataset performance is worse. The variance of the results is higher for the FNP-CGP, likely because the constraints on the covariances make finding an optimal solution during optimisation more difficult. These results illustrate that the NP-CGP model can provide benefits for regression problems with multiple outputs, when compared to standard MOGP models.

5.5 Large-scale regression

To show the scalability of our model to regression problems with hundreds of thousands of observations, we evaluate the NP-CGP on two large-scale single-output regression benchmark datasets, *airline* and *houseelectric*. *airline* is a commonly used benchmark for GPs, where each observation has dimensionality $P = 8$. Specifically, we use the first 700k observations for training and the next 100k for testing, with a mini-batch size of 10k. For *houseelectric*, there are approximately 2.1 millions observations, each with dimensionality $P = 11$. We evaluate on a randomly selected 10% subset of the dataset and train on the remaining observations, with a mini-batch size of 5k.

The results displayed in Table 1 show that the NP-CGP with

[1]No results available for CMOGP on *naval* as dataset is too large for inference in a reasonable amount of time.

	RMSE (↓)		MNLL (↓)	
	airline	house	airline	house
SGP	26.2	0.052	4.68	-1.53
SGP-500	25.9	0.050	4.67	-1.57
SGP-1000	25.0	0.049	4.64	-1.58
DGP2	24.9	**0.044**	4.63	**-1.71**
NP-CGP	**24.6**	0.048	**4.62**	-1.59

Table 1: Results for the large-scale regression experiments, *airline* and *houseelectric* (*house*).

100 inducing points outperforms conventional variational GPs with 100, 500 and 1000 inducing points (SGP, SGP-500 and SGP-1000) across both datasets and metrics. Additionally, we see that on the *airline* experiment, the NP-CGP also achieves superior performance to the two layer deep GP of Salimbeni and Deisenroth (2017) (DGP2). The latter is a particularly encouraging result, as it suggests that in some settings, the performance gap between shallow and hierarchical models may be bridged by utilising our nonparametric approach to learning covariances, although it should be noted that adding additional layers to the DGP can increase performance further, beyond that of our model.

Figure 7 shows the nonparametric covariances learned by the NP-CGP for each feature in the *airline* dataset; in our model, we take the product over all of these in order to obtain the full covariance. We can see here that for some features such as *PlaneAge*, *AirTime* and *Distance*, the form of the covariance is very similar to that of the EQ. However, in the covariances for the other features, we see that the NP-CGP has been able to learn a much richer representation of the data, which likely explains the considerable performance gap between the NP-CGP and SGP models on this problem.

6 DISCUSSION

The results of our experiments show that the NP-CGP can provide significant improvements over standard approaches, most notably for large scale regression, however the model does not outperform all of the competing approaches we consider on all problems. The proposed approach seems to exhibit worse performance on the smaller scale problems, and those which have simple structure (i.e. they are well described by linear models). We believe this is due to the mean field variational approximation that is used for the posterior distribution over the covariances. In the mean field scheme, we approximate the posterior over the convolutional kernel and input processes as independent, which presents a problem, since many combinations of input and kernel can lead to the same output. This case cannot be properly represented under the mean field assumption. The result of this is that the NP-CGP can over-fit for some problems, learn-

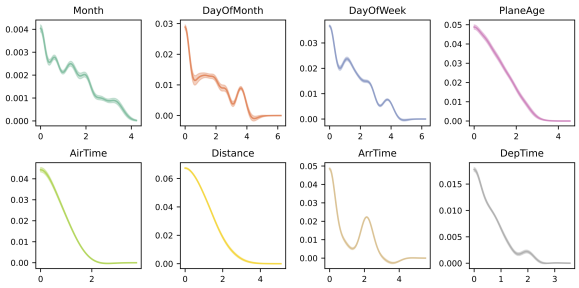


Figure 7: Visualisation of the covariances learned by the NP-CGP for the airline dataset. The solid line is the mean computed using 50 samples from the model, and the shaded confidence interval denotes $\pm 2\sigma$.

ing complex covariances with high confidence, when there is insufficient evidence for such covariances in the data, leading to poor performance on the test set. This does not appear to be a problem for larger datasets, where the model is able to learn covariances much more robustly, leading to improved performance. Bruinsma et al. (2022) recently developed a structured variational scheme for the GPCM, in which the posterior over the input process and convolutional kernel is expressed jointly, allowing for significantly improved quantification of uncertainty for the covariance. The implementation of this improved scheme for the NP-CGP would likely address the over-fitting problem, but is highly non-trivial, therefore we plan to investigate this in future work.

An additional possible extension to this work is to incorporate our approach into a deep GP. In deep GPs, the covariances of the internal layers are often set to be EQ, with little motivation or reasoning for this choice, since it is difficult to interpret the effect that the prior on the internal layers has on the output. Since the NP-CGP uses a similar DSVI scheme to Salimbeni and Deisenroth (2017) it should be possible to stack layers of NP-CGPs to form a deep model, in which the covariances of the internal layers are inferred from the data.

7 CONCLUSION

In this work we have presented a nonparametric process convolutions model that is suitable for regression tasks with multiple outputs and inputs, along with efficient sampling and inference schemes based on the adaptation of fast functional sampling methods to interdomain GPs. We have shown that allowing the form of the covariance to be directly inferred from the data can lead to increased performance compared to standard GP models across a number of different datasets, in particular for large-scale regression tasks. As we have discussed, the mean field inference scheme we employ has some limitations, which we plan to address in future work.

Acknowledgments

Thomas M. McDonald and Magnus Ross thank the Departments of Computer Science at both the University of Manchester and the University of Sheffield for their financial support, as part of this work was carried out whilst they and Mauricio A. Álvarez were at the University of Sheffield. Mauricio A. Álvarez was partially funded by EPSRC Research Projects EP/R034303/1, EP/T00343X/2 and EP/V029045/2, and by the Wellcome Trust project 217068/Z/19/Z. The authors would like to acknowledge the assistance given by Research IT and the use of the Computational Shared Facility at The University of Manchester. Additionally, the authors wish to acknowledge CSC – IT Center for Science, Finland, for computational resources.

References

M. Altamirano and F. Tobar. Nonstationary multi-output Gaussian processes via harmonizable spectral mixtures. In *International Conference on Artificial Intelligence and Statistics*, pages 3204–3218. PMLR, 2022.

M. Alvarez, D. Luengo, and N. D. Lawrence. Latent force models. In *Artificial Intelligence and Statistics*, pages 9–16. PMLR, 2009.

M. Álvarez, D. Luengo, M. Titsias, and N. D. Lawrence. Efficient multioutput Gaussian processes through variational inducing kernels. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 25–32, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. JMLR Workshop and Conference Proceedings. URL `http://proceedings.mlr.press/v9/alvarez10a.html`.

M. A. Álvarez and N. D. Lawrence. Computationally efficient convolved multiple output Gaussian processes. *The Journal of Machine Learning Research*, 12:1459–1500, 2011.

M. A. Álvarez, L. Rosasco, and N. D. Lawrence. Kernels for vector-valued functions: A review. *Found. Trends Mach. Learn.*, 4(3):195–266, Mar. 2012. ISSN 1935-8237. doi: 10.1561/2200000036. URL `https://doi.org/10.1561/2200000036`.

R. P. Barry and J. M. Ver Hoef. Blackbox kriging: spatial prediction without specifying variogram models. *Journal of Agricultural, Biological and Environmental Statistics*, 1(3):297–322, 1996.

G. Benton, W. J. Maddox, J. Salkey, J. Albinati, and A. G. Wilson. Function-space distributions over kernels. In *Advances in Neural Information Processing Systems*, pages 14965–14976, 2019.

K. Blomqvist, S. Kaski, and M. Heinonen. Deep convolutional Gaussian processes. In *Joint European Confer-*

ence on Machine Learning and Knowledge Discovery in Databases, pages 582–597. Springer, 2019.

W. Bruinsma. *The Generalised Gaussian Process Convolution Model*. MPhil. thesis, University of Cambridge, 2016.

W. P. Bruinsma, M. Tegnér, and R. E. Turner. Modelling non-smooth signals with complex spectral structure. In *International Conference on Artificial Intelligence and Statistics*, pages 5166–5195. PMLR, 2022.

A. Damianou and N. D. Lawrence. Deep Gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. *Advances in neural information processing systems*, 31, 2018.

P. Hegde, M. Heinonen, H. Lähdesmäki, and S. Kaski. Deep learning with differential Gaussian process flows. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1812–1821. PMLR, 16–18 Apr 2019. URL `http://proceedings.mlr.press/v89/hegde19a.html`.

D. Higdon. Space and space-time modeling using process convolution. In *Quantitative methods for current environmental issues*, pages 37–56. Springer, 2002.

P. A. Jang, A. Loeb, M. Davidow, and A. G. Wilson. Scalable Levy process priors for spectral kernel learning. *Advances in neural information processing systems*, 30, 2017.

A. G. Journel and C. J. Huijbregts. Mining geostatistics, Jan 1976.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

D. Kong, Y. Chen, and N. Li. Gaussian process regression for tool wear prediction. *Mechanical systems and signal processing*, 104:556–574, 2018.

M. Lázaro-Gredilla and A. Figueiras-Vidal. Inter-domain Gaussian processes for sparse inference using inducing features. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22, pages 1087–1095. Curran Associates, Inc., 2009.

T. McDonald and M. Alvarez. Compositional modeling of nonlinear dynamical systems with ODE-based random features. *Advances in Neural Information Processing Systems*, 34, 2021.

G. Parra and F. Tobar. Spectral mixture kernels for multi-output Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 6681–6690, 2017.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 3, 2007.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

R. R. Richardson, C. R. Birkl, M. A. Osborne, and D. A. Howey. Gaussian process regression for in situ capacity estimation of lithium-ion batteries. *IEEE Transactions on Industrial Informatics*, 15(1):127–138, 2018.

M. Ross, M. T. Smith, and M. Alvarez. Learning nonparametric Volterra kernels with Gaussian processes. *Advances in Neural Information Processing Systems*, 34, 2021.

H. Salimbeni and M. P. Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, 2017.

A. Shafieloo, A. G. Kim, and E. V. Linder. Gaussian process cosmography. *Physical Review D*, 85(12):123530, 2012.

Z. Shen, M. Heinonen, and S. Kaski. Harmonizable mixture kernels with variational Fourier features. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3273–3282. PMLR, 2019.

F. Tobar, T. D. Bui, and R. E. Turner. Learning stationary time series using Gaussian processes with nonparametric kernels. In *Advances in Neural Information Processing Systems*, pages 3501–3509, 2015.

A. Wilson and R. Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International conference on machine learning*, pages 1067–1075, 2013.

A. G. Wilson, E. Gilboa, A. Nehorai, and J. P. Cunningham. Fast kernel learning for multidimensional pattern extrapolation. *Advances in Neural Information Processing Systems*, 27, 2014.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378, 2016.

J. Wilson, V. Borovitskiy, A. Terenin, P. Mostowsky, and M. Deisenroth. Efficiently sampling functions from Gaussian process posteriors. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10292–10302. PMLR, 13–18 Jul 2020. URL `http://proceedings.mlr.press/v119/wilson20a.html`.

A ADDITIONAL DERIVATIONS

A.1 Pathwise sampling derivations

To sample from the NP-CGP, we first must sample from the inducing point variational distributions, $\mathbf{V}^{\mathbf{G}} \sim q(\mathbf{V}^{\mathbf{G}})$ and $\mathbf{V}^{\tilde{\mathbf{u}}} \sim q(\mathbf{V}^{\tilde{\mathbf{u}}})$, then use the pathwise sampling method introduced by Wilson et al. (2020) to sample input functions and convolutional kernel functions. We then map these analytically through $\mathbf{f}^{(s)}(\mathbf{x}) = \int_{\mathbb{R}^P} \mathbf{G}^{(s)}(\mathbf{x}-\mathbf{z})\mathbf{u}^{(s)}(\mathbf{z})d\mathbf{z}$ in a similar fashion to Ross et al. (2021), however in our case, as we have a multi-dimensional input, we split the multi-dimensional integrals involved into products of one dimensional integrals, which yields a closed form expression for a sample function from the model,

$$\begin{aligned}\left(\mathbf{f}^{(s)}|\mathbf{V}^{\mathbf{G}}, \mathbf{V}^{\tilde{\mathbf{u}}}\right)(\mathbf{x}) = \sum_{k=1}^{N_b} w_k^u \Bigg(\frac{e^{i\beta_k^u}}{2} \prod_{p=1}^{P} \Bigg(\sum_{i=1}^{N_b} w_i^{G_p} I_{1\text{A}}\Big(x_p; \alpha_p, \theta_i^{G_p}, \beta_i^{G_p}, \theta_{k,p}^u\Big) \\ + \sum_{j=1}^{M} q_j^{G_p} I_{1\text{B}}\Big(x_p; \alpha_p, z_j^{G_p}, \rho^{G_p}, \theta_{k,p}^u\Big)\Bigg) \\ + \frac{e^{-i\beta_k^u}}{2} \prod_{p=1}^{P} \Bigg(\sum_{i=1}^{N_b} w_i^{G_p} I_{1\text{A}}\Big(x_p; \alpha_p, \theta_i^{G_p}, \beta_i^{G_p}, -\theta_{k,p}^u\Big) \\ + \sum_{j=1}^{M} q_j^{G_p} I_{1\text{B}}\Big(x_p; \alpha_p, z_j^{G_p}, \rho^{G_p}, -\theta_{k,p}^u\Big)\Bigg)\Bigg) \\ + \sum_{l=1}^{M} q_l^u \prod_{p=1}^{P} \Bigg(\sum_{i=1}^{N_b} w_i^{G_p} I_{2\text{A}}\Big(x_p; \alpha_p, \theta_i^{G_p}, \beta_i^{G_p}, \rho_p^u, z_{l,p}^u\Big) \\ + \sum_{j=1}^{M} q_j^{G_p} I_{2\text{B}}\Big(x_p; \alpha_p, \rho^{G_p}, z_j^{G_p}, \rho_p^u, z_{l,p}^u\Big)\Bigg),\end{aligned} \tag{8}$$

where,

$$\begin{aligned}I_{1\text{A}}\Big(x; \alpha, \theta_1, \beta, \theta_2\Big) &= \int_{-\infty}^{\infty} e^{-\alpha(x-\tau)^2} \cos\left(\theta_1\tau + \beta\right) e^{i\theta_2\tau} d\tau \\ &= \frac{\sqrt{\pi}}{2\sqrt{\alpha}} \left(\exp\left(\frac{\theta_1\theta_2}{\alpha}\right) + \exp\left(2i(\theta_1 x + \beta)\right)\right) \times \\ &\qquad \exp\left(-i(\beta + (\theta_1 - \theta_2)x) - \frac{-(\theta_1+\theta_2)^2}{4\alpha}\right),\end{aligned} \tag{9}$$

$$\begin{aligned}I_{1\text{B}}\Big(x; \alpha, z, \rho, -\theta\Big) &= \int_{-\infty}^{\infty} e^{-\alpha(x-\tau)^2} k(\tau, z) e^{i\theta\tau} d\tau \\ &= \frac{\sqrt{\pi}}{\sqrt{\alpha+\rho}} \exp\left(-\frac{\theta^2 + 4\alpha\rho(x-z)^2 - 4i\theta(\alpha x + \rho z)}{4(\alpha+\rho)}\right),\end{aligned} \tag{10}$$

$$\begin{aligned}I_{2\text{A}}\Big(x; \alpha, \theta, \beta, \rho, z\Big) &= \int_{-\infty}^{\infty} e^{-\alpha(x-\tau)^2} \cos\left(\theta\tau + \beta\right) e^{-\rho(\tau-z)^2} d\tau \\ &= \frac{\sqrt{\pi}}{\sqrt{\alpha+\rho}} \cos\left(\beta + \theta\left(x - \frac{\rho z}{\sqrt{\alpha+\rho}}\right)\right) \exp\left(-\frac{(\theta)^2 + 4\alpha}{\rho(z)^2}\right),\end{aligned} \tag{11}$$

$$\begin{aligned}I_{2\text{B}}\Big(x; \alpha, \rho_1, z_1, \rho_2, z_2\Big) &= \int_{-\infty}^{\infty} e^{-\alpha(x-\tau)^2} k(\tau, z_2) e^{-\rho_2(\tau-z_2)^2} d\tau \\ &= \frac{\sqrt{\pi}}{\sqrt{\alpha+\rho_1+\rho_2}} \exp\left(\frac{\alpha\left(\rho_1(z_1)^2 + \rho_2(x-z_2)^2\right) + \rho_1\rho_2\left(-x+z_1+z_2\right)^2}{\alpha+\rho_1+\rho_2}\right).\end{aligned} \tag{12}$$

A.2 Interdomain sampling derivations

In the main paper, we use the closed form expression for the Gaussian transformed basis functions in order to sample from the interdomain input process in our model. For an input observation $\mathbf{x} \in \mathbb{R}^P$, and a given latent process $\tilde{u}_q(\mathbf{x})$ (we omit the subscript for ease of exposition below), this expression takes the form,

$$\begin{aligned}\tilde{u}(\mathbf{x}) &= \sum_{j=1}^{N_b} w_j \int_{\mathbb{R}^P} e^{-\sum_{i=1}^P \rho_i (z_i - x_i)^2} \cos\left(\boldsymbol{\theta}_j^\top \mathbf{z} + \beta_j\right) d\mathbf{z} \\ &= \sum_{j=1}^{N_b} \frac{e^{-\beta_j}}{2} \prod_{i=1}^P \int_{-\infty}^{\infty} e^{-\rho_i (z-x_i)^2 - i\theta_{i,j} z} dz + \frac{e^{\beta_j}}{2} \prod_{i=1}^P \int_{-\infty}^{\infty} e^{-\rho_i (z-x_i)^2 + \theta_{i,j} z} dz,\end{aligned} \tag{13}$$

where $\mathbf{w}$, $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ define our random Fourier feature basis. $\mathbf{w}$ consists of entries $w_j \sim \mathcal{N}(0,1)$, $\boldsymbol{\beta}$ consists of entries $\beta_j \sim U(0, 2\pi)$ and $\boldsymbol{\theta}$ consists of entries $\boldsymbol{\theta}_j \sim \text{FT}(k)$, where FT is the Fourier transform and k represents the covariance of the untransformed process. This transform allows us to compute the elements of $\tilde{\boldsymbol{\Phi}}$ in Eq. 5 of the main paper, which in turn allows us to sample from our input process. Eq. 5 itself follows from a reformulation of Eq. 13 in Wilson et al. (2020),

$$(u_q^{(s)} \mid \mathbf{v}^{u_q})(\cdot) = \sum_{i=1}^{N_b} w_i \phi_i(\cdot) + k\left(\cdot, \boldsymbol{z}^{u_q}\right) \mathbf{K}_{u_q,u_q}^{-1} (\mathbf{v}^{u_q} - \boldsymbol{\Phi}\mathbf{w}). \tag{14}$$

In this work, as we are using an interdomain input process, we must ensure that the update term in this expression is computed in the transformed domain. This is achieved by replacing the cross-covariance $k\left(\cdot, \boldsymbol{z}^{u_q}\right)$ with $k_{u_q,\tilde{u}_q}\left(\cdot, \boldsymbol{z}^{\tilde{u}_q}\right)$ and the covariance matrix $\mathbf{K}_{u_q,u_q}^{-1}$ with its interdomain equivalent $\mathbf{K}_{\tilde{u}_q,\tilde{u}_q}^{-1}$. We also replace the inducing points $\mathbf{v}^{u_q}$ with the interdomain inducing points $\mathbf{v}^{\tilde{u}_q}$, and the basis functions $\boldsymbol{\Phi}$ with their transformed counterpart $\tilde{\boldsymbol{\Phi}}$. Applying these changes yields Eq. 5, as stated in the main paper. The covariance used to compute the elements of $\mathbf{K}_{\tilde{u}_q,\tilde{u}_q}$ can be expressed as,

$$k_{\tilde{u}_q,\tilde{u}_q}\left(\mathbf{z}^{\tilde{u}_q}, \mathbf{z}^{\tilde{u}_q\prime}\right) = \int\int_{\mathbb{R}^P} k_{u_q}(\mathbf{x}, \mathbf{x}') g_q(\mathbf{x}, \mathbf{z}^{\tilde{u}_q}) g_q(\mathbf{x}', \mathbf{z}^{\tilde{u}_q\prime}) d\mathbf{x}' d\mathbf{x}, \tag{15}$$

whilst the aforementioned cross-covariance can be expressed as,

$$k_{u_q,\tilde{u}_q}\left(\mathbf{x}, \mathbf{z}^{\tilde{u}_q\prime}\right) = \int_{\mathbb{R}^P} k_{u_q}(\mathbf{x}, \mathbf{x}') g_q(\mathbf{x}', \mathbf{z}^{\tilde{u}_q\prime}) d\mathbf{x}'. \tag{16}$$

A.3 Variational lower bound derivation

In this section, we present the full derivation of the variational lower bound for the NP-CGP. Denoting our input data as $\mathbf{X} \in \mathbb{R}^{N \times P}$, and the corresponding outputs as $\mathbf{Y} \in \mathbb{R}^{N \times D}$, we can express the joint distribution of the NP-CGP as,

$$p(\mathbf{Y}, \mathbf{G}, \mathbf{V}^{\mathbf{G}}, \mathbf{u}, \mathbf{V}^{\tilde{\mathbf{u}}}) = \prod_{i=1}^N p(\mathbf{y}_i|\mathbf{f}_i) p(\mathbf{G}|\mathbf{V}^{\mathbf{G}}) p(\mathbf{V}^{\mathbf{G}}) p(\mathbf{u}|\mathbf{V}^{\tilde{\mathbf{u}}}) p(\mathbf{V}^{\tilde{\mathbf{u}}}), \tag{17}$$

where the likelihood is given by $p(\mathbf{y}_i|\mathbf{f}_i) = \mathcal{N}(\mathbf{y}_i; \mathbf{f}_i, \sigma_Y^2)$. As all of the convolutional kernel and input GPs are independent, we have $p(\mathbf{G}|\mathbf{V}^{\mathbf{G}}) = \prod_{d=1}^{D_\ell} p(G_d|\mathbf{v}^{G_d})$, where $p(G_d|\mathbf{v}^{G_d})$ is the GP posterior distribution given the inducing points, and likewise $p(\mathbf{u}|\mathbf{V}^{\tilde{\mathbf{u}}}) = \prod_{q=1}^{Q_\ell} p(u_q|\mathbf{v}^{\tilde{u}_q})$, where again $p(u_q|\mathbf{v}^{\tilde{u}_q})$ are GP posteriors. $p(\mathbf{V}^{\mathbf{G}})$ and $p(\mathbf{V}^{\tilde{\mathbf{u}}})$ represent the priors over the inducing points. Our variational posterior takes the form,

$$q(\mathbf{G}, \mathbf{V}^{\mathbf{G}}, \mathbf{u}, \mathbf{V}^{\tilde{\mathbf{u}}}) = p(\mathbf{G}|\mathbf{V}^{\mathbf{G}}) q(\mathbf{V}^{\mathbf{G}}) p(\mathbf{u}|\mathbf{V}^{\tilde{\mathbf{u}}}) q(\mathbf{V}^{\tilde{\mathbf{u}}}), \tag{18}$$

where $q(\mathbf{V}^{\mathbf{G}}) = \prod_{d=1}^D \mathcal{N}(\mathbf{v}^{G_d}; \boldsymbol{\mu}^{G_d}, \boldsymbol{\Sigma}^{G_d})$ and $q(\mathbf{V}^{\tilde{\mathbf{u}}}) = \prod_{q=1}^Q \mathcal{N}(\mathbf{v}^{\tilde{u}_q}; \boldsymbol{\mu}^{\tilde{u}_q}, \boldsymbol{\Sigma}^{\tilde{u}_q})$ are variational distributions, whose means and covariance matrices are variational parameters. We can write down the variational lower bound as,

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{G}, \mathbf{V}^{\mathbf{G}}, \mathbf{u}, \mathbf{V}^{\tilde{\mathbf{u}}})} \left[\frac{p(\mathbf{Y}, \mathbf{G}, \mathbf{V}^{\mathbf{G}}, \mathbf{u}, \mathbf{V}^{\tilde{\mathbf{u}}})}{q(\mathbf{G}, \mathbf{V}^{\mathbf{G}}, \mathbf{u}, \mathbf{V}^{\tilde{\mathbf{u}}})}\right] \tag{19}$$

$$= \int q(\mathbf{G}, \mathbf{V}^{\mathbf{G}}, \mathbf{u}, \mathbf{V}^{\tilde{\mathbf{u}}}) \log \left[\frac{p(\mathbf{Y}, \mathbf{G}, \mathbf{V}^{\mathbf{G}}, \mathbf{u}, \mathbf{V}^{\tilde{\mathbf{u}}})}{q(\mathbf{G}, \mathbf{V}^{\mathbf{G}}, \mathbf{u}, \mathbf{V}^{\tilde{\mathbf{u}}})}\right] d\mathbf{S}, \tag{20}$$

where $d\mathbf{S}$ represents the integral over all of the inducing points, convolutional kernel and input processes. Using Eq. 17 and Eq. 18, we can derive a form of the evidence lower bound (ELBO) which we can use to perform approximate inference as follows:

$$\begin{aligned}\mathcal{L} = &\int \prod_{\ell=1} p(\mathbf{G}|\mathbf{V}^{\mathbf{G}})q(\mathbf{V}^{\mathbf{G}})p(\mathbf{u}|\mathbf{V}^{\tilde{\mathbf{u}}})q(\mathbf{V}^{\tilde{\mathbf{u}}}) \\ &\times \log\left[\frac{\prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{f}_i)p(\mathbf{G}|\mathbf{V}^{\mathbf{G}})p(\mathbf{V}^{\mathbf{G}})p(\mathbf{u}|\mathbf{V}^{\tilde{\mathbf{u}}})p(\mathbf{V}^{\tilde{\mathbf{u}}})}{\prod_{\ell=1} p(\mathbf{G}|\mathbf{V}^{\mathbf{G}})q(\mathbf{V}^{\mathbf{G}})p(\mathbf{u}|\mathbf{V}^{\tilde{\mathbf{u}}})q(\mathbf{V}^{\tilde{\mathbf{u}}})}\right] d\mathbf{S} \qquad (21)\\ = &\int p(\mathbf{G}|\mathbf{V}^{\mathbf{G}})q(\mathbf{V}^{\mathbf{G}})p(\mathbf{u}|\mathbf{V}^{\tilde{\mathbf{u}}})q(\mathbf{V}^{\tilde{\mathbf{u}}}) \\ &\times \log\left[\frac{\prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{f}_i)p(\mathbf{V}^{\mathbf{G}})p(\mathbf{V}^{\tilde{\mathbf{u}}})}{\prod_{\ell=1} q(\mathbf{V}^{\mathbf{G}})q(\mathbf{V}^{\tilde{\mathbf{u}}})}\right] d\mathbf{S} \qquad (22)\\ = &\int p(\mathbf{G}|\mathbf{V}^{\mathbf{G}})q(\mathbf{V}^{\mathbf{G}})p(\mathbf{u}|\mathbf{V}^{\tilde{\mathbf{u}}})q(\mathbf{V}^{\tilde{\mathbf{u}}}) \log\left[\prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{f}_i)\right] d\mathbf{S} \qquad (23)\\ &- \mathrm{KL}[q(\mathbf{V}^{\tilde{\mathbf{u}}})\|p(\mathbf{V}^{\tilde{\mathbf{u}}})] + \mathrm{KL}[q(\mathbf{V}^{\mathbf{G}})\|p(\mathbf{V}^{\mathbf{G}})] \\ = &\sum_{i=1}^{N} \mathbb{E}_{q(\{\mathbf{G},\mathbf{V}^{\mathbf{G}},\mathbf{u},\mathbf{V}^{\tilde{\mathbf{u}}}\})}\left[\log p(\mathbf{y}_i|\mathbf{f}_i)\right] \qquad (24)\\ &- \mathrm{KL}[q(\mathbf{V}^{\tilde{\mathbf{u}}})\|p(\mathbf{V}^{\tilde{\mathbf{u}}})] + \mathrm{KL}[q(\mathbf{V}^{\mathbf{G}})\|p(\mathbf{V}^{\mathbf{G}})]\end{aligned}$$

As the KL divergences present are between sets of independent multivariate Gaussian distributions, we have $\mathrm{KL}[q(\mathbf{V}^{\tilde{\mathbf{u}}})\|p(\mathbf{V}^{\tilde{\mathbf{u}}})] = \sum_{q=1}^{Q} \mathrm{KL}[\mathcal{N}(\mathbf{v}^{\tilde{u}_q};\boldsymbol{\mu}^{\tilde{u}_q},\boldsymbol{\Sigma}^{\tilde{u}_q})\|\mathcal{N}(\mathbf{v}^{\tilde{u}_q};0,\boldsymbol{K}^{\tilde{u}_q})]$ and $\mathrm{KL}[q(\mathbf{V}^{\mathbf{G}})\|p(\mathbf{V}^{\mathbf{G}})] = \sum_{d=1}^{D} \mathrm{KL}[\mathcal{N}(\mathbf{v}^{G_d};\boldsymbol{\mu}^{G_d},\boldsymbol{\Sigma}^{G_d})\|\mathcal{N}(\mathbf{v}^{G_d};0,\boldsymbol{K}^{G_d})]$, which have well known tractable form. Conversely, we approximate the intractable expectation from the first line of Eq. (24) stochastically using S Monte Carlo samples,

$$\mathbb{E}_{q(\{\mathbf{G},\mathbf{V}^{\mathbf{G}},\mathbf{u},\mathbf{V}^{\tilde{\mathbf{u}}}\})}\left[\log p(\mathbf{y}_i|\mathbf{f}_i)\right] \approx \frac{1}{S}\sum_{s=1}^{S} \log p(\mathbf{y}_i|\mathbf{f}_i^{(s)}). \qquad (25)$$

where $\mathbf{f}_i^{(s)}$ denotes a sample from the model.

B MODEL COMPLEXITIES AND RUNTIMES

Table 2 shows complexities for the computation of the bound/sampling for the regular version of our model and the fast approximation, for the cases of single and multiple outputs. We present these alongside the complexities associated with comparable prior work. Note that the complexities do not depend on the size of the data because we employ mini-batching. Recall that M_u is the number of input process inducing points, M_G the number of convolutional kernel process inducing points, M is the number of regular GP inducing points, P denotes the input dimension, D denotes the output dimension, and Q denotes the number of latent functions.

	Model Type		
	Regular	Fast	Prior work
Single output	$\mathcal{O}(M_u^3 + PM_G^3)$	$\mathcal{O}(M_u^3 + PM_G^3)$	$\mathcal{O}(M^3)$ (SGP)
Multi output	$\mathcal{O}(QM_u^3 + PDM_G^3)$	$\mathcal{O}(QM_u^3 + PM_G^3)$	$\mathcal{O}(DM^3)$(SLMC)

Table 2: Computational complexities for the standard and fast versions of the model, and the equivalent prior work.

In practice this corresponds to the run-time of our model being around $15\times$ longer than a standard SGP model. Although this seems drastically slower, this is not a fundamental limitation of the model, and is due to the somewhat complex implementation of our models. We believe the code could likely be optimised in such a way that the model could be made

			RMSE				
	N	P	NP-CGP	NP-CGP-300	SGP	SGP-300	SGP-M32
energy	768	8	0.52 (0.03)	0.81 (0.04)	0.46 (0.01)	0.46 (0.01)	0.45 (0.01)
kin8nm	8192	8	0.08 (0.00)	0.09 (0.00)	0.08 (0.00)	0.08 (0.00)	0.09 (0.00)
power	9568	4	3.82 (0.03)	3.76 (0.03)	3.85 (0.03)	3.73 (0.03)	3.77 (0.03)
protein	45730	9	4.22 (0.01)	3.98 (0.01)	4.41 (0.01)	4.16 (0.01)	4.31 (0.01)

			MNLL				
	N	P	NP-CGP	NP-CGP-300	SGP	SGP-300	SGP-M32
energy	768	8	1.08 (0.07)	1.67 (0.08)	0.65 (0.03)	0.64 (0.03)	0.64 (0.03)
kin8nm	8192	8	-1.02 (0.01)	-0.37 (0.03)	-1.03 (0.00)	-1.13 (0.00)	-0.98 (0.00)
power	9568	4	2.78 (0.01)	2.77 (0.01)	2.77 (0.01)	2.74 (0.01)	2.75 (0.01)
protein	45730	9	2.86 (0.00)	2.81 (0.00)	2.90 (0.00)	2.84 (0.00)	2.88 (0.00)

Table 3: Results over 20 train/test splits for the UCI regression experiments. The mean values are reported, with the standard error in brackets. N represents the number of observations in each dataset, and P the number of input dimensions.

to run much faster. In particular, the computation of gradients with respect to the integrals required for sampling took significantly longer than expected, as the PyTorch framework is not well optimised for computing gradients with respect to the complex point-wise operations represented by the integrals in the model.

C EXPERIMENTAL DETAILS

The experiments in this work were performed on HPC clusters, using nodes containing a range of different GPUs, including 40GB and 80GB NVIDIA A100-SXM4 GPUs, as well as 16GB and 32GB NVIDIA Tesla V100-SXM2 GPUs. Throughout we utilise the interdomain transform for the input process, since without this element we were unable to achieve convergence during training for most problems. Throughout we use $S = 2$ Monte Carlo samples, since we found this to give the best trade off between computation speed and variance of the estimates of the variational objective.

C.1 Toy experiment

For our toy experiment, we used an input $\mathbf{x} \in \mathbb{R}^{N \times P}$, with $N = 3000$ and $P = 2$, where the entries of $\mathbf{x}$ were sampled from a standard normal distribution. As mentioned in the main paper, the ground truth function values in this experiment, $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, were generated by sampling from two different linear combinations of two GP priors $u_{\text{EQ}} \sim \mathcal{GP}(0, k_{\text{EQ}})$ and $u_{\text{P}} \sim \mathcal{GP}(0, k_{\text{P}})$, which have EQ and weakly periodic kernels respectively. k_{EQ} uses a lengthscale of 1.5 for each input dimension, and k_{P} is constructed from the product of an EQ kernel and periodic kernel both using lengthscales of 1.5 per dimension, with periods of 1.8 and 2.1 for each input dimension in the periodic component. Specifically, the linear combinations we use are $f_1 = 0.9u_{\text{EQ}} + 0.1u_P$ and $f_2 = 0.5u_{\text{EQ}} + 0.5u_P$. Prior to sampling our ground truth output values, we also applied independent Gaussian noise to each output with $\sigma = 0.01$.

C.2 UCI regression

The numerical values used to generate the boxplots in Figure 4 in the main paper, are shown in Table 3. As discussed in the main paper, all data is freely available from the UCI Machine Learning Repository (Dua and Graff, 2017). For these experiments, we performed 20 different random splits of the standardised data (which were kept in common across all of the models evaluated), using 90% of each dataset for training and the remaining 10% to evaluate the test set metrics which we report in the paper and this appendix. For our NP-CGP models, we used 16 basis functions for estimating the lower bound, initialising the likelihood variance to 0.01. We used 15 inducing points for the convolutional kernel processes, and either 100 (NP-CGP) or 300 (NP-CGP-300) inducing points for the input processes, and these input inducing points were initialised using k-means clustering. Training was performed for 40,000 iterations with a batch size of 1000, using the *Adam* optimiser (Kingma and Ba, 2015) with a learning rate of 0.001. For the variational GP models (SGP, SGP-300 and

				RMSE			
	N	P	D	FNP-CGP	NP-CGP	S-LMC	CMOGP
energy	768	8	2	1.19 (0.02)	0.84 (0.03)	1.27 (0.00)	0.89 (0.08)
naval	11934	16	2	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	-
polymer	60	10	4	0.08 (0.01)	0.08 (0.00)	0.12 (0.00)	0.19 (0.03)

				MNLL			
	N	P	D	FNP-CGP	NP-CGP	S-LMC	CMOGP
energy	768	8	2	1.60 (0.03)	1.46 (0.04)	2.16 (0.01)	1.44 (0.23)
naval	11934	16	2	-5.38 (0.06)	-5.21 (0.04)	-7.55 (0.01)	-
polymer	60	10	4	0.63 (0.91)	-0.77 (0.04)	0.00 (0.00)	-0.15 (0.16)

Table 4: Results over 20 train/test splits for the experiments with multiple inputs and outputs. The mean values are reported, with the standard error in brackets. N represents the number of observations in each dataset, P the number of input dimensions and D the number of outputs.

SGP-M32), we mirrored these settings as closely as possible, with the only difference being that we used a learning rate of 0.01 for the optimiser.

C.3 Regression with multiple inputs and outputs

The numerical values used to generate the boxplots in Figure 5 in the main paper, are shown in Table 4. We include results for regression with multiple inputs and outputs on three different datasets. Firstly, the *energy* dataset is the same dataset used in the UCI experiments, however rather than just using one of the two outputs, we infer both. Similarly, *naval* is another UCI dataset commonly used as a single output benchmark, but we infer both of its outputs in this work. Finally, the *polymer* dataset, freely available at `ftp://ftp.cis.upenn.edu/pub/ungar/chemdata`, was selected in order to test the predictive capability of the model in a small-data setting. The settings used for our NP-CGP models in this experiment broadly mirror those described in Section C.2. For the stochastic multi-output GP (S-LMC), implemented using GPyTorch (Gardner et al., 2018), we used a number of latent GPs equal to the number of outputs for the given dataset, and all experimental settings for this model were the same as those used for the NP-CGP. Similarly, for the convolved MOGP (CMOGP), we also used a number of latent functions equal to the number of outputs.

C.4 Large-scale regression

The settings used for the NP-CGP on the large-scale regression experiment once again broadly mirror those described in Section C.2, with three key differences due to the scale of the datasets: firstly the batch size was increased to 10,000 for *airline* and 5,000 for *houseelectric*, secondly, the number of training iterations was increased to 100,000, and finally, we did not perform repeats. For the DGP2 model we employ EQ ARD kernels and the same initialisations used by Salimbeni and Deisenroth (2017).