

# The Gaussian Process Autoregressive Regression Model (GPARG)

James Requeima\*  
 University of Cambridge,  
 and Invenia Labs, Cambridge, UK  
 jrr41@cam.ac.uk

Wessel Bruinsma\*†  
 University of Cambridge  
 wpb23@cam.ac.uk

Will Tebbutt\*†  
 University of Cambridge  
 wct23@cam.ac.uk

Richard E. Turner  
 University of Cambridge  
 ret26@cam.ac.uk

February 22, 2018

## Abstract

Multi-output regression models must exploit dependencies between outputs to maximise predictive performance. The application of Gaussian processes (GPs) to this setting typically yields models that are computationally demanding and have limited representational power. We present the Gaussian Process Autoregressive Regression (GPARG) model, a scalable multi-output GP model that is able to capture nonlinear, possibly input-varying, dependencies between outputs in a simple and tractable way: the product rule is used to decompose the joint distribution over the outputs into a set of conditionals, each of which is modelled by a standard GP. GPARG’s efficacy is demonstrated on a variety of synthetic and real-world problems, outperforming existing GP models and achieving state-of-the-art performance on the tasks with existing benchmarks.

## 1 Introduction

The Gaussian process (GP) probabilistic modelling framework provides a powerful and popular approach to nonlinear single-output regression (Rasmussen & Williams, 2006). The popularity of GP methods stems from their modularity, tractability, and interpretability: it is simple to construct rich, nonlinear models by compositional covariance function design, which can then be evaluated in a principled way (e.g. via the marginal likelihood), before being interpreted in terms of their component parts. This leads to an attractive plug-and-play approach to modelling and understanding data, which is so robust that it can even be automated

(Duvenaud et al., 2013).

Most regression problems, however, do not comprise a single output at each input location. Typically, multiple outputs are recorded at a single input location, and it is key in such modelling tasks to capture the dependencies between these outputs. For example, the noise in the output space might be correlated, or, whilst one output might depend on the inputs in a complex (deterministic) way, it may depend quite simply on other output variables. In both cases multi-output GP models are required. There are a plethora of existing multi-output GP models that can capture linear correlations between output variables if these correlations are fixed across the input space (Goovaerts, 1997; Wackernagel, 2003; Teh & Seeger, 2005; Bonilla et al., 2008; Nguyen & Bonilla, 2014). However, one of the main reasons for the popularity of the GP approach is that a suite of different types of nonlinear input dependencies can be modelled, and it is disappointing that this flexibility is not extended to interactions between the outputs. There are some approaches that do allow limited modelling of nonlinear output dependencies (Wilson et al., 2012; Bruinsma, 2016), but this flexibility comes from sacrificing tractability with complex and computationally demanding approximate inference and learning schemes now required. This complexity significantly slows down the modelling flow.

What is needed is a flexible and analytically tractable modelling approach to multi-output regression that supports plug-and-play modelling and model interpretation. The Gaussian Process Autoregressive Regression (GPARG) model achieves these aims by taking an approach to multi-output regression that is analogous to that employed by the Neural Autoregressive Density Estimator (Larochelle & Murray, 2011) for density

\*Equal contributions.

†Research primarily conducted whilst at Invenia Labs, Cambridge, UK.

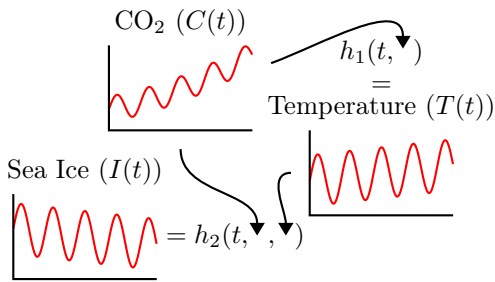


Figure 1: Cartoon motivating a factorisation for the joint distribution  $p(I(t), T(t), C(t))$

modelling. The product rule is used to decompose the distribution of the outputs given the inputs into a set of one-dimensional conditional distributions. Critically, these distributions can be interpreted as a decoupled set of single-output regression problems, and learning and inference in GPAR therefore amount to a set of standard GP regression tasks; that is to say, training is closed-form and fast. GPAR converts the modelling of output dependencies that are possibly nonlinear and input-dependent into a set of standard GP covariance function design problems.

In this paper, we motivate GPAR from a probabilistic perspective. We show that GPAR can express certain prior beliefs about the possibly complex dependencies between inputs and outputs. We then illustrate that GPAR can capture nonlinear relationships between underlying latent processes as well as structured noise. Finally, we apply GPAR to multi-output regression and examine its performance as a Bayesian optimisation objective model, achieving state-of-the-art results on the tasks with existing benchmarks.

## 2 GPAR

Consider the problem of modelling the world’s average CO<sub>2</sub> level  $C(t)$ , temperature  $T(t)$ , and Arctic sea ice extent  $I(t)$  as a function of time  $t$ . By the greenhouse effect, one can imagine that  $T$  is some complicated function  $h_1$  of  $t$  and  $C$ :  $T(t) = h_1(t, C(t))$ . Similarly, one might hypothesise that  $I$  can be modelled as some function  $h_2$  of  $t$ ,  $T$ , and  $C$ :  $I(t) = h_2(t, T(t), C(t))$ . These functional relationships are depicted in Figure 1 and motivate a natural factorisation of the model where the conditionals model the postulated  $h_1$  and  $h_2$ :

$$p(I(t), T(t), C(t)) = \underbrace{p(I(t) | T(t), C(t))}_{\text{model for } h_2} \underbrace{p(T(t) | C(t))}_{\text{model for } h_1} \underbrace{p(C(t))}_{\text{model for } C}.$$

Expressed as such, this multi-output regression problem has been reduced to three single-output regression

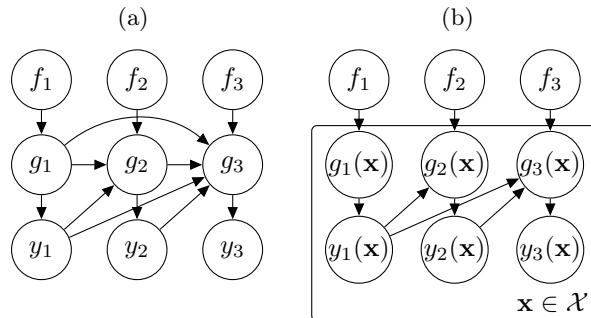


Figure 2: (a) The graphical model corresponding to Full GPAR (FPGAR) and (b) the graphical model corresponding to GPAR

problems: modelling  $h_2$ ,  $h_1$ , and  $C$ . We propose to solve each of these using GPs, enabling us to design kernels that specify nonlinear dependencies between outputs. Note that a variable like sea ice will depend not just on the current temperature, but on the entire history of temperatures; this leads to generalisations where  $I(t)$  depends on  $T(t')$  not just for  $t' = t$ , but for all  $t' \leq t$ . The modelling idea illustrated in this section, which lies at the heart of GPAR, will be formalised in the remainder of this section. Henceforth, we collect the multiple outputs of a multi-output regression problem into a vector-valued function  $\mathbf{g}$ , and the postulated functions  $g$ ,  $f$ , and  $C$  will be generalised to functions  $f_i$ .

Consider a vector-valued stochastic process  $\mathbf{g}$  over an input space  $\mathcal{X} = \mathbb{R}^D$ , where elements  $\mathbf{g}(\mathbf{x})$  take values in  $\mathcal{Y}^M$ ,  $\mathcal{Y} = \mathbb{R}$ .<sup>1</sup> Let  $g_i$  denote the  $i^{\text{th}}$  element of  $\mathbf{g}$ : elements  $g_i(\mathbf{x})$  thus take values in  $\mathcal{Y}$ . We emphasise that  $\mathbf{g}(\mathbf{x})$  refers to the particular  $\mathcal{Y}^M$ -valued random variable at  $\mathbf{x}$ , and that  $\mathbf{g}$  refers to the entire process: the collection of all  $\mathbf{g}(\mathbf{x})$ . Let  $y_i$  be  $g_i$  plus independent noise. Then, according to the product rule,

$$p(\mathbf{y}, \mathbf{g}) = \prod_{i=1}^M p(y_i | g_i) p(g_i | \mathbf{g}_{1:i-1}, \mathbf{y}_{1:i-1}), \quad (1)$$

where  $\mathbf{g}_{1:i-1}$  denotes the vector-valued process consisting of elements  $g_1, \dots, g_{i-1}$ . Recall that a Gaussian process (GP)  $f$  over some index set  $\mathcal{T}$  defines a process where, for any  $t_1, \dots, t_N \in \mathcal{T}$ ,  $f(t_1), \dots, f(t_N)$  are jointly Gaussian distributed in a consistent manner. Let  $\mathcal{Y}^{\mathcal{X}}$  denote the set of all functions  $\mathcal{X} \rightarrow \mathcal{Y}$ . Then Full GPAR (FPGAR) models each  $p(g_i | \mathbf{g}_{1:i-1}, \mathbf{y}_{1:i-1})$  with a GP  $f_i \sim \mathcal{GP}(0, k_{f_i})$  over the input space  $\mathcal{X} \times (\mathcal{Y}^{\mathcal{X}})^{2(i-1)}$ :

$$g_i(\mathbf{x}) | f_i, \mathbf{g}_{1:i-1}, \mathbf{y}_{1:i-1} = f_i(\mathbf{x}, \mathbf{g}_{1:i-1}, \mathbf{y}_{1:i-1}).$$

<sup>1</sup>That is,  $\mathbf{g}$  is a random variable whose realisations are functions  $\mathcal{X} \rightarrow \mathcal{Y}^M$ .

Thus, in the most general case, the stochastic process  $f_i$  depends upon  $\mathbf{x}$  as well as the entirety of each of the preceding stochastic processes  $\mathbf{g}_{1:i-1}$  and  $\mathbf{y}_{1:i-1}$ ; for example,  $g_i$  might be a integral transform of an element of  $\mathbf{g}_{1:i-1}$  or  $\mathbf{y}_{1:i-1}$ . Figure 2a depicts the corresponding graphical model in the case of three outputs.

The choice of kernels for  $\mathbf{f}$  is crucial to FGPARG, as they determine the types of relationships between inputs and outputs that can be learned. Furthermore, as a consequence of assuming that the conditionals  $p(g_i | \mathbf{g}_{1:i-1}, \mathbf{y}_{1:i-1})$  are Gaussian, different factorisations of the joint  $p(\mathbf{y}, \mathbf{g})$  yield different FGPARG models; hence, in specifying FGPARG, one must choose an ordering of the outputs  $\mathbf{y}$ .

The described model will generally not be tractable. Instead, consider the simplification where  $g_i(\mathbf{x})$  depends only on  $\mathbf{y}_{1:i-1}(\mathbf{x}')$ , only for  $\mathbf{x} = \mathbf{x}'$ :  $f_i(\mathbf{x}, \mathbf{g}_{1:i-1}, \mathbf{y}_{1:i-1}) = f_i(\mathbf{x}, \mathbf{y}_{1:i-1}(\mathbf{x}))$ . Note that this simplification simplifies the input space over which  $f_i$  is defined. We refer to the resulting model as GPAR, whose graphical model is depicted in Figure 2b.

For a certain type of data set, which we call *closed-downwards*, the graphical model for GPAR expresses conditional independence properties that make inference unusually easy: for closed-downwards data sets, the posterior of  $y_i$  is conditionally independent of observations of  $y_j$  if  $j > i$ . Formally, let a data set  $\mathcal{D} = \{y_i(\mathbf{x}) \text{ for some } i \text{ and some } \mathbf{x}\}$  be a random variable representing our observations. Call  $\mathcal{D}$  *closed downwards* if  $y_j(\mathbf{x}) \in \mathcal{D}$  implies that  $y_i(\mathbf{x}) \in \mathcal{D}$  for all  $i \leq j$ .

**Theorem 1.** Let  $\mathcal{D}$  be closed downwards,  $y_j(\mathbf{x}) \in \mathcal{D}$ , and  $i < j$ . Then  $y_i \perp y_j(\mathbf{x}) | \mathcal{D}$ .

The proof of Theorem 1 can be found in Appendix A. Letting  $\mathcal{X}_{\mathcal{D}} = \{\mathbf{x} : y_i(\mathbf{x}) \in \mathcal{D} \text{ for some } i\}$  and  $\mathbf{y}_{1:i}(\mathcal{X}_{\mathcal{D}}) = \{\mathbf{y}_{1:i}(\mathbf{x}) : \mathbf{x} \in \mathcal{X}_{\mathcal{D}}\}$ , we have by Theorem 1 that

$$\begin{aligned} p(\mathbf{y} | \mathcal{D}) &= \prod_{i=1}^M p(y_i | \mathbf{y}_{1:i-1}, \mathcal{D}) \\ &= \prod_{i=1}^M p(y_i | \mathbf{y}_{1:i-1}, \mathbf{y}_{1:i}(\mathcal{X}_{\mathcal{D}})), \end{aligned} \quad (2)$$

where each posterior  $p(y_i | \mathbf{y}_{1:i-1}, \mathbf{y}_{1:i}(\mathcal{X}_{\mathcal{D}}))$  is obtained from simply conditioning  $f_i$  on input–output pairs  $\{((\mathbf{x}, \mathbf{y}_{1:i-1}(\mathbf{x})), y_i(\mathbf{x})) : \mathbf{x} \in \mathcal{X}_{\mathcal{D}}\}$ , taking into account the noise introduced by  $y_i | g_i$ . Hence, denoting  $N = |y_1(\mathcal{X}_{\mathcal{D}})|$ , computing  $p(y_i | \mathbf{y}_{1:i-1}, \mathbf{y}_{1:i}(\mathcal{X}_{\mathcal{D}}))$  takes  $\mathcal{O}(N^3 + N^2(D + i))$  time and requires  $\mathcal{O}(N^2)$  space, meaning that computing  $p(\mathbf{y} | \mathcal{D})$  takes  $\mathcal{O}(N^3DM + N^2(M + D)M)$  time and requires  $\mathcal{O}(N^2M)$  space. Fur-

thermore, it holds that

$$p(\mathcal{D}) = \prod_{i=1}^M p(y_i(\mathcal{X}_{\mathcal{D}}) | y_{1:i-1}(\mathcal{X}_{\mathcal{D}})), \quad (3)$$

where the  $i^{\text{th}}$  factor is simply the likelihood of input–output pairs  $\{((\mathbf{x}, \mathbf{y}_{1:i-1}(\mathbf{x})), y_i(\mathbf{x})) : \mathbf{x} \in \mathcal{X}_{\mathcal{D}}\}$  under  $f_i$ , taking into account the noise introduced by  $y_i | g_i$ . Therefore,  $p(y_i(\mathcal{X}_{\mathcal{D}}) | y_{1:i-1}(\mathcal{X}_{\mathcal{D}}))$  depends on and only on the hyperparameters of  $f_i$  and  $y_i | g_i$ , so optimising  $p(\mathcal{D})$  corresponds to optimising each  $p(y_i(\mathcal{X}_{\mathcal{D}}) | y_{1:i-1}(\mathcal{X}_{\mathcal{D}}))$  individually, meaning that learning in GPAR can be performed by learning each  $f_i$  individually. Note that for each  $y_i(\mathbf{x}) \in \mathcal{D}$ , its location  $(\mathbf{x}, \mathbf{y}_{1:i}(\mathbf{x}))$  is in  $\mathcal{D}$ , because  $\mathcal{D}$  is closed downwards.

In summary, GPAR forms a multi-output model in which noisy output  $y_i(\mathbf{x})$  is generated from the preceding noisy outputs  $\mathbf{y}_{1:i-1}(\mathbf{x})$  at that particular  $\mathbf{x}$ , where the relationships between inputs and outputs can be specified through the kernels for  $\mathbf{f}$ . Importantly, given data that is closed downwards, inference and learning in GPAR can be performed efficiently.

Finally, note that Theorem 1 only depends on the graphical model depicted in Figure 2b. This means that one can implement the conditionals in Equation (1) with *any* models—not just GPs—that satisfy  $p(g_i(\mathbf{x}) | \mathbf{g}_{1:i-1}, \mathbf{y}_{1:i-1}) = p(g_i(\mathbf{x}) | \mathbf{y}_{1:i-1}(\mathbf{x}))$ , and training and inference still decouples; in other words, training and inference of Equation (1) is tractable and decouples if  $p(g_i(\mathbf{x}) | \mathbf{g}_{1:i-1}, \mathbf{y}_{1:i-1}) = p(g_i(\mathbf{x}) | \mathbf{y}_{1:i-1}(\mathbf{x}))$  and training and inference of each  $p(g_i | \mathbf{y}_{1:i-1})$  is tractable.

## 2.1 Equivalent Models

To get insight into the class of models that FGPARG defines, we study two equivalent models. For functions  $\mathbf{A}, \mathbf{B}: \mathcal{X} \times (\mathcal{Y}^{\mathcal{X}})^M \rightarrow \mathcal{Y}^M$ , define composition  $\circ$  as follows:  $(\mathbf{A} \circ \mathbf{B})(\mathbf{x}, \mathbf{y}) = \mathbf{A}(\mathbf{x}, \mathbf{B}(\cdot, \mathbf{y}))$ . Note that  $\circ$  is well-defined and associative. For a function  $\mathbf{u}: \mathcal{X} \rightarrow \mathcal{Y}^M$ , denote  $\mathbf{A} \circ \mathbf{u}: \mathcal{X} \rightarrow \mathcal{Y}^M$ ,  $\mathbf{A} \circ \mathbf{u} = \mathbf{A}(\cdot, \mathbf{u})$ ; that is,  $\mathbf{A}$  transforms a function  $\mathbf{u}$  into another function  $\mathbf{A} \circ \mathbf{u}$ , which is why we denote  $\mathbf{A}$  with a capital letter. Again, note that  $(\mathbf{A} \circ \mathbf{B}) \circ \mathbf{u} = \mathbf{A} \circ (\mathbf{B} \circ \mathbf{u})$ . Furthermore, denote  $\underbrace{\mathbf{A} \circ \dots \circ \mathbf{A}}_{n \text{ times}} = \mathbf{A}^n$ .

**Lemma 1** (Nonlinear Equivalent Model). Let  $\mathbf{A}$  be an  $M$ -dimensional vector-valued process over  $\mathcal{X} \times (\mathcal{Y}^{\mathcal{X}})^M$ , each  $A_i$  drawn from  $\mathcal{GP}(0, k_{A_i})$  independently, and let  $\mathbf{u}$  be an  $M$ -dimensional vector-valued process over  $\mathcal{X}$ , each  $u_i$  drawn from  $\mathcal{GP}(0, k_{u_i})$  independently. Furthermore, let  $A_i(\mathbf{x}, \mathbf{y}): \mathcal{X} \times (\mathcal{Y}^{\mathcal{X}})^M \rightarrow \mathcal{Y}$  depend only on  $(\mathbf{x}, \mathbf{y}_{1:i-1})$ , meaning that  $k_{A_i} =$

$k_{A_i}(\mathbf{x}, \mathbf{y}_{1:i-1}, \mathbf{x}', \mathbf{y}'_{1:i-1})$ , and let  $A_1 = 0$ . Denote  $\mathbf{T}\mathbf{f} = \mathbf{u} + \mathbf{A} \circ \mathbf{f}$ , and denote  $N$  consecutive applications of  $\mathbf{T}$  by  $\mathbf{T}^N$ . Then

$$\mathbf{g} | \mathbf{A}, \mathbf{u} = \mathbf{T}^{M-1} \mathbf{u} \iff g_i | \mathbf{g}_{1:i-1} \sim \mathcal{GP}(0, k_{u_i} + k_{A_i}(\cdot, \mathbf{g}_{1:i-1}, \cdot, \mathbf{g}_{1:i-1})).$$

**Lemma 2** (Linear Equivalent Model). Suppose that  $\mathbf{A}$  was instead generated from

$$\mathbf{A}(\mathbf{x}, \mathbf{y}) | \hat{\mathbf{A}} = \int \hat{\mathbf{A}}(\mathbf{x} - \mathbf{z}) \mathbf{y}(\mathbf{z}) d\mathbf{z},$$

where  $\hat{\mathbf{A}}$  is an  $(M \times M)$ -matrix-valued process over  $\mathcal{X}$ , each  $\hat{A}_{i,j}$  drawn from  $\mathcal{GP}(0, k_{\hat{A}_{i,j}})$  independently if  $i > j$  and  $\hat{A}_{i,j} = 0$  otherwise. Then

$$\mathbf{g} | \mathbf{A}, \mathbf{u} = \left( \sum_{i=0}^{M-1} \mathbf{A}^i \right) \circ \mathbf{u} \iff g_i | \mathbf{g}_{1:i-1} \sim \mathcal{GP}(0, k_{u_i} + k_{A_i}(\cdot, \mathbf{g}_{1:i-1}, \cdot, \mathbf{g}_{1:i-1})), \quad (4)$$

where

$$\begin{aligned} k_{A_i}(\mathbf{x}, \mathbf{g}_{1:i-1}, \mathbf{x}', \mathbf{g}'_{1:i-1}) \\ = \sum_{j=1}^{i-1} \int k_{\hat{A}_{i,j}}(\mathbf{x} - \mathbf{z}, \mathbf{x}' - \mathbf{z}') g_j(\mathbf{z}) g'_j(\mathbf{z}') d\mathbf{z} d\mathbf{z}'. \end{aligned}$$

The proofs of Lemmas 1 and 2 can be found in Appendix B. Note that the right-hand sides of the equivalences in Lemmas 1 and 2 are equivalent to FPGAR with

$$\begin{aligned} k_{f_i}(\mathbf{x}, \mathbf{g}_{1:i-1}, \mathbf{y}_{1:i-1}, \mathbf{x}', \mathbf{g}'_{1:i-1}, \mathbf{y}'_{1:i-1}) \\ = k_{u_i}(\mathbf{x}, \mathbf{x}') + k_{A_i}(\mathbf{x}, \mathbf{g}_{1:i-1}, \mathbf{x}', \mathbf{g}'_{1:i-1}). \end{aligned}$$

As mentioned before, the kernels for  $\mathbf{f}$  determine the types of relationships between inputs and outputs that can be learned. Lemmas 1 and 2 make this explicit: Lemma 1 shows that FGPARG can recover a model where  $M$  latent GPs  $\mathbf{u}$  are repeatedly composed with another latent GP  $\mathbf{A}$ , where  $\mathbf{A}$  has a particular dependency structure, and Lemma 2 shows that FGPARG can recover a model where  $M$  latent GPs  $\mathbf{u}$  are linearly transformed, where the linear transform  $\mathbf{T} = \sum_{i=0}^{M-1} \mathbf{A}^i$  is lower triangular and may vary with the input.

In Lemma 2, note that it is not restrictive that  $\mathbf{T}$  is lower triangular: Suppose that  $\mathbf{T}$  were dense. Then, letting  $\mathbf{g} | \mathbf{T}, \mathbf{u} = \mathbf{T} \circ \mathbf{u}$ ,  $\mathbf{g} | \mathbf{T}$  is jointly Gaussian. Hence  $g_i | \mathbf{g}_{1:i-1}$ ,  $\mathbf{T}$  is a GP whose mean linearly depends upon  $\mathbf{g}_{1:i-1}$  via  $\mathbf{T}$ , meaning that  $g_i | \mathbf{g}_{1:i-1}$  is of the form of Equation (4) where  $k_{u_i}$  may be more complicated. If,

however,  $\hat{\mathbf{A}}(\mathbf{z}) = \delta(\mathbf{z})\mathbf{B}$  for some random  $(M \times M)$ -matrix  $\mathbf{B}$ , each  $B_{i,j}$  drawn from  $\mathcal{N}(0, \sigma_{B_{i,j}}^2)$  if  $i > j$  and  $B_{i,j} = 0$  otherwise, then it is restrictive that  $\mathbf{T}$  is lower triangular: In this case,  $\mathbf{g}(\mathbf{x}) | \mathbf{B}, \mathbf{u} = \sum_{i=0}^{M-1} \mathbf{B}^i \mathbf{u}(\mathbf{x})$ . If  $\mathbf{T} = \sum_{i=0}^{M-1} \mathbf{B}^i$  were dense, then, letting  $\mathbf{g} | \mathbf{T}, \mathbf{u} = \mathbf{T}\mathbf{u}$ ,  $\mathbf{g}$  can be represented with Lemma 2 if and only if  $\mathbf{g} | \mathbf{T}$ 's covariance can be diagonalised by a constant, invertible, lower-triangular matrix. This condition does not hold in general, as Lemma 3 from Appendix C proves.

## 2.2 Kernel Design

Now that we have gained some insight into the class of models that FGPARG defines, we can use this insight to narrow down the choice of kernels for  $\mathbf{f}$  in GPAR. Define two specialisations of GPAR:

$$\begin{aligned} k_{f_i}(\mathbf{x}, \mathbf{y}_{1:i-1}(\mathbf{x}), \mathbf{x}', \mathbf{y}_{1:i-1}(\mathbf{x}')) \\ = k_i(\mathbf{x}, \mathbf{x}') + \sum_{j=1}^{i-1} k_{i,j}(\mathbf{x}, \mathbf{x}') y_j(\mathbf{x}) y_j(\mathbf{x}'), \end{aligned} \quad (\text{GPAR-L})$$

$$\begin{aligned} k_{f_i}(\mathbf{x}, \mathbf{y}_{1:i-1}(\mathbf{x}), \mathbf{x}', \mathbf{y}_{1:i-1}(\mathbf{x}')) \\ = k_{i,x}(\mathbf{x}, \mathbf{x}') + k_{i,y}(\mathbf{y}_{1:i-1}(\mathbf{x}), \mathbf{y}_{1:i-1}(\mathbf{x}')), \end{aligned} \quad (\text{GPAR-NL})$$

where  $k_i$ ,  $k_{i,j}$ ,  $k_{i,x}$ , and  $k_{i,y}$  can be any kernels. Based on Lemmas 1 and 2, we conclude that GPAR-L models instantaneous, input-varying, linear correlations between outputs, whereas GPAR-NL models instantaneous, input-constant, nonlinear correlations between outputs.

Furthermore, one might believe that  $g_i(\mathbf{x})$  depends on  $\mathbf{g}_{1:i-1}(\mathbf{x})$  instead of  $\mathbf{y}_{1:i-1}(\mathbf{x})$ ; in other words, we might believe the outputs to have *latent dependencies*. Were one to change Figure 2b accordingly, then, unfortunately, there are no longer the conditional independence properties that make inference efficient. One possible mitigating solution is to employ an input transformation that estimates  $\mathbf{g}_{1:i-1}(\mathbf{x})$  from  $\mathbf{y}_{1:i-1}(\mathbf{x})$ :

$$\begin{aligned} k_{f_i}(\mathbf{x}, \mathbf{y}_{1:i-1}(\mathbf{x}), \mathbf{x}', \mathbf{y}'_{1:i-1}(\mathbf{x}')) \\ = \hat{k}_{f_i}(\mathbf{x}, \mathbb{E}[\mathbf{g}_{1:i-1}(\mathbf{x}) | \mathbf{y}_{1:i-1}(\mathbf{x})], \\ \mathbf{x}', \mathbb{E}[\mathbf{g}'_{1:i-1}(\mathbf{x}') | \mathbf{y}'_{1:i-1}(\mathbf{x}')]); \end{aligned} \quad (5)$$

that is,  $f_i$  is now a function of the posterior mean of the preceding processes, as opposed to their observed values. Let D-GPAR-\* denote models that use this approach. Note that the conditional expectation employed here could discard useful information and thereby diminish performance. Also, note that model parameters of earlier outputs are now hyperparameters of later outputs, which couples the outputs during learning.

### 3 Related Work

Multi-output GP models are usually constructed by letting the  $i^{\text{th}}$  output,  $g_i$ , be a linear transform of a number of latent GPs,  $\{u_i\}_i$ :  $g_i(\mathbf{x}) = \sum_k \int A_{i,k}(\mathbf{z}, \mathbf{x}) u_k(\mathbf{z}) d\mathbf{z}$ . If  $A_{i,k}(\mathbf{z}, \mathbf{x}) = A_{i,k} \delta(\mathbf{z} - \mathbf{x})$  where  $\delta$  is the Dirac delta function, then  $\mathbf{f}(\mathbf{x})$  depends on  $\mathbf{u}(\mathbf{z})$  only for  $\mathbf{z} = \mathbf{x}$ ; examples of such models are (Goovaerts, 1997; Stein, 1999; Wackernagel, 2003; Teh & Seeger, 2005; Bonilla et al., 2008; Nguyen & Bonilla, 2014). Other models let  $A_{i,k}(\mathbf{z}, \mathbf{x}) = A_{i,k}(\mathbf{z}) \delta(\mathbf{z} - \mathbf{x})$ , such as (Wilson et al., 2012), meaning that the linear dependence of  $\mathbf{f}(\mathbf{x})$  on  $\mathbf{u}(\mathbf{x})$  varies with  $\mathbf{x}$ . Furthermore, (Álvarez et al., 2009; Álvarez & Lawrence, 2009, 2011) treat  $A_{i,k}(\mathbf{z}, \mathbf{x})$  parametrically, whilst (Bruinsma, 2016) does so nonparametrically. Lemma 2 shows that (F)GPARG can recover similar models, where the mixing matrix  $\mathbf{A}$  is lower triangular instead.

Orthogonal to the above work is that presented by Wilson et al. (2016), which induces correlations between outputs by additionally applying a nonlinear transform to the input domain. A similar idea is employed in Deep Gaussian Processes (Damianou, 2014), where multiple GPs are composed to obtain an expressive model. (F)GPARG can be considered a Deep GP, where the composed latent GPs form the outputs of the model, and Lemma 1 shows that (F)GPARG can recover a Deep GP with a particular composition structure. Furthermore, the Gaussian Process Network (Friedman & Nachman, 2000) is similar to GPARG, but it was developed for identifying causal dependencies between variables in a probabilistic graphical models context, rather than multi-output regression. The work in (Yuan, 2011) also discusses a model similar to GPARG, but specifies a different generative procedure for the outputs.

Finally, the multi-fidelity modelling literature is closely related. Whereas in the multi-output regression task we are interested in predicting all output dimensions, the multi-fidelity task is concerned with predicting a high-fidelity function, while incorporating information from observations from various levels of fidelity. The idea of iteratively conditioning on lower fidelity models in the construction of higher fidelity ones has been a well-used strategy (Kennedy & O’Hagan, 2000; Le Gratiet & Garnier, 2014). The model presented by (Perdikaris et al., 2017) is nearly identical to GPARG applied in the multi-fidelity framework, but applications outside this setting have not been considered.

### 4 Synthetic Data Experiments

GPARG is well-suited for problems where there is a strong functional relationship between output dimen-

sions, and for problems where observation noise is correlated between output dimensions. In this section we demonstrate GPARG’s ability to model both. We parametrise GPARG’s kernel using a rational quadratic (RQ) kernel (Rasmussen & Williams, 2006) for the first output dimension and a sum of two RQ kernels  $k_1(x, x') + k_2((x, y), (x', y'))$  for the second and third dimensions, where  $y$  consists of the previous output dimensions. GPARG is constructed using the natural ordering of outputs.

#### 4.1 Learning Functional Structure

Consider three outputs  $y_1, y_2$ , and  $y_3$  depending nonlinearly on each other as follows:

$$\begin{aligned} y_1(x) &= -\sin(10\pi(x+1))(2x+1) - x^4 + \epsilon_1, \\ y_2(x) &= \cos^2(y_1(x)) + \sin(3x) + \epsilon_2, \\ y_3(x) &= y_2(x)y_1^2(x) + 3x + \epsilon_3, \end{aligned}$$

where  $\epsilon_1, \epsilon_2, \epsilon_3 \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$ . With a large enough training set, GPARG and independent GPs perform similarly with these data: enough data is present to learn the complicated functional structure on  $x$  directly without leveraging the shared structure between the functions. With fewer data points—Figure 3 shows plots of independent GPs (IGP) and GPARG fit to approximately 40 data points from  $y_1, y_2$  and  $y_3$ —observe that GPARG is able to learn  $y_2$ ’s dependence on  $y_1$ , and  $y_3$ ’s dependence on  $y_1$  and  $y_2$ , resulting in better predictive performance compared to independent GPs.

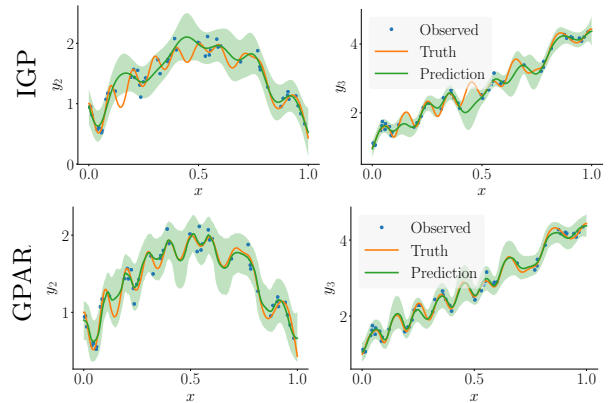


Figure 3: Learning functional structure across output dimensions

#### 4.2 Learning Noise Structure

Consider three schemes in which two outputs are observed under various noise correlations:  $y_1(x) = f_1(x) + \epsilon_1$  and

$$(1) \quad y_2(x) = f_2(x) + \sin^2(2\pi x)\epsilon_1 + \cos^2(2\pi x)\epsilon_2;$$

$$(2) \quad y_2(x) = f_2(x) + \sin(\pi\epsilon_1) + \epsilon_2; \text{ and}$$

$$(3) \quad y_2(x) = f_2(x) + \sin(\pi x)\epsilon_1 + \epsilon_2;$$

where  $\epsilon_1, \epsilon_2 \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$ , and  $f_1$  and  $f_2$  are complicated, nonlinear functions:

$$f_1(x) = -\sin(10\pi(x+1))/(2x+1) - x^4,$$

$$f_2(x) = \frac{1}{5}e^{2x}(\theta_1 \cos(\theta_2\pi x) + \theta_3 \cos(\theta_4\pi x)) + \sqrt{2}x.$$

All three schemes have i.i.d. homoscedastic Gaussian noise in  $y_1$ , making  $y_1$  easy to learn for GPAR. However, the noise in  $y_2$  depends on that in  $y_1$  and can be heteroscedastic. The task for GPAR is to learn this complicated noise structure. Figure 4 shows the noise produced by the schemes (representing truth) and the noise sampled from GPAR’s posterior, and Figure 5 shows  $y_2$  samples produced by the schemes (representing truth) and  $y_2$  samples from GPAR’s posterior. Colours in Figure 5 correspond to the magnitude of corresponding noise sample (i.e. at the same  $x$ ) in  $y_1$ ; hence, colours visualise the correlation between the noises in  $y_1$  and  $y_2$ . For a particular  $x$ , if the colour pattern is preserved, then the model has successfully captured how the noise in  $y_1(x)$  correlates to that in  $y_2(x)$ .

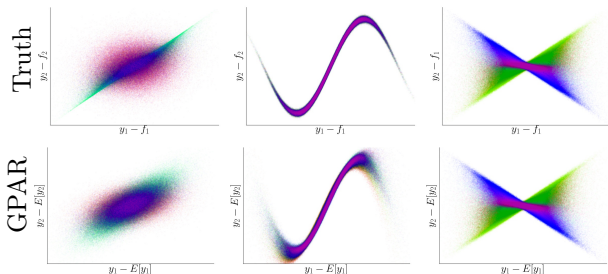


Figure 4: Correlation between the sample residues (deviation from the mean) for  $y_1$  and  $y_2$ . Left, centre and, right plots correspond to schemes (1), (2) and (3) respectively. Samples are coloured according to input value  $x$ ; that is, all samples for a particular  $x$  have the same colour.

Figures 4 and 5 show that GPAR is able to learn various noise structures. Considering Figure 5, the left plots show that GPAR can roughly learn a structure where the correlation between the noise in  $y_1$  and in  $y_2$  depends on  $x$ ; the middle plots show that GPAR can learn noise in  $y_2$  that correlates to that in  $y_1$  in a complicated manner; and the right plots show that GPAR can learn heteroscedastic noise.

GPAR is able to model complicated noise structure by letting the kernel of  $y_2$  also depend on  $y_1$ . By letting the component in the kernel of  $y_2$  that models the

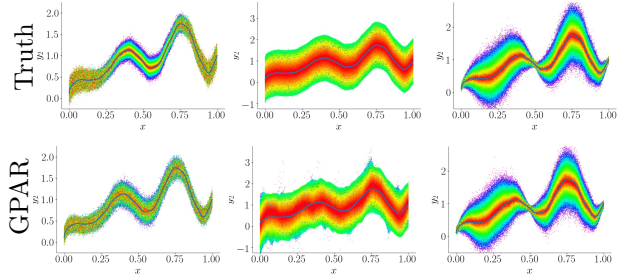


Figure 5: Visualisation of  $y_2$  samples. Left, centre and, right plots correspond to schemes (1), (2) and (3) respectively.

complicated noise structure be additive, we can additionally decompose GPAR’s predictions into a part that models just the noise, and a part that models just the underlying function. Figure 6 visualises this decomposition for predictions on data generated by  $y_1 = f_1 + \epsilon_1$  and  $y_2 = f_2 + \epsilon_2$  where  $\epsilon_1$  and  $\epsilon_2$  are jointly Gaussian and correlated. The decomposition provides insight in GPAR’s predictions: one component models the function uncertainty while the other models the uncertainty introduced by correlated noise. Furthermore, the different parts of the prediction could be used in more complicated scenarios to estimate particular quantities of interest.

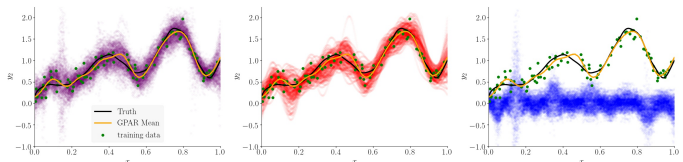


Figure 6: Decomposition of GPAR’s predictions. Posterior samples of  $y_2$  (left), the additive parts of those samples that correspond to  $f_2$  (middle), and the additive parts of those samples that correspond to the noise ( $y_2 - f_2$ ; right).

## 5 Real-World Data Experiments

In this section we evaluate GPAR’s performance and compare to other models on three standard data sets commonly used to evaluate multi-output models. We also consider a recently-introduced data set in the field of Bayesian optimisation, which is a downstream application area that could benefit from GPAR.

Table 1 lists the models against which we compare GPAR. We always compare against IGP and CK, ICM, SLFM, and CGP, and compare against CMOGP and GPRN if results for the considered task are available. Since CK and ICM are much simplified versions of SLFM (Álvarez et al., 2010; Goovaerts, 1997) and CGP is an approximation to SLFM, we sometimes omit

Acronym	Model
IGP	Independent GPs
CK	Cokriging (Goovaerts, 1997; Stein, 1999; Wackernagel, 2003)
ICM	Intrinsic Coregionalisation Model (Goovaerts, 1997; Stein, 1999; Wackernagel, 2003)
SLFM	Semiparametric Latent Factor Model (Teh & Seeger, 2005)
CGP	Collaborative Multi-Output GPs (Nguyen & Bonilla, 2014)
CMOGP	Convolved Multi-output GP Model (Álvarez & Lawrence, 2011; Álvarez et al., 2010)
GPRN	GP Regression Network (Wilson et al., 2012)
[D-]GPAR-[L-][N]L	[Denoising] GPAR with [linear and] [non]linear output dependencies (Equations (5), (GPAR-L) and (GPAR-NL))

Table 1: List of models against which GPAR is compared

results for CK, ICM, and CGP.

### 5.1 Electroencephalograms

The electroencephalogram (EEG) data set<sup>2</sup> consists of 256 voltage measurements from 7 electrodes placed on a subject’s scalp whilst the subject is shown a certain image; (Zhang et al., 1995) describes the data collection process in detail. In particular, we use frontal electrodes FZ and F1–F6 from the first trial on control subject 337. The task is to predict the last 100 samples for electrodes FZ, F1, and F2, given that the first 156 samples of FZ, F1, and F2 and the whole signals of F3–F6 are observed. Performance is measured with the standardised mean squared error (SMSE), mean log loss (MLL) (Rasmussen & Williams, 2006), and training time (TT); the models were trained on a late-2013 MacBook Pro.

Model	SMSE	MLL	TT
IGP	1.75	2.60	2 sec
SLFM	1.06	4.00	11 min
GPAR-NL	<b>0.26</b>	<b>1.63</b>	5 sec

Table 2: Experimental results for the EEG data set for IGP, the SLFM with four latent dimensions, and GPAR.

Figure 7 visualises predictions for electrode F2, and Ta-

<sup>2</sup>The EEG data set can be downloaded at <https://archive.ics.uci.edu/ml/datasets/eeg+database>.

ble 2 quantifies the results. We observe that GPAR-NL outperforms independent in terms of SMSE and MLL; note that independent GPs completely fail to provide an informative prediction. Furthermore, independent GPs were trained in two seconds, and GPAR-NL took only a three more seconds; in comparison, training SLFM took 11 minutes.

### 5.2 Metal Concentration Measurements in the Swiss Jura

The Jura data set comprises metal concentration measurements collected from the topsoil in a 14.5 km<sup>2</sup> region of the Swiss Jura.<sup>3</sup> We follow the experimental protocol in (Goovaerts, 1997) also followed in (Álvarez & Lawrence, 2011): The training data comprises 259 data points distributed spatially with three output variables—nickel, zinc, and cadmium—and 100 additional data points for which only two of the three outputs—nickel and zinc—are observed. The task is to predict cadmium at the locations of those 100 additional data. Performance is evaluated with the mean absolute error (MAE).

Model	IGP	CK <sup>†</sup>	ICM	SLFM	CMOGP <sup>†</sup>
MAE	0.5739	0.51	0.4601	0.4606	0.4552
MAE*	0.5753		0.4114	0.4145	

Model	GPRN <sup>†</sup>	GPAR-NL	D-GPAR-NL
MAE	0.4525	0.4324	<b>0.4114</b>
MAE*	0.4040	0.4168	<b>0.3996</b>

Table 3: Results for the Jura data set for IGP, cokriging (CK) and ICM with two latent dimensions, the SLFM with two latent dimensions, CMOGP, GPRN, and GPAR. \* These results are obtained by first log-transforming the data, then performing prediction, and finally transforming the predictions back to the original domain. <sup>†</sup> These numbers are taken from (Wilson, 2014).

Similarity between the colour patterns in Figure 8 indicates that concentrations of the three minerals are positively correlated. The comparatively poor performance of independent GPs in Table 3 highlights the importance of exploiting this relationship. Furthermore, Table 3 shows that D-GPAR-NL significantly outperforms the other models, achieving a new state-of-the-art result.

<sup>3</sup>The data can be downloaded at <https://sites.google.com/site/goovaertspierre/pierregoovaertswesite/download/>.

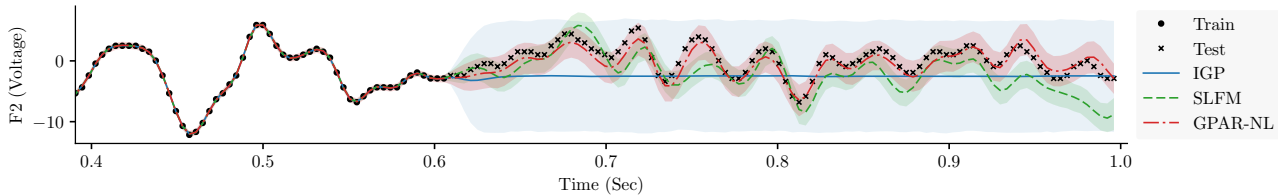


Figure 7: Predictions for electrode F2 from the EEG data set

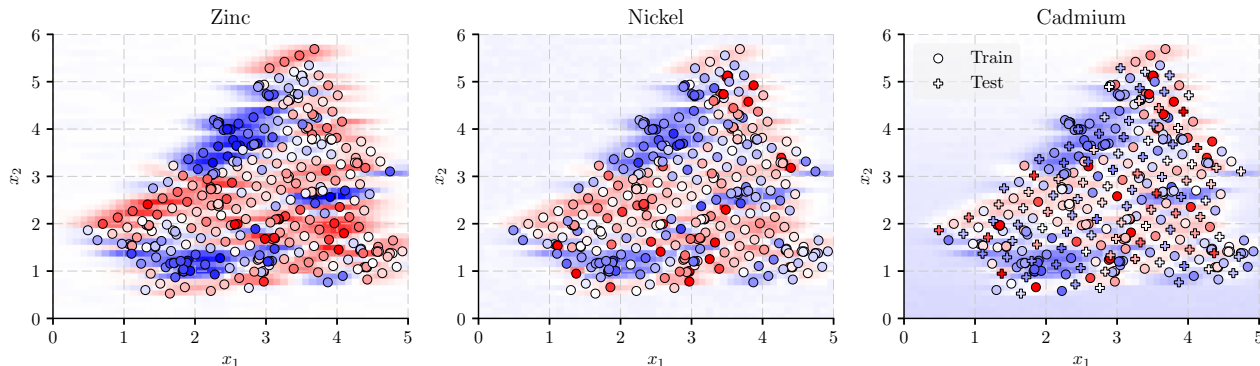


Figure 8: The Jura data set and GPAR’s prediction for it. Colours represent rescaled concentration levels; different scales are used for each metal. Circles ( $\circ$ ) correspond to training points, and pluses ( $\oplus$ ) correspond to test points; the points’s inner fill colours represent the true concentration level at those locations. Finally, the background colours represent GPAR’s prediction.

### 5.3 Exchange Rates of International Currencies and Precious Metals

The exchange rates data set consists of the daily exchange rate w.r.t. USD of the top ten international currencies (CAD, EUR, JPY, GBP, CHF, AUD, HKD, NZD, KRW, and MXN) and three precious metals (gold, silver, and platinum) in the year 2007.<sup>4</sup> The task is to predict CAD on days 50–100, JPY on days 100–150, and AUD on days 150–200, given that CAD is observed on days 1–49 and 101–251, JPY on days 1–49 and 151–251, and AUD on days 1–49 and 201–251; and that all other currencies are observed throughout the whole year. This comprises the same test set as in (Nguyen & Bonilla, 2014), against which we compare performance, but a smaller training set. (Nguyen & Bonilla, 2014) additionally observes JPY on days 50–99 and AUD on days 50–149; accordingly, the task we present here is a slightly harder one than that in (Nguyen & Bonilla, 2014), but their results can be compared. Performance is measured with the standardised mean squared error (SMSE) (Rasmussen & Williams, 2006).

Figure 9 visualises GPAR’s prediction for data set, and Table 4 quantifies the result. We observe that GPAR

<sup>4</sup>The exchange rates data set can be downloaded at <http://fx.sauder.ubc.ca>.

Model	IGP*	CMOGP*	CGP*	GPAR-L-NL
SMSE	0.5996	0.2427	0.2125	<b>0.0519</b>

Table 4: Experimental results for the exchange rates data set for IGP, CMOGP, CGP, and GPAR. \* These numbers are taken from (Nguyen & Bonilla, 2014).

significantly outperforms all other models, which shows visually by comparing with the plots in (Nguyen & Bonilla, 2014). Note that GPAR outperforms all other models despite being presented a harder task.

### 5.4 Objective Surface Modelling for Bayesian Optimisation

We finally consider the problem of modelling the objective surface of the validation error of a multi-layer perceptron (MLP) on the MNIST data, trained using categorical cross-entropy, and set as a function of six hyperparameters: the number of hidden layers, the number of neurons per hidden layer, the dropout rate, the learning rate to use with the ADAM optimizer, the  $L_1$  weight penalty, and the  $L_2$  weight penalty. This experiment was implemented using code made available by Hernández-Lobato (2016). An improved model for the objective surface could translate directly into im-



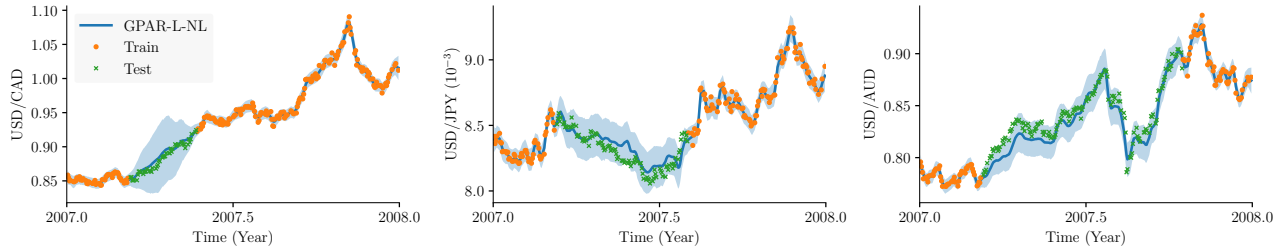


Figure 9: Visualisation of the exchange rates data set and GPAR’s prediction for it

proved performance in Bayesian Optimisation (Snoek et al., 2012), as a better model of the objective surface will result in a more informed search of the hyperparameter space.

To generate a data set we sample 291 sets of hyperparameters randomly from a rectilinear grid and train the MLP for 21 epochs under each set of hyperparameters, recording the validation performance after 1, 5, 11, 16 and 21 epochs. We construct a training set of 175 of these hyperparameter settings and, crucially, discard roughly 30% of the validation performance results at 5 epochs at random, and again discard roughly 30% of those results at 11 epochs, and so forth. The resulting data set has 175 labels after 1 epoch, 124 after 5, 88 after 11, 64 after 15 and 44 after 21, simulating the partial completion of the majority of runs. Crucially, a Bayesian Optimisation system typically exploits only completed training runs to inform the objective surface, whereas GPAR can also exploit partially complete runs.

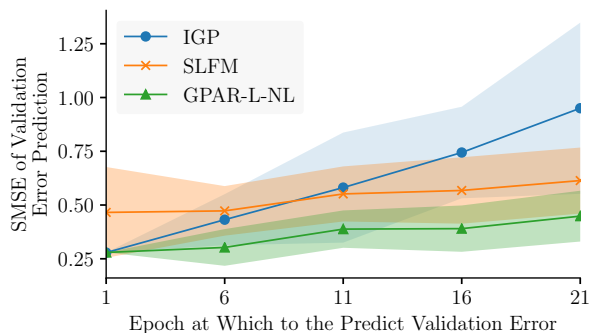


Figure 10: Results for the machine learning data set for a GP, the SLFM with two latent dimensions, and GPAR.

The results presented in Figure 10 show the SMSE in predicting validation performance at each epoch using GPAR, the SLFM, and independent GPs on the test set, averaged over 10 seeds for the pseudo-random number generator used to select which outputs from the training set to discard. GPs trained independently to predict performance after a particular number of epochs

perform worse than the SLFM and GPAR, which both have learned to exploit the extra information available to them. In turn, GPAR performs noticeably better than the SLFM.

## 6 Conclusion and Future Work

This paper introduced GPAR: a flexible, tractable, and interpretable approach to multi-output GP regression. GPAR can model nonlinear relationships between outputs, capture correlations in the noise, and scale to a large number of output dimensions. In effect, GPAR transforms problems high-dimensional data modelling problems into set of single-output modelling problems, which are the bread and butter of the GP approach. GPAR was rigorously tested on a variety of synthetic and real-world problems, consistently outperforming existing GP models for multi-output regression. Insights into the structure of the data can be gained by decomposing GPAR’s posterior over kernel components which could be developed into a useful tool in automatic structure discovery in data (Lloyd et al., 2014). In this light, we believe particularly exciting future applications of GPAR are modelling of environmental phenomena and improving the data-efficiency of existing Bayesian Optimisation tools (Snoek et al., 2012), through providing an improved model of the objective surface. GPAR is trivially compatible with the wide array of GP approximation techniques designed to scale inference to a large number of observations (Titsias, 2009; Hensman et al., 2013; Saatçi, 2012; Cunningham et al., 2008). Another area for future work are extensions to arbitrarily-structured missing output data. This would remove the constraints currently imposed on the ordering of the outputs when some observations are missing. Leveraging inference schemes for Deep GPs could provide a promising starting point here. In the other direction, GPAR could be harnessed to provide initialisations for Deep GP inference schemes.

## Acknowledgements

Richard E. Turner is supported by Google as well as EPSRC grants EP/M0269571 and EP/L000776/1.

## References

- Álvarez, Mauricio and Lawrence, Neil D. Sparse convolved Gaussian processes for multi-output regression. pp. 57–64, 2009.
- Álvarez, Mauricio A. and Lawrence, Neil D. Computationally efficient convolved multiple output Gaussian processes. *Journal of Machine Learning Research*, (12):1459–1500, jul 2011.
- Álvarez, Mauricio A., Luengo, David, and Lawrence, Neil D. Latent force models. *Artificial Intelligence and Statistics*, 5:9–16, 2009.
- Álvarez, Mauricio A., Luengo, David, Titsias, Michalis K., and Lawrence, Neil D. Efficient multioutput Gaussian processes through variational inducing kernels. 9:25–32, 2010.
- Bonilla, Edwin V., Chai, Kian Ming, and Williams, Christopher K. I. Multi-task Gaussian process prediction. *Advances in Neural Information Processing Systems*, 20:153–160, 2008.
- Bruinsma, W. P. The generalised Gaussian convolution process model. 2016.
- Cunningham, John P, Shenoy, Krishna V, and Sahani, Maneesh. Fast gaussian process methods for point process intensity estimation. In *International Conference on Machine learning*, pp. 192–199, 2008.
- Damianou, Andreas. *Deep Gaussian Processes and Variational Propagation of Uncertainty*. PhD thesis, Department of Neuroscience, University of Sheffield, 2014.
- Duvenaud, David, Lloyd, James Robert, Grosse, Roger, Tenenbaum, Joshua B., and Ghahramani, Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, June 2013.
- Friedman, Nir and Nachman, Iftach. Gaussian process networks. In *Uncertainty in Artificial Intelligence*, pp. 211–219. Morgan Kaufmann Publishers Inc., 2000.
- Goovaerts, Pierre. *Geostatistics for Natural Resources Evaluation*. Oxford University Press, 1 edition, 1997.
- Hensman, James, Fusi, Nicolo, and Lawrence, Neil D. Gaussian processes for big data. *ArXiv E-Prints*, 2013.
- Hernández-Lobato, José Miguel. Neural networks with optimal accuracy and speed in their predictions, 2016.
- Kennedy, Marc C and O’Hagan, Anthony. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1): 1–13, 2000.
- Koller, Daphne and Friedman, Nir. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.
- Larochelle, Hugo and Murray, Iain. The neural autoregressive distribution estimator. In *AISTATS*, volume 1, pp. 2, 2011.
- Le Gratiet, Loic and Garnier, Josselin. Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. *International Journal for Uncertainty Quantification*, 4(5), 2014.
- Lloyd, James Robert, Duvenaud, David, Grosse, Roger, Tenenbaum, Joshua B., and Ghahramani, Zoubin. Automatic construction and natural-language description of nonparametric regression models. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2014.
- Nguyen, Trung V. and Bonilla, Edwin V. Collaborative multi-output Gaussian processes. *Conference on Uncertainty in Artificial Intelligence*, 30, 2014.
- Perdikaris, Paris, Raissi, Maziar, Damianou, Andreas, Lawrence, ND, and Karniadakis, George Em. Non-linear information fusion algorithms for data-efficient multi-fidelity modelling. In *Proc. R. Soc. A*, volume 473, pp. 20160751. The Royal Society, 2017.
- Rasmussen, Carl Edward and Williams, Christopher K. I. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- Saatçi, Yunus. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, 2012.
- Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 2951–2959, 2012.
- Stein, Michael. *Interpolation of Spatial Data*. Springer-Verlag New York, 1 edition, 1999.
- Teh, Yee Whye and Seeger, Matthias. Semiparametric latent factor models. *International Workshop on Artificial Intelligence and Statistics*, 10, 2005.

- Titsias, Michalis K. Variational learning of inducing variables in sparse Gaussian processes. *Artificial Intelligence and Statistics*, 12:567–574, 2009.
- Wackernagel, Hans. *Multivariate Geostatistics*. Springer-Verlag Berlin Heidelberg, 3 edition, 2003.
- Wilson, Andrew G., Knowles, David A., and Ghahramani, Zoubin. Gaussian process regression networks. *International Conference on Machine Learning*, 29, june 2012.
- Wilson, Andrew G, Hu, Zhiting, Salakhutdinov, Ruslan R, and Xing, Eric P. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pp. 2586–2594, 2016.
- Wilson, Andrew Gordon. *Covariance Kernels for Fast Automatic Pattern Discovery and Extrapolation With Gaussian Processes*. PhD thesis, University of Cambridge, 2014.
- Yuan, Chao. Conditional multi-output regression. In *International Joint Conference on Neural Networks*, pp. 189–196. IEEE, 2011.
- Zhang, X.L., Begleiter, H., Porjesz, B., Wang, W., and Litke, A. Event related potentials during object recognition tasks. *Brain Research Bulletin*, 38(6): 531–538, 1995.

## A Conditional Independence in Figure 2b

Before considering Figure 2b, we review some basic notions concerning graphical models.

Let a path be a sequence of nodes  $v_1, \dots, v_n$  from some directed graph  $G$  where, for each  $v_i$ ,  $G$  either contains an edge from  $v_i$  to  $v_{i+1}$ , or an edge from  $v_{i+1}$  to  $v_i$ . If  $G$  contains an edge from  $a$  to  $b$ , then write  $a \rightarrow b$  to mean the two-node path in which node  $b$  follows node  $a$ . Similarly, if  $G$  contains an edge from  $b$  to  $a$ , then write  $a \leftarrow b$  to mean the two-node path in which  $b$  follows  $a$ . Write  $a \rightleftharpoons b$  to mean either  $a \rightarrow b$  or  $b \leftarrow a$ .

**Definition 1** (Active Path (Definition 3.6 from (Koller & Friedman, 2009))). Let  $\mathcal{P} = v_1 \rightleftharpoons \dots \rightleftharpoons v_n$  be a path in a graphical model. Let  $Z$  be a subset of the variables from the graphical model. Then, call  $\mathcal{P}$  active given  $Z$  if

- for every v-structure  $v_{i-1} \rightarrow v_i \leftarrow v_{i+1}$  in  $\mathcal{P}$ ,  $v_i$  or a descendant of  $v_i$  is in  $Z$ ; and
- no other node in  $\mathcal{P}$  is in  $Z$ .

**Definition 2** (d-Separation (Definition 3.7 from (Koller & Friedman, 2009))). Let  $X$ ,  $Y$ , and  $Z$  be three sets of nodes from a graphical model. Then, call  $X$  and  $Y$  d-separated given  $Z$  if no path between any  $x \in X$  and  $y \in Y$  is active given  $Z$ .

**Theorem 2** (d-Separation Implies Conditional Independence (Theorem 3.3 from (Koller & Friedman, 2009))). Let  $X$ ,  $Y$ , and  $Z$  be three sets of nodes from a graphical model. If  $X$  and  $Y$  are d-separated given  $Z$ , then  $X \perp Y \mid Z$ .

Consider Figure 11. Define the layers of a node in our graphical model to be

$$\text{layer}(f_i) = \text{layer}(g_i(\mathbf{x})) = \text{layer}(y_i(\mathbf{x})) = i.$$

*Proof of Lemma 1.* Since  $A_i(\mathbf{x}, \mathbf{y})$  depends only on  $(\mathbf{x}, \mathbf{y}_{1:i-1})$ , any sample from  $\mathbf{g} \mid \mathbf{A}, \mathbf{u}$  satisfies  $g_i = u_i + A_i \circ \mathbf{g}$ , by Proposition 1, so  $g_i = u_i + A_i \circ (\mathbf{g}_{1:i-1}, \mathbf{0})$ , where  $(\mathbf{g}_{1:i-1}, \mathbf{0})$  represents the concatenation of  $\mathbf{g}_{1:i-1}$  and  $M - i + 1$  zeros. The equivalence now follows.  $\square$

*Proof.* Let  $\mathcal{P}$  be a path between any  $y_i(\mathbf{x}') \in y_i$  and  $y_j(\mathbf{x})$ :

$$\mathcal{P} = \underbrace{v_1}_{y_i(\mathbf{x}')} \rightleftharpoons v_2 \rightleftharpoons \dots \rightleftharpoons v_{n-1} \rightleftharpoons \underbrace{v_n}_{y_j(\mathbf{x})}.$$

$\mathcal{P}$  defines a sequences of layers  $L = \text{layer}(v_1), \dots, \text{layer}(v_n)$ . Let  $k$  be the maximum

of this sequence. Since  $\text{layer}(v_n) = j$ , it holds that  $k \geq j > i$ ; in other words, at some point,  $\mathcal{P}$  transitions from a layer strictly lower than  $k$  to layer  $k$ : there must exist an  $m < k$  and  $\hat{\mathbf{x}}$  such that

$$\dots \rightarrow y_m(\hat{\mathbf{x}}) \rightarrow g_k(\hat{\mathbf{x}}) \rightleftharpoons \dots \quad (6)$$

is part of  $\mathcal{P}$ .

If  $y_k(\hat{\mathbf{x}}) \in \mathcal{D}$ , then, since  $\mathcal{D}$  is closed downwards,  $y_m(\hat{\mathbf{x}}) \in \mathcal{D}$ , meaning that  $\mathcal{P}$  is inactive; note that Equation (6) shows that  $y_m(\hat{\mathbf{x}})$  is not in a v-structure.

If, on the other hand,  $y_k(\hat{\mathbf{x}}) \notin \mathcal{D}$ , then, since  $\mathcal{D}$  is closed downwards, neither  $g_k(\hat{\mathbf{x}})$  nor a descendant of  $g_k(\hat{\mathbf{x}})$  is in  $\mathcal{D}$ . If  $y_k(\hat{\mathbf{x}})$  would follow  $g_k(\hat{\mathbf{x}})$  in  $\mathcal{P}$ , then, looking at Figure 11, any node following  $y_k(\hat{\mathbf{x}})$  would be in a layer strictly higher than  $k$ , which would be a contradiction:  $k$  is chosen to be the highest layer in  $\mathcal{P}$ . Therefore,  $y_k(\hat{\mathbf{x}})$  cannot follow  $g_k(\hat{\mathbf{x}})$  in  $\mathcal{P}$ , meaning that, again looking at Figure 11,  $g_k(\hat{\mathbf{x}})$  must be in a v-structure. Hence,  $\mathcal{P}$  is inactive, because we previously concluded that neither  $g_k(\hat{\mathbf{x}})$  nor a descendant of  $g_k(\hat{\mathbf{x}})$  is in  $\mathcal{D}$ .  $\square$

## B Proofs of Lemmas 1 and 2

For functions  $\mathbf{A}, \mathbf{B}: \mathcal{X} \times (\mathcal{Y}^{\mathcal{X}})^M \rightarrow \mathcal{Y}^M$ , define composition  $\circ$  as follows:  $(\mathbf{A} \circ \mathbf{B})(\mathbf{x}, \mathbf{y}) = \mathbf{A}(\mathbf{x}, \mathbf{B}(\cdot, \mathbf{y}))$ . Note that  $\circ$  is well-defined and associative. For a function  $\mathbf{u}: \mathcal{X} \rightarrow \mathcal{Y}^M$ , denote  $\mathbf{A} \circ \mathbf{u}: \mathcal{X} \rightarrow \mathcal{Y}^M$ ,  $\mathbf{A} \circ \mathbf{u} = \mathbf{A}(\cdot, \mathbf{u})$ . Again, note that  $(\mathbf{A} \circ \mathbf{B}) \circ \mathbf{u} = \mathbf{A} \circ (\mathbf{B} \circ \mathbf{u})$ . Furthermore, denote

$$\underbrace{\mathbf{A} \circ \dots \circ \mathbf{A}}_{n \text{ times}} = \mathbf{A}^n.$$

Consider a function  $\mathbf{A}: \mathcal{X} \times (\mathcal{Y}^{\mathcal{X}})^M \rightarrow \mathcal{Y}^M$  such that  $A_i(\mathbf{x}, \mathbf{y}): \mathcal{X} \times (\mathcal{Y}^{\mathcal{X}})^M \rightarrow \mathcal{Y}$  depends only on  $(\mathbf{x}, \mathbf{y}_{1:i-1})$ , where  $A_1 = 0$ . Further let  $\mathbf{u}, \mathbf{g}: \mathcal{X} \rightarrow \mathcal{Y}^M$ , denote  $\mathbf{T}\mathbf{f} = \mathbf{u} + \mathbf{A} \circ \mathbf{f}$ , and denote  $N$  consecutive applications of  $\mathbf{T}$  by  $\mathbf{T}^N$ .

To proof Lemmas 1 and 2, we first show that  $\mathbf{T}^{M-1}\mathbf{u}$  is the unique solution of a functional equation which is much easier to analyse.

**Proposition 1.** The unique solution of  $\mathbf{g} = \mathbf{u} + \mathbf{A} \circ \mathbf{g}$  is  $\mathbf{g} = \mathbf{T}^{M-1}\mathbf{u}$ .

*Proof of Proposition 1.* First, we show that  $\mathbf{g} = \mathbf{u} + \mathbf{A} \circ \mathbf{g}$  has a solution, and that this solution is unique. Because  $A_i(\mathbf{x}, \mathbf{y})$  depends only on  $(\mathbf{x}, \mathbf{y}_{1:i-1})$ , it holds that

$$g_i = u_i + A_i \circ \mathbf{g} = u_i + A_i \circ (\mathbf{g}_{1:i-1}, \mathbf{0}),$$

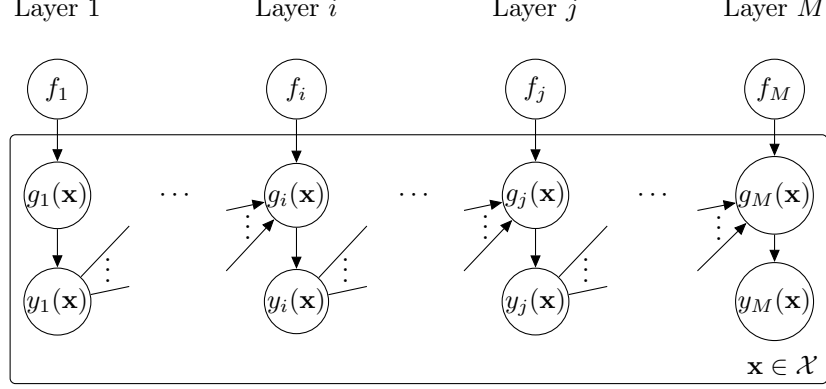


Figure 11: A more general representation of the graphical model corresponding to GPAR

where  $(\mathbf{g}_{1:i-1}, \mathbf{0})$  represents the concatenation of  $\mathbf{g}_{1:i-1}$  and  $M - i + 1$  zeros. Thus,  $g_i$  can uniquely be constructed from  $u_i$ ,  $A_i$ , and  $\mathbf{g}_{1:i-1}$ ; therefore,  $g_1$  exists and is unique, so  $g_2$  exists and is unique: by induction we find that  $\mathbf{g}$  exists and is unique.

Second, we show that  $\mathbf{g} = \mathbf{T}^{M-1} \mathbf{u}$  satisfies  $\mathbf{g} = \mathbf{u} + \mathbf{A} \circ \mathbf{g} = \mathbf{T} \mathbf{g}$ . To show this, we show that  $(\mathbf{T}^n \mathbf{u})_i = (\mathbf{T}^{n-1} \mathbf{u})_i$  for  $i = 1, \dots, n$ , for all  $n$ . To begin with, we show the base case,  $n = 1$ :

$$(\mathbf{T} \mathbf{u})_1 = u_1 + A_1 \circ \mathbf{u} = u_1 = (\mathbf{T}^0 \mathbf{u})_1,$$

since  $A_1 = 0$ . Finally, suppose that the claim holds for a particular  $n$ . We show that the claim then holds for  $n + 1$ : Let  $i \leq n + 1$ . Then

$$\begin{aligned} (\mathbf{T}^{n+1} \mathbf{u})_i &= u_i + A_i \circ \mathbf{T}^n \mathbf{u} \\ &= u_i + A_i \circ ((\mathbf{T}^n \mathbf{u})_{1:i-1}, (\mathbf{T}^n \mathbf{u})_{i:M}) \\ &= u_i + A_i \circ ((\mathbf{T}^{n-1} \mathbf{u})_{1:i-1}, (\mathbf{T}^n \mathbf{u})_{i:M}) \\ &\quad \text{(By assumption)} \\ &\stackrel{*}{=} u_i + A_i \circ ((\mathbf{T}^{n-1} \mathbf{u})_{1:i-1}, (\mathbf{T}^{n-1} \mathbf{u})_{i:M}) \\ &= u_i + A_i \circ \mathbf{T}^{n-1} \mathbf{u} \\ &= (\mathbf{T}^n \mathbf{u})_i, \end{aligned}$$

where  $*$  holds because  $A_i(\mathbf{x}, \mathbf{y})$  depends only on  $(\mathbf{x}, \mathbf{y}_{1:i-1})$ .  $\square$

In the linear case,  $\mathbf{T}^{M-1} \mathbf{u}$  turns out to greatly simplify.

**Proposition 2.** If  $\mathbf{A}(\mathbf{x}, \mathbf{y})$  is linear in  $\mathbf{y}$ , then  $\mathbf{T}^{M-1} \mathbf{u} = (\sum_{i=0}^{M-1} \mathbf{A}^i) \circ \mathbf{u}$ .

*Proof of Proposition 2.* If  $\mathbf{A}(\mathbf{x}, \mathbf{y})$  is linear in  $\mathbf{y}$ , then

one verifies that  $\circ$  distributes over addition. Therefore,

$$\begin{aligned} \mathbf{T}^{M-1} \mathbf{u} &= \mathbf{u} + \mathbf{A} \circ \mathbf{T}^{M-2} \mathbf{u} \\ &= \mathbf{u} + \mathbf{A} \circ \mathbf{u} + \mathbf{A}^2 \circ \mathbf{T}^{M-3} \mathbf{u} \\ &\quad \vdots \\ &= \mathbf{u} + \mathbf{A} \circ \mathbf{u} + \dots + \mathbf{A}^{M-1} \circ \mathbf{u}. \end{aligned}$$

$\square$

We can now use Propositions 1 and 2 to prove Lemmas 1 and 2.

*Proof of Lemma 1.* Since  $A_i(\mathbf{x}, \mathbf{y})$  depends only on  $(\mathbf{x}, \mathbf{y}_{1:i-1})$ , it holds by Proposition 1 that any sample from  $\mathbf{g} | \mathbf{A}, \mathbf{u}$  satisfies  $g_i = u_i + A_i \circ \mathbf{g}$ , so  $g_i = u_i + A_i \circ (\mathbf{g}_{1:i-1}, \mathbf{0})$ , where  $(\mathbf{g}_{1:i-1}, \mathbf{0})$  represents the concatenation of  $\mathbf{g}_{1:i-1}$  and  $M - i + 1$  zeros. The equivalence now follows.  $\square$

*Proof of Lemma 2.* First, one verifies that  $A_i(\mathbf{x}, \mathbf{y})$  still depends only on  $(\mathbf{x}, \mathbf{y}_{1:i-1})$ , and that  $A_i(\mathbf{x}, \mathbf{y})$  is linear in  $\mathbf{y}$ . The result then follows from Lemma 1 and Proposition 2, where the expression for  $k_{A_i}$  follows from straightforward calculation.  $\square$

### C Lemma 3

Call functions  $k_1, \dots, k_M: \mathcal{X} \rightarrow \mathbb{R}$  linearly independent if

$$\left( \forall \mathbf{x} : \sum_{i=1}^M c_i k_i(\mathbf{x}) = 0 \right) \implies c_1 = \dots = c_M = 0.$$

**Lemma 3.** Let  $k_1, \dots, k_M: \mathcal{X} \rightarrow \mathbb{R}$  be linearly independent and arrange them in a diagonal matrix  $\mathbf{K} = \text{diag}(k_1, \dots, k_n)$ . Let  $\mathbf{A}$  be an invertible  $M \times M$  matrix such that its columns cannot be permuted into a

triangular matrix. Then there does not exist an invertible triangular matrix  $\mathbf{T}$  such that  $\mathbf{T}^{-1}\mathbf{BK}(\mathbf{x})\mathbf{B}^T\mathbf{T}^{-T}$  is diagonal for all  $\mathbf{x}$ .

*Proof.* Suppose, on the contrary, that such  $\mathbf{T}$  does exist. Then two different rows  $\mathbf{a}_p$  and  $\mathbf{a}_q$  of  $\mathbf{A} = \mathbf{T}^{-1}\mathbf{B}$  share nonzero elements in some columns  $C$ ; otherwise,  $\mathbf{A}$  would have exactly one nonzero entry in every column— $\mathbf{A}$  is invertible—so  $\mathbf{A}$  would be the product of a permutation matrix and a diagonal matrix, meaning that  $\mathbf{B} = \mathbf{TA}$ 's columns could be permuted into a triangular matrix. Now, by  $\mathbf{T}^{-1}\mathbf{BK}(\mathbf{x})\mathbf{B}^T\mathbf{T}^{-T} = \mathbf{AK}(\mathbf{x})\mathbf{A}^T$  being diagonal for all  $\mathbf{x} \in \mathcal{X}$ ,  $\sum_i \mathbf{a}_{p,i}\mathbf{a}_{q,i}k_i(\mathbf{x}) = 0$  for all  $\mathbf{x}$ . Therefore, by linear independence of  $k_1, \dots, k_N$ , it holds that  $\mathbf{a}_{p,i}\mathbf{a}_{q,i} = 0$  for all  $i$ . But  $\mathbf{a}_{p,i}\mathbf{a}_{q,i} \neq 0$  for any  $i \in C$ , which is a contradiction.  $\square$