Emulating computer models with step-discontinuous outputs using Gaussian processes

Hossein Mohammadi [*1, 2], Peter Challenor[1, 2], Marc Goodfellow[1, 2], and Daniel Williamson[1]

[1]College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, UK
[2]EPSRC Centre for Predictive Modelling in Healthcare, University of Exeter, Exeter, UK

Abstract

In many real-world applications, we are interested in approximating functions that are analytically unknown. An emulator provides a "fast" approximation of such functions relying on a limited number of evaluations. Gaussian processes (GPs) are commonplace emulators due to their properties such as the ability to quantify uncertainty. GPs are essentially developed to emulate smooth, continuous functions. However, the assumptions of continuity and smoothness is unwarranted in many situations. For example, in computer models where *bifurcation*, *tipping points* occur in their systems of equations, the outputs can be discontinuous.

This paper examines the capacity of GPs for emulating step-discontinuous functions using two approaches. The first approach is based on choosing covariance functions/kernels, namely neural network and Gibbs, that are most appropriate for modelling discontinuities. The predictive performance of these two kernels is illustrated using several examples. The results show that they have superior performance to standard covariance functions, such as the Matérn family, in capturing sharp jumps. The second approach is to transform the input space such that in the new space a GP with a standard kernel is able to predict the function well. A parametric transformation function is used whose parameters are estimated by maximum likelihood.

Keywords: Covariance function, Discontinuity, Emulator, Gaussian processes, Warping.

1   Introduction

Computer models (or simulators) are widely used in many applications ranging from modelling the ocean and atmosphere [1, 6] to healthcare [2, 7]. By simulating real-

---

*Corresponding Author: h.mohammadi@exeter.ac.uk

world phenomena, computer models allow us to better understand/analyse them as a complement to conducting physical experiments. However, on the one hand, the simulators are "black box" since they are often available as commercial packages, and on the other hand, they are computationally expensive. The latter is due to the fact that each simulation outcome is actually the solution of some complex mathematical equations, such as partial differential equations.

One of the main purposes of using a computer model is to perform prediction. However, such prediction is uncertain because simulators are simplifications of physical phenomena. In addition, due to factors such as lack of knowledge or measurement error, the inputs to the model are subject to uncertainty which yield uncertain outputs. Under this condition, decision makers need to know how good the prediction is. In other words, they need an estimation of the uncertainty propagated through the model [23]. This entails running the simulator very many times which is impractical in the context of time-consuming simulators. To overcome this, one can replace the simulator with an *emulator*.

Emulation is a statistical approach for representing unknown functions by approximating the input/output relationship based on evaluations at a finite set of points. Gaussian process (GP) models (also known as kriging) are widely used to predict the outputs of a simulator and are regarded as an important class of emulators [24]. GPs are nonparametric probabilistic models that provide not only a mean predictor but also a quantification of the associated uncertainty. They have become a standard tool for the design and analysis of computer experiments over the last two decades. This includes uncertainty propagation [22, 18], calibration [17, 14], design of experiments [28, 26], optimisation [16, 3] and sensitivity analysis [23, 15].

GPs can be applied to fit any smooth, continuous function [20]. The basic assumption when using a GP emulator is that the unknown function depends smoothly on its input parameters. However, there are many situations where the model outputs are not continuous. It is very common in computer models that at some regions of the input space, a minor change in the input parameters leads to a sharp jump in the output. For example, models described by nonlinear differential equations often exhibit different modes (phases). Shifting from one mode to another relies on a different set of equations which raises a discontinuity in the model output.

To our knowledge, there are only a few studies that investigate the applicability of GPs in modelling discontinuities. The reason may be due to the fact that they are essentially developed to model smooth and continuous surface forms. However, a natural way of emulating discontinuous functions is to partition the input space by finding discontinuities and then fit separate GP models within each partition. In [4], for example, a simulator with tipping point behaviour is emulated such that the boundary of the regions with discontinuity is found first and the simulator output is emulated separately in each region. It is reported that finding the discontinuous regions is a time-consuming operation.

The treed Gaussian process (TGP) [11] is a popular model introduced by Gramacy and Lee. The TGP makes binary splits (parallel to the input axes) on the value of a single variable recursively, i.e., each partition (leaf of the tree) is a subregion of the previous section. Then, an independent stationary GP emulator is applied within each section. The disadvantage of the TGP is that it requires many simulation runs which

2

is not affordable in the context of computationally expensive simulators. A similar approach is presented in [25] where Voronoi tessellation is applied to partition the input space. The procedure uses the reversible jump Markov chain Monte Carlo [12] that is time-consuming.

Here we provide an alternative perspective in which a single kernel is used to capture discontinuities. This includes two nonstationary covariance functions, namely neural network and Gibbs, and the idea of transforming the input space. It is shown how these techniques coming from machine learning can be applied to model discontinuous functions in the field of computer experiments. The advantage is that there is no need to detect discontinuous boundaries separately which is burdensome.

2 Overview of Gaussian process emulators

The random (or stochastic) process $Z = (Z(\mathbf{x}))_{\mathbf{x} \in \mathcal{D}}$, i.e. a collection of random variables indexed by the set $\mathcal{D}$, is a Gaussian process if and only if $\forall\ N \in \mathbb{N}$, $\forall\ \mathbf{x}^i \in \mathcal{D}$, $\left(Z(\mathbf{x}^1), \ldots, Z(\mathbf{x}^N) \right)^\top$ has a multivariate normal distribution on $\mathbb{R}^N$ [27]. Suppose $(\Omega, \mathcal{B}, \mathbb{P})$ is the probability space on which $Z(\mathbf{x})$ is defined:

$$Z : (\mathbf{x}, \omega) \in \mathcal{D} \times (\Omega, \mathcal{B}, \mathbb{P}) \longmapsto Z(\mathbf{x}).$$

For a given $\omega_o \in \Omega$, $Z(\mathbf{x}, \omega_o)$ is called a *realisation* (or *sample path*) and for a given $\mathbf{x}_o \in \mathcal{D}$, $Z(\mathbf{x}_o, \omega)$ is a Gaussian random variable. In this framework, GPs can be regarded as the probability distribution over functions such that the function being approximated is considered as a particular realisation of the distribution. Herein, $f : \mathcal{D} \longmapsto \mathcal{F}$ denotes the unknown function that maps the input space $\mathcal{D} \subset \mathbb{R}^d$ to the output space $\mathcal{F}$. In this work, $\mathcal{F} = \mathbb{R}$.

A GP is fully determined by its mean function $\mu(.)$ and covariance kernel $k(.,.)$ which are defined as:

$$Z \sim \mathcal{GP}\left(\mu(.), k(.,.)\right) ; \quad \begin{array}{l} \mu : \mathcal{D} \longmapsto \mathbb{R}\ ,\ \mu(\mathbf{x}) = \mathbb{E}\left[Z(\mathbf{x})\right] \\ k : \mathcal{D} \times \mathcal{D} \longmapsto \mathbb{R}\ ,\ k(\mathbf{x}, \mathbf{x}') = \mathbb{C}\mathrm{ov}\left(Z(\mathbf{x}), Z(\mathbf{x}')\right). \end{array}$$

While $\mu(.)$ could be any function, $k(.,.)$ needs to be symmetric positive definite. The function $\mu(.)$ captures the global trend and $k(.,.)$ controls the structure of sample paths such as differentiability, symmetry, periodicity, etc. In this work, an unknown constant function is used as the GP mean ($\mu(\mathbf{x}) = \mu$) which is estimated from data, see Equation (5).

Generally, covariance functions are divided into two groups: *stationary* and *nonstationary*. Stationary kernels depend only on the separation vector $\mathbf{x} - \mathbf{x}'$. As a result, they are translation invariant in the input space:

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} + \boldsymbol{\tau}, \mathbf{x}' + \boldsymbol{\tau})\,,\ \boldsymbol{\tau} \in \mathbb{R}^d. \tag{1}$$

Nonstationary kernels are applied to model functions that do not have uniform smoothness within the input space and change significantly in some regions compared to others [32]. In this paper, we are interested in kernels that are able to model discontinuity.

Sections 3 and 4 study two nonstationary covariance functions with this capability, namely neural network and Gibbs.

A list of the most common univariate stationary kernels is given in Table 1. They can be generalised to d-dimensional space using the tensor product of one-dimensional forms. For example, a d-dimensional squared exponential (SE) kernel is expressed by

$$k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma^2 \prod_{i=1}^{d} \exp\left(-\frac{|x_i - x_i'|^2}{2l_i^2}\right),$$

where the parameters σ^2 and l_i are called *process variance* and correlation *length-scale* along the i-th coordinate, respectively. The former determines variation of sample paths along y-axis and the latter controls how quickly they vary along x-axis. In this paper, these parameters are estimated via maximum likelihood estimation (MLE) [27, 16], see Appendix A. Figure 1 shows the shape of the SE kernel and two sample paths with different length-scales.

Covariance function	Formula
Exponential	$\sigma^2 \exp\left(-\frac{|x-x'|}{l}\right)$
Matérn 3/2	$\sigma^2 \left(1 + \frac{\sqrt{3}|x-x'|}{l}\right) \exp\left(-\frac{\sqrt{3}|x-x'|}{l}\right)$
Matérn 5/2	$\sigma^2 \left(1 + \frac{\sqrt{5}|x-x'|}{\theta} + \frac{5(x-x')^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}|x-x'|}{l}\right)$
Squared exponential (SE)	$\sigma^2 \exp\left(-\frac{|x-x'|^2}{2l^2}\right)$

Table 1: The most common univariate stationary covariance functions.

The GP prediction of f is obtained by conditioning Z on function evaluations. Let $\mathbf{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ denote n sample locations in the input space and $\mathbf{y} = \left(f(\mathbf{x}^1), \ldots, f(\mathbf{x}^n)\right)^\top$ represent the corresponding outputs (observations). Together, the set $\mathcal{A} = \{\mathbf{X}, \mathbf{y}\}$ is called the *training* set. The conditional distribution of Z on $\mathcal{A}$ is again a GP

$$Z|\mathcal{A} \sim \mathcal{GP}\left(m(.), c(.,.)\right), \tag{2}$$

specified by

$$m(\mathbf{x}) = \hat{\mu} + \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1}\left(\mathbf{y} - \hat{\mu}\mathbf{1}\right) \tag{3}$$

$$c(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1}\mathbf{k}(\mathbf{x}')$$
$$+ \left(1 - \mathbf{1}^\top \mathbf{K}^{-1}\mathbf{k}(\mathbf{x})\right)^\top \left(\mathbf{1}^\top \mathbf{K}^{-1}\mathbf{1}\right)^{-1} \left(1 - \mathbf{1}^\top \mathbf{K}^{-1}\mathbf{k}(\mathbf{x}')\right). \tag{4}$$

Here, $\hat{\mu}$ is the best linear unbiased estimate of μ obtained by

$$\hat{\mu} = \frac{\mathbf{1}^\top \mathbf{K}^{-1}\mathbf{y}}{\mathbf{1}^\top \mathbf{K}^{-1}\mathbf{1}}. \tag{5}$$
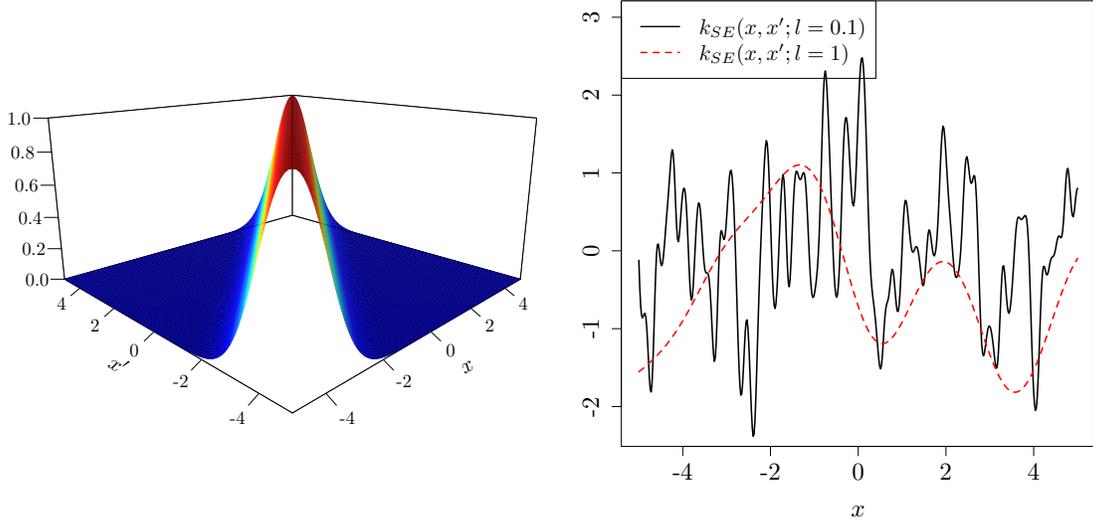
Figure 1: Left: shape of the squared exponential kernel. Right: sample paths corresponding to the SE kernel with $l = 0.1$ (solid) and 1 (red dased). In both cases $\sigma^2 = 1$

Also, $\mathbf{k}(\mathbf{x}) = \big(k(\mathbf{x}, \mathbf{x}^1), \ldots, k(\mathbf{x}, \mathbf{x}^n)\big)^\top$, $\mathbf{K}$ is an $n \times n$ covariance matrix whose elements are: $\mathbf{K}_{ij} = k(\mathbf{x}^i, \mathbf{x}^j)$, $\forall i, j$; $1 \leq i, j \leq n$ and $\mathbf{1}$ is a $n \times 1$ vector of ones. We call $m(\mathbf{x})$ and $s^2(\mathbf{x}) = c(\mathbf{x}, \mathbf{x})$ the GP mean and variance which reflect the prediction and the associated uncertainty at $\mathbf{x}$, respectively.

If the covariance function $k(.,.)$ is stationary, the predictive mean expressed by Equation (3) interpolates the points in the training set. Also, the prediction uncertainty (Equation (4)) vanishes there. To clarify, we obtain the prediction and the associated uncertainty at $\mathbf{x} = \mathbf{x}^j$, the j-th training point. In this case, $\mathbf{k}(\mathbf{x}^j)$ is equivalent to the jth column of the covariance matrix $\mathbf{K}$. Because $\mathbf{K}$ is a positive definite matrix, the term $\mathbf{k}(\mathbf{x}^j)^\top \mathbf{K}^{-1}$ yields vector $\mathbf{e}_j = (0, \ldots, 0, 1, 0, \ldots, 0)$ whose elements are zero except the jth element which is one. As a result

$$m(\mathbf{x}^j) = \hat{\mu} + \overbrace{\mathbf{k}(\mathbf{x}^j)^\top \mathbf{K}^{-1}}^{\mathbf{e}_j}(\mathbf{y} - \hat{\mu}\mathbf{1}) = f(\mathbf{x}^j), \tag{6}$$

$$s^2(\mathbf{x}^j) = k(\mathbf{x}^j, \mathbf{x}^j) - \mathbf{k}(\mathbf{x}^j)^\top \mathbf{K}^{-1}\mathbf{k}(\mathbf{x}^j)$$
$$+ \frac{\big(1 - \mathbf{1}^\top \mathbf{K}^{-1}\mathbf{k}(\mathbf{x}^j)\big)^2}{\mathbf{1}^\top \mathbf{K}^{-1}\mathbf{1}} = 0, \tag{7}$$

since $\mathbf{k}(\mathbf{x}^j) = (k(\mathbf{x}^j, \mathbf{x}^1), \ldots, \overbrace{k(\mathbf{x}^j, \mathbf{x}^j)}^{\sigma^2}, \ldots, k(\mathbf{x}^j, \mathbf{x}^n))^\top$.

3 Neural network kernel

A neural network (NN) kernel is derived from a one-hidden-layer neural network with infinite number of hidden units (neurons) [31]. Let $\tilde{f}(\mathbf{x})$ is a neural network with J units that maps inputs to outputs according to

$$\tilde{f}(\mathbf{x}) = b + \sum_{j=1}^{J} v_j h(\mathbf{x}; \mathbf{u}_j), \tag{8}$$

where b is the intercept, v_js are weights to the units, $h(.)$ represents the transfer (activation) function in which weight $\mathbf{u}_j$ is assigned to the input $\mathbf{x}$. Suppose b and every v_j have zero mean Gaussian distribution with variances σ_b^2 and σ_v^2/J, respectively. If $\mathbf{u}_j$s have independent and identical distribution, then the mean and covariance of $\tilde{f}(\mathbf{x})$ are ($\mathbf{w}$ represents all weights together)

$$\mathbb{E}_{\mathbf{w}}\big[\tilde{f}(\mathbf{x})\big] = 0\,, \tag{9}$$

$$\begin{aligned}
\mathbb{C}\mathrm{ov}\left(\tilde{f}(\mathbf{x}), \tilde{f}(\mathbf{x}')\right) &= \mathbb{E}_{\mathbf{w}}\big[(\tilde{f}(\mathbf{x}) - 0)(\tilde{f}(\mathbf{x}') - 0)\big] \\
&= \sigma_b^2 + \frac{1}{J}\sum_{j=1}^{J}\sigma_v^2 \mathbb{E}_{\mathbf{u}}\big[h(\mathbf{x}; \mathbf{u}_j)h(\mathbf{x}'; \mathbf{u}_j)\big] \\
&= \sigma_b^2 + \sigma_v^2 \mathbb{E}_{\mathbf{u}}\big[h(\mathbf{x}; \mathbf{u})h(\mathbf{x}'; \mathbf{u})\big]. \tag{10}
\end{aligned}$$

Since $\tilde{f}(\mathbf{x})$ is the sum of independent random variables, it tends towards a normal distribution as , $J \to \infty$ according to the central theorem. In this situation, any collection $\big\{\tilde{f}(\mathbf{x}^1), \ldots, \tilde{f}(\mathbf{x}^N) | \ \forall N \in \mathbb{N}\big\}$ has a joint normal distribution and $\tilde{f}(\mathbf{x})$ becomes a zero mean Gaussian process with a covariance function specified in Equation (10).

The NN kernel is a particular case of the covariance structure expressed by Equation (10) in which $h(\mathbf{x}; \mathbf{u}) = \mathrm{erf}\left(u_0 + \sum_{j=1}^{d} u_j x_j\right)$, $\mathrm{erf}(.)$ is the *error function*: $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x \exp(-t^2)dt$, and $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ [31]. This leads to

$$k_{NN}(\mathbf{x}, \mathbf{x}') = \frac{2\sigma^2}{\pi}\arcsin\left(\frac{2\tilde{\mathbf{x}}^\top \boldsymbol{\Sigma}\tilde{\mathbf{x}}'}{\sqrt{(1 + 2\tilde{\mathbf{x}}^\top \boldsymbol{\Sigma}\tilde{\mathbf{x}})(1 + 2\tilde{\mathbf{x}}'^\top \boldsymbol{\Sigma}\tilde{\mathbf{x}}')}}\right). \tag{11}$$

where $\tilde{\mathbf{x}} = (1, x_1, \ldots, x_d)^\top$ is the augmented input vector and $\boldsymbol{\Sigma}$ is a diagonal matrix with elements $\sigma_0^2, \sigma_1^2, \ldots, \sigma_d^2$, i.e., the variances of $u_0, u_1, \ldots, u_d$. The length-scale of the jth coordinate is of order $1/\sigma_j$; the larger σ_j, the sample functions vary more quickly in the jth coordinate [19, 27]. This is illustrated in Figure 2 where the shapes of the NN kernel for two different values of σ_1 and the corresponding sample paths are plotted. The NN kernel is nonstationary, see Figure 2 and also Equation (11) which does not depend on $\mathbf{x} - \mathbf{x}'$. It can take negative values contrary to classic covariance functions such as the SE kernel depicted in Figure 1. In this kernel due to the superposition of the function $\mathrm{erf}(u_0 + u_1 x)$, sample paths tend to constant values for large positive or negative x [27]. Also, each observation has a correlation less than one with itself:

$$\mathbb{C}\mathrm{orr}(Z(\mathbf{x}), Z(\mathbf{x})) = k_{NN}(\mathbf{x}, \mathbf{x}; \sigma^2 = 1) = \frac{2}{\pi}\arcsin\left(\frac{2\tilde{\mathbf{x}}^\top \boldsymbol{\Sigma}\tilde{\mathbf{x}}}{1 + 2\tilde{\mathbf{x}}^\top \boldsymbol{\Sigma}\tilde{\mathbf{x}}}\right) < 1, \tag{12}$$
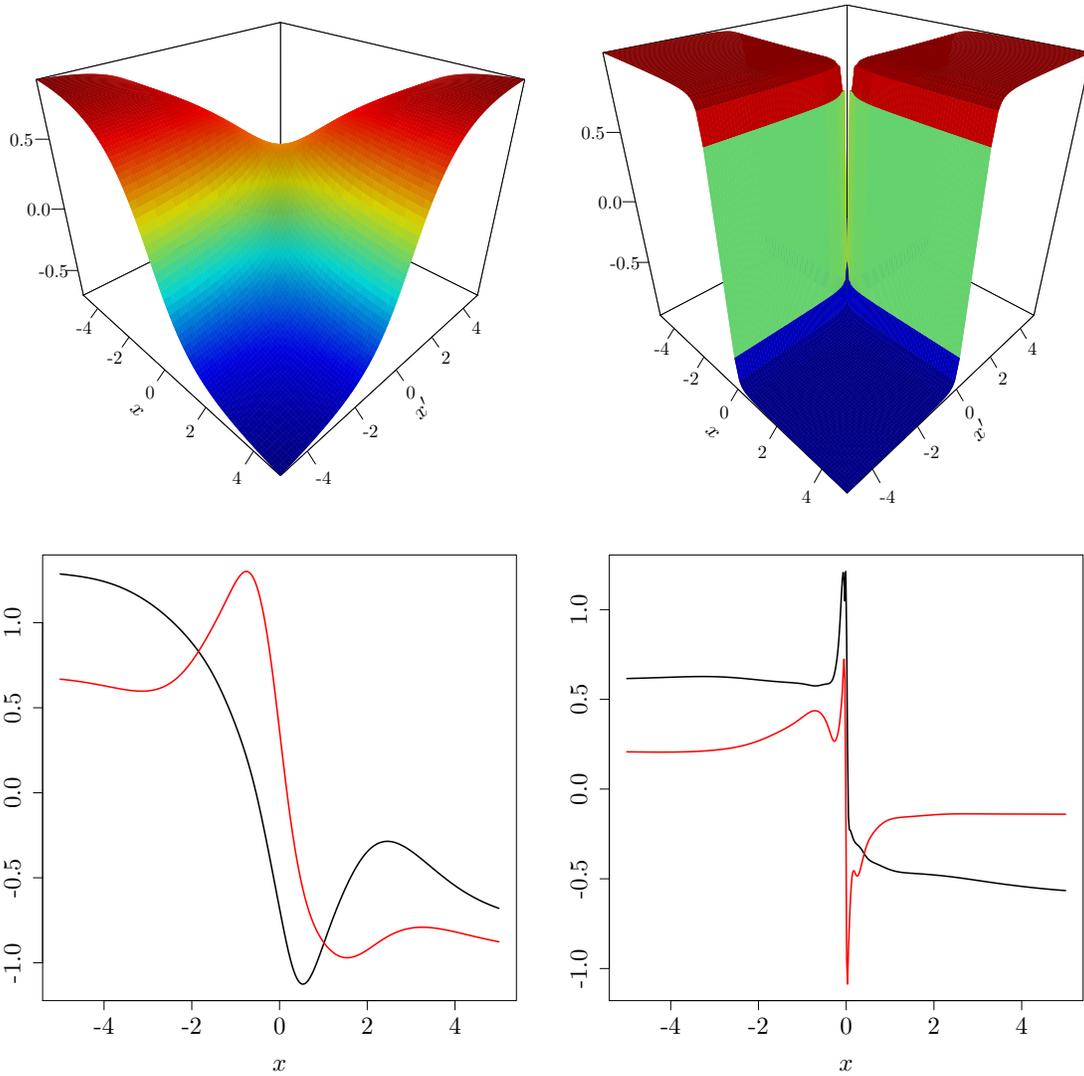
Figure 2: Top: The shapes of the NN kernel for two different values of σ_1: 1 (left) and 50 (right). Bottom: Two sample paths corresponding to the kernels on top. The kernel with $\sigma_1 = 50$ is a more suitable choice for modelling discontinuities. Here, $\sigma = \sigma_0 = 1$.

since $\arcsin\left(\frac{2\tilde{\mathbf{x}}^\top \boldsymbol{\Sigma}\tilde{\mathbf{x}}}{1+2\tilde{\mathbf{x}}^\top \boldsymbol{\Sigma}\tilde{\mathbf{x}}}\right) < \pi/2$. Thus, the mean predictor obtained by the NN kernel does not interpolate the points in the training data and the prediction variances are greater than zero there.

Figure 3 compares the Matérn 3/2 and NN kernels in modelling a step-function defined as

$$f(\mathbf{x}) = \begin{cases} -1 & x_1 \leq 0 \\ 1 & x_1 > 0 \ . \end{cases} \tag{13}$$

As can be seen, the NN kernel has superior performance to Matérn 3/2 in both 1D and 2D cases. The predictive mean of the GP with Matérn 3/2 neither captures the discontinuity nor performs well in the flat regions. In the NN kernel, the MLE estimation of the parameter that controls the horizontal scale of fluctuation, i.e. σ_1, takes its maximum possible value which is 10^3.

Figure 4 illustrates a function whose step-discontinuity is located at $x = 0.5$. As can be seen from the picture on the left of Figure 4, the NN kernel is not able to model f well. This problem can be solved if the NN kernel is modified as follows

$$k(x, x') = \frac{2\sigma^2}{\pi} \arcsin\left(\frac{2\tilde{\mathbf{x}}_\tau^\top \boldsymbol{\Sigma}^{-1}\tilde{\mathbf{x}}_\tau'}{\sqrt{(1 + 2\tilde{\mathbf{x}}_\tau^\top \boldsymbol{\Sigma}^{-1}\tilde{\mathbf{x}}_\tau)(1 + 2\tilde{\mathbf{x}}_\tau'^\top \boldsymbol{\Sigma}^{-1}\tilde{\mathbf{x}}_\tau')}}\right), \tag{14}$$

where $\tilde{\mathbf{x}}_\tau = (1, x - \tau)^\top$ and is estimated together with other parameters using MLE. In this case, $\hat{\tau} = 0.457$ which is an estimation for the location of the discontinuity.

4 Gibbs kernel

Mark Gibbs [10] in his PhD thesis derived the following covariance function:

$$k_{Gib}(\mathbf{x}, \mathbf{x}') = \sigma^2 \prod_{i=1}^d \left(\frac{2l_i(\mathbf{x})l_i(\mathbf{x}')}{l_i(\mathbf{x})^2 + l_i(\mathbf{x}')^2}\right)^{1/2} \exp\left(-\sum_{i=1}^d \frac{(x_i - x_i')^2}{l_i(\mathbf{x})^2 + l_i(\mathbf{x}')^2}\right), \tag{15}$$

where $l_i(.)$ is the length-scale in the ith input dimension. The length-scales could be any arbitrary positive functions of $\mathbf{x}$. This allows the kernel to model sudden variations in the observations: a process with Gibbs kernel is smooth at regions of the input space where the length-scales are relatively high and it changes rapidly where the length-scales reduce. Note that the correlation is one when $\mathbf{x} = \mathbf{x}'$, i.e. $k(\mathbf{x}, \mathbf{x}) = 1$. In this work, we use identical length-scale functions for all dimensions:

$$k_{Gib}(\mathbf{x}, \mathbf{x}') = \sigma^2 \left(\frac{2l(\mathbf{x})l(\mathbf{x}')}{l^2(\mathbf{x}) + l^2(\mathbf{x}')}\right)^{d/2} \exp\left(-\frac{\sum_{i=1}^d (x_i - x_i')^2}{l^2(\mathbf{x}) + l^2(\mathbf{x}')}\right). \tag{16}$$

Figure 5 shows the shapes of the Gibbs kernel for three different length-scale functions and corresponding sample paths. As can be seen, it is possible to model both nonstationary and discontinuous functional forms with the Gibbs kernel if a suitable length-scale function is chosen. For example, the nonstationary function depicted in Figure 6 varies more quickly in the region $x \in [0, 0.3]$ than in the region $[0.3, 1]$. Thus,

8
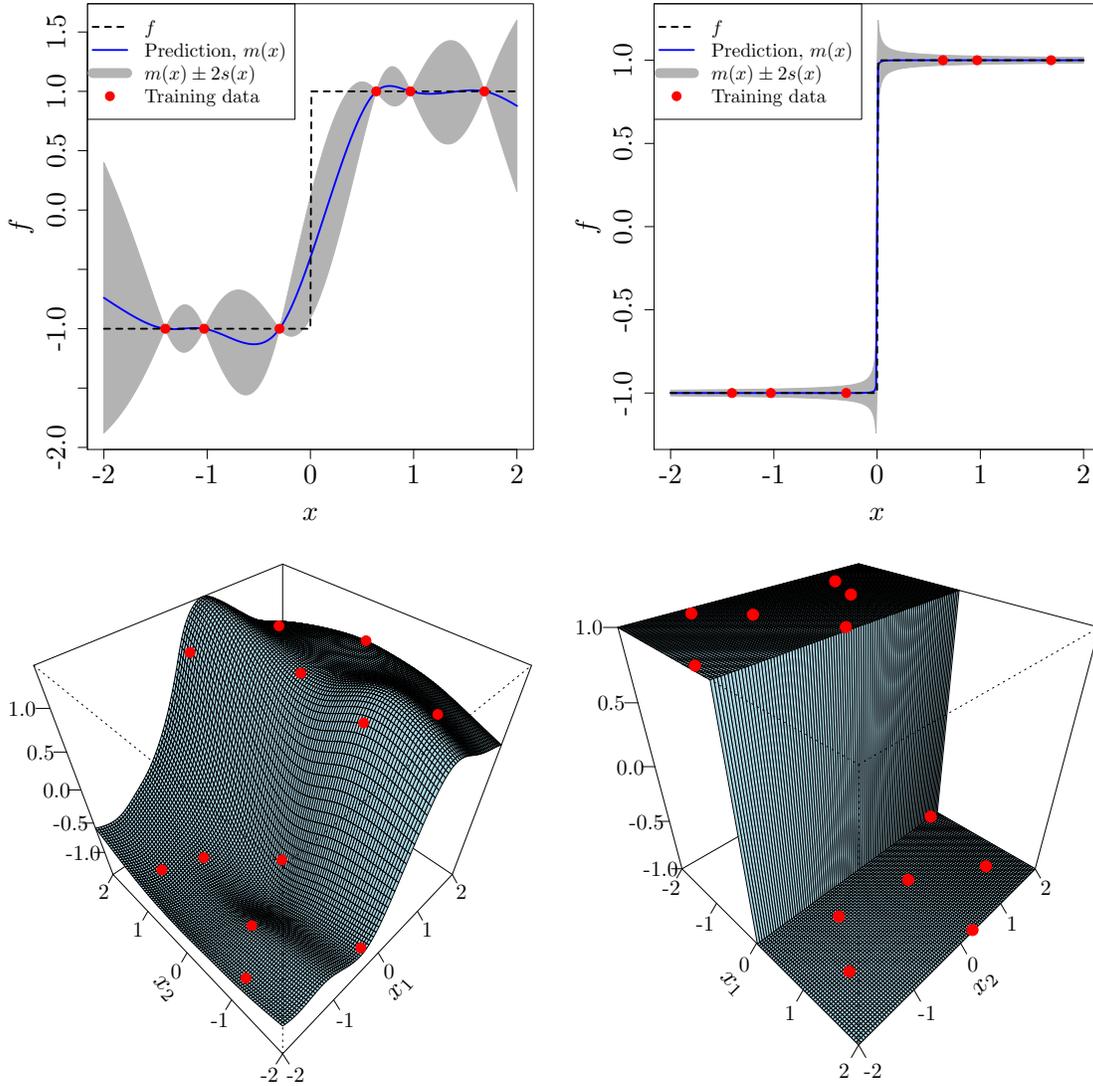
Figure 3: Emulation of the step-function f defined in (13) with the Matérn 3/2 (left panel) and NN (right panel) kernels in $1D$ (top row) and $2D$ (bottom row). Red points are the training data. In the NN kernel, $1D$: $\hat{\sigma}_1 = 10^3$ which is the upper bound in the likelihood optimisation. $2D$: $\hat{\sigma}_1 = 10^3$ (corresponding to x_1) and $\hat{\sigma}_1 = 10^{-2}$ (corresponding to x_2) which is the upper bound in the likelihood optimisation.

9
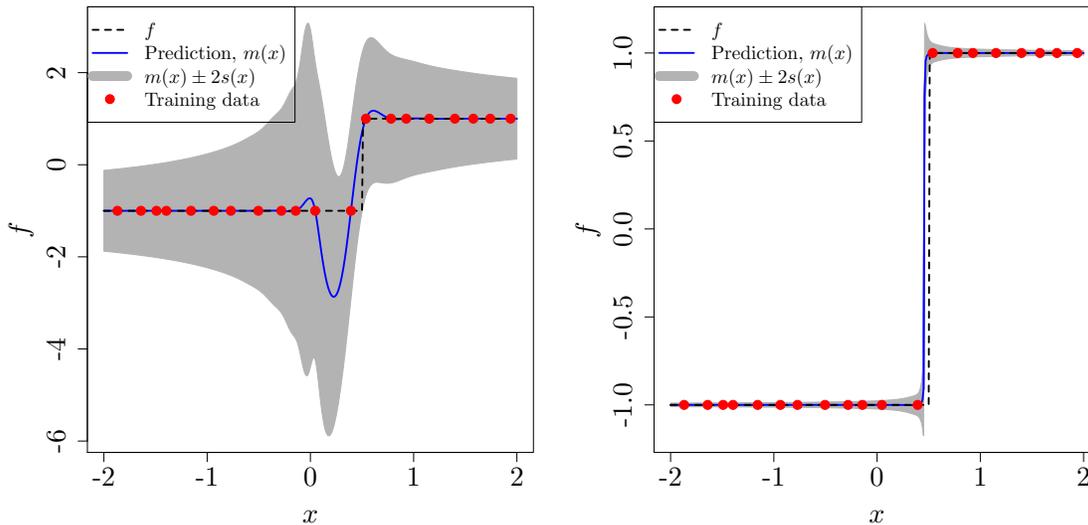
Figure 4: Left: The NN kernel given by Equation (11) is not able to well-approximate f (dashed red) whose discontinuity is at 0.5. Right: The modified NN kernel based on Equation (14) can well-model f. The MLE estimation of τ is 0.457 which is the estimated location of the discontinuity.

a suitable length-scale function should have "small" values when $x \in [0, 0.3]$ and larger values when $x \in [0.3, 1]$. The length-scale used for the Gibbs kernel (right picture) is of the form $l(x) = c_1 x^2 + c_2$ whose unknown parameters c_1 and c_2 are estimated by MLE. This choice of the length-scale allows the GP to predict f with a higher accuracy in comparison to the GP with the Matérn 3/2 kernel (left picture). The estimated parameters of the length-scale are $\hat{c}_1 \approx 45.63$ and $\hat{c}_2 \approx 0.11$ which are in line with the nonstationarity of f.

In this paper, the Gibbs kernel with sigmoid shaped length-scale is employed to model discontinuities. We conduct a simple experiment with four different sigmoid functions, as the length-scale of the Gibbs kernel, to compare their performances in emulating the step-function given by Equation (13) in $2D$. The length-scales are: error function (*erf*), *logistic*, *hyperbolic tangent (tanh)* and *arctangent (arctan)*, see Table 2. They are slightly modified to be positive: the logistic function, for example, is of the form $l(\mathbf{x}) = \frac{1}{1+\exp(c_1 \mathbf{e}_j \mathbf{x})} + c_2$ where c_1 and $c_2 > 0$ are constant parameters estimated by MLE. The former controls the slope of the transition in the sigmoid function and the latter makes $l(\mathbf{x})$ strictly positive. Also, $\mathbf{e}_j$ (in this example $\mathbf{e}_1 = (1, 0)^\top$) determines the j-th axis in which the function is discontinuous.

The accuracy of prediction is measured by the root mean square error (RMSE) criterion defined as

$$RMSE = \sqrt{\frac{1}{n_t} \sum_{t=1}^{n_t} \left(f(\mathbf{x}^t) - \hat{f}(\mathbf{x}^t) \right)^2} \tag{17}$$

10
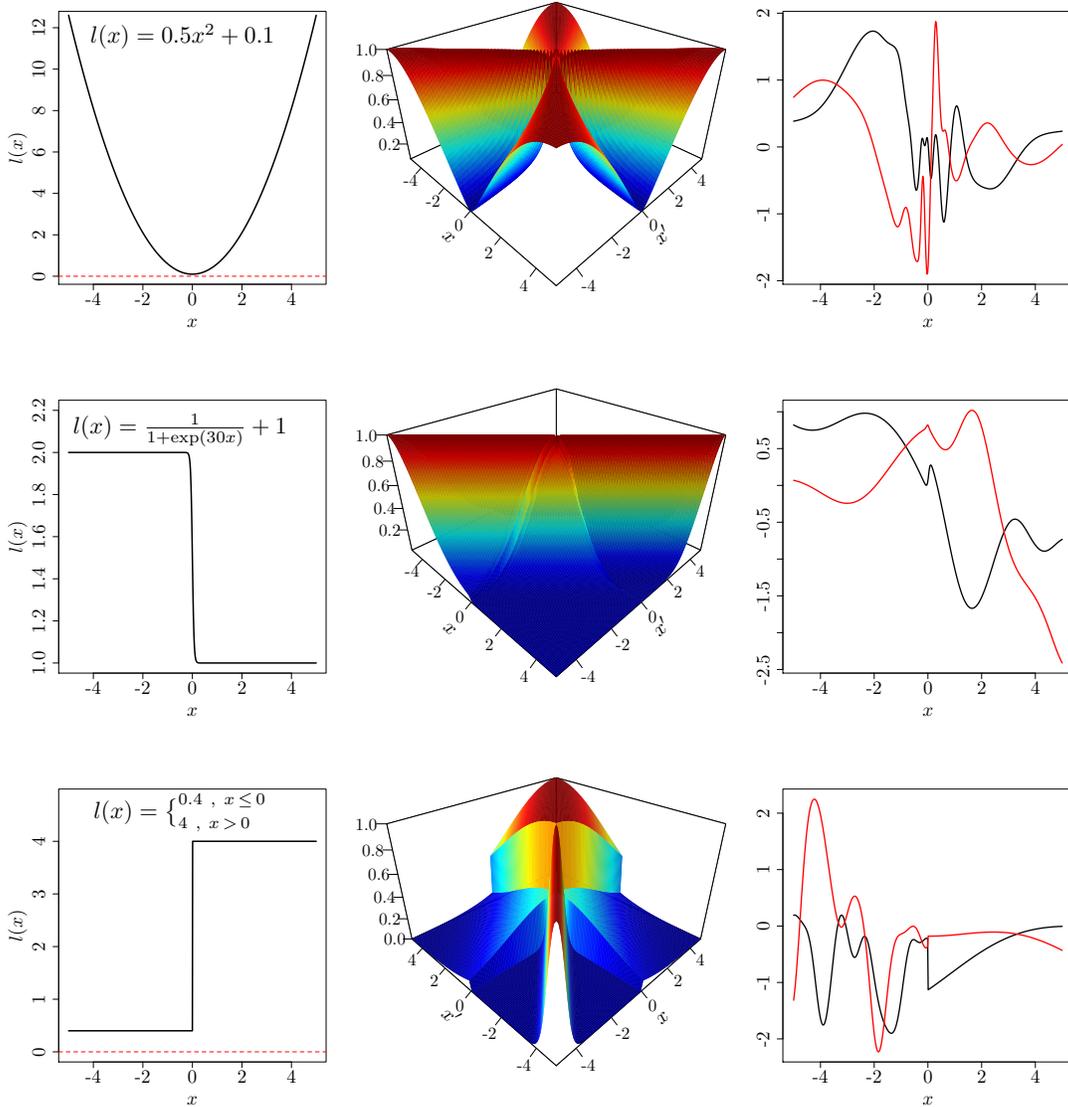
Figure 5: Left panel: three different length-scale functions. Middle panel: shapes of the Gibbs kernel based on the corresponding length-scale functions. Right panel: two GP sample paths with the Gibbs kernel on the left. With the Gibbs kernel, one can model both nonstationary (first row) and discontinuous (second and third rows) functions.
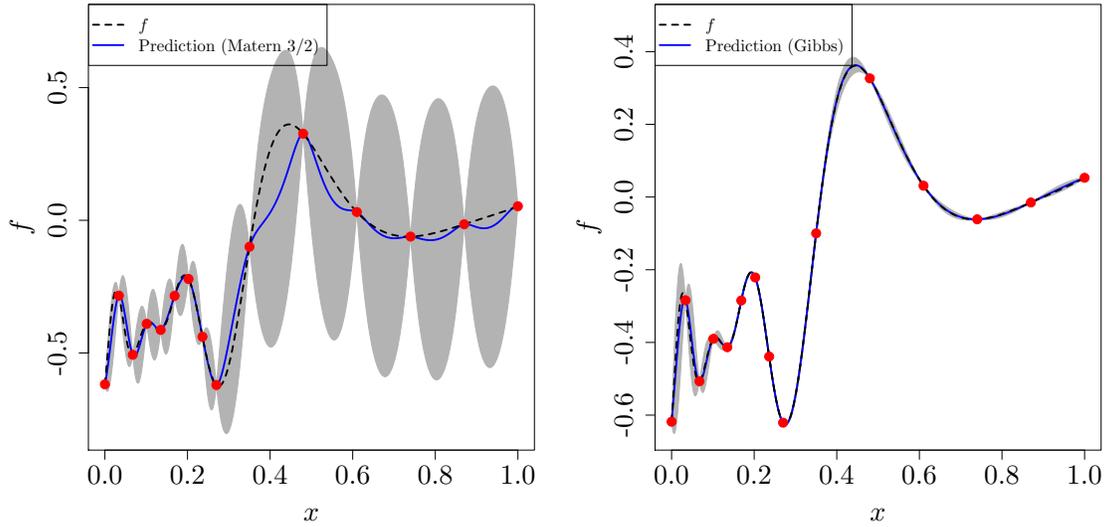
Figure 6: GP prediction (solid blue) of a nonstationary function (dashed) with the Matérn 3/2 (left) and Gibbs (right) kernels. The function is defined as $f(x) = \sin\left(30(x-0.9)^4\right)\cos\left(2(x-0.9)\right) + \frac{(x-0.9)}{2}$ [32] which varies more quickly in the region $x \in [0, 0.3]$ than in the region $[0.3, 1]$. The length-scale function used in the Gibbs kernel is $l(x) = c_1 x^2 + c_2$ whose parameters are estimated by MLE: $\hat{c_1} \approx 45.63$ and $\hat{c_2} \approx 0.11$. The shaded area represents the prediction uncertainty and the red points are the training data.

where $\mathbf{x}^t$ and n_t represent the tth test point and the size of test set, here $n_t = 50$. For each length-scale, the emulation procedure is repeated 70 times with different training sets, each of size 20, and the RMSE criterion is measured. The mean of 70 RMSEs is demonstrated in Table 2 where the Gibbs kernel with the arctangent length-scale function has the lowest RMSE and, therefore, best performance.

	erf	logistic	tanh	arctan
RMSE	0.031	0.063	0.033	0.029

Table 2: Mean of 70 RMSEs associated with four length-scale functions; 1) Error function: $\mathrm{erf}(c_1\mathbf{e}_j\mathbf{x}) + c_2$; $c_2 > 1$. 2) Logistic function: $\frac{1}{1+\exp(c_1\mathbf{e}_j\mathbf{x})} + c_2$; $c_2 > 0$. 3) Hyperbolic tangent: $\tanh(c_1\mathbf{e}_j\mathbf{x}) + c_2$; $c_2 > 1$. 4) Arctangent: $\arctan(c_1\mathbf{e}_j\mathbf{x}) + c_2$; $c_2 > \pi$. f is the step-function given by Equation (13) in $2D$, $\mathbf{e}_{j=1} = (1,0)^\top$. 70 different training sets are used to build $\hat{f}$ and each time the RMSE criterion is calculated based on a fixed test set of size 50.

5   Transformation of the input space (warping)

In this section, *warping* or *embedding* is studied as an alternative approach to emulate functions with discontinuities. The method first uses a non-linear parametric function to map the input space into a feature space. Then, a GP with a standard kernel is applied to approximate the map from the feature space to the output space [19, 5]. A similar idea is used in [30] where the transformation is performed on the output space to model non-Gaussian processes.

In warping, we assume that f is a composition of two functions

$$f = G \circ M : \ M : \mathcal{D} \longmapsto \mathcal{D}' \ , \ G : \mathcal{D}' \longmapsto \mathcal{F}, \tag{18}$$

where M is the transformation function and $\mathcal{D}'$ represents the feature space. The function G is approximated by a GP relying on the training set $\{\tilde{\mathbf{X}}, \mathbf{Y}\}$ in which $\tilde{\mathbf{X}} = \{M(\mathbf{x}^1), \ldots, M(\mathbf{x}^n)\}$. Notice that $\mathcal{D}$ and $\mathcal{D}'$ need not have the same dimensionality [19]. For example, if the squared exponential kernel $k_{SE} : \mathbb{R} \times \mathbb{R} \longmapsto \mathbb{R}$ is composed with $M(x) = \left[\cos(\frac{2\pi x}{T}), \sin(\frac{2\pi x}{T})\right]^\top \in \mathbb{R}^2$, the result is a periodic kernel with period T [29, 13]. In practice, a parametric family of M is selected and its parameters are estimated together with the kernel parameters via MLE.

Such modelling is equivalent to emulate f with a GP whose covariance function $\tilde{k}$ is

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = k\left(M(\mathbf{x}), M(\mathbf{x}')\right). \tag{19}$$

Note that $\tilde{k}$ is generally nonstationary even if k is a stationary kernel, see Figure 7. The prediction (conditional mean) and the associated uncertainty (conditional variance) at

13

a generic point $\mathbf{x}$ read

$$m(\mathbf{x}) = \hat{\tilde{\mu}} + \tilde{\mathbf{k}}(\mathbf{x})^\top \tilde{\mathbf{K}}^{-1}(\mathbf{y} - \hat{\tilde{\mu}}\mathbf{1}) \tag{20}$$

$$s^2(\mathbf{x}) = \tilde{k}(\mathbf{x}, \mathbf{x}) - \tilde{\mathbf{k}}(\mathbf{x})^\top \tilde{\mathbf{K}}^{-1}\tilde{\mathbf{k}}(\mathbf{x}) + \frac{\left(1 - \mathbf{1}^\top \tilde{\mathbf{K}}^{-1}\tilde{\mathbf{k}}(\mathbf{x})\right)^2}{\mathbf{1}^\top \tilde{\mathbf{K}}^{-1}\mathbf{1}}, \tag{21}$$

where $\hat{\tilde{\mu}}$ is obtained by Equation (5) replacing k with $\tilde{k}$. Other elements in the above equations are calculated in the same way.
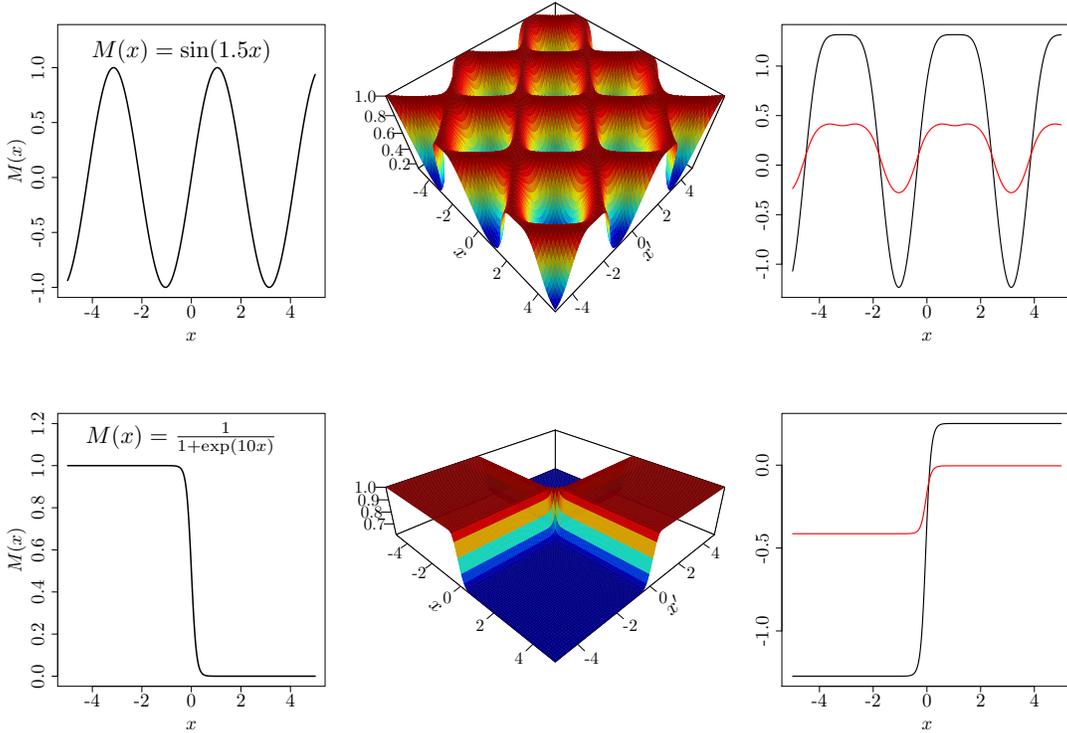
Figure 7: Left panel: two different transformation functions, $M(x)$. Middle panel: shapes of the warped kernel $\tilde{k}(x, x') = k(M(x), M(x'))$ in which k is the squared exponential kernel. Right panel: two sample paths of a GP with the covariance function $\tilde{k}$. As can be seen, a sigmoid transformation function is a suitable choice for modelling step-discontinuities.

According to Figure 7 (second row), a sigmoid transformation is a suitable choice for modelling step-discontinuities. We conduct a similar experiment, explained in Section 4, to test the performance of erf, logistic, hyperbolic tangent and arctangent functions as the transformation mappings. The results are shown in Table 3 in which the arctangent function has the best performance.

14

	erf	logistic	tanh	arctan
RMSE	0.049	0.101	0.069	0.007

Table 3: Mean of 70 RMSEs in emulating step-function f (Equation (13)) in $2D$ using the warped kernel $\tilde{k}(\mathbf{x}, \mathbf{x}') = k\left(M(\mathbf{x}), M(\mathbf{x}')\right)$ where k is the square exponential covariance kernel. Four different sigmoid functions are considered for $M(.)$: erf, logistic, tanh and arctan. See Table 2 for the form of these mappings.

6   Conclusions

Gaussian processes are mainly used to predict smooth, continuous functions. However, there are many situations in which the output of a complex computer code has discontinuity, e.g. when tipping points occur. This paper deals with the problem of emulating step-discontinuous functions using GPs. Several methods, including two covariance kernels and the idea of transforming the input space (warping), are proposed. The two covariance functions are neural network and Gibbs kernels whose properties are demonstrated using several examples. Our experiments show that these techniques have superior performance to GPs with standard kernels in capturing sharp jumps.

Acknowledgements

References

[1] A. Adcroft, C. Hill, J-M. Campin, J. Marshall, and P. Heimbach. Overview of the formulation and numerics of the MIT GCM. In *Seminar on recent developments in numerical methods for atmospheric and ocean modelling*, pages 139–150, Shinfield Park, Reading, September 2004.

[2] P.J. Birrell, G. Ketsetzis, N.J. Gay, B.S. Cooper, A.M. Presanis, R.J. Harris, A. Charlett, X-S. Zhang, P.J. White, R.G. Pebody, and Angelis D. De. Bayesian modeling to unmask and predict influenza A/H1N1 pdm dynamics in London. *Proceedings of the national academy of sciences of The United States of America*, 108:18238–18243, 2011.

[3] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010.

[4] C.C.S. Caiado and M. Goldstein. Bayesian uncertainty analysis for complex physical systems modelled by computer simulators with applications to tipping points. *Communications in Nonlinear Science and Numerical Simulation*, 26(1):123 – 136, 2015.

[5] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold Gaussian processes for regression. *ArXiv e-prints*, 2014.

[6] Peter Challenor. The probability of rapid climate change. *Significance*, 1(4):155–158, 2004.

[7] Proctor CJ., Boche D., Gray DA., and Nicoll JAR. Investigating interventions in alzheimers disease with computer simulation models. *PLoS ONE*, 8(9):e73631, 2013.

[8] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, Computational and Biological Learning Laboratory, University of Cambridge, 2014.

[9] Alexander I. J. Forrester, Andras Sobester, and Andy J. Keane. *Engineering design via surrogate modelling - A practical guide*. Wiley, 2008.

[10] Mark N. Gibbs. *Bayesian Gaussian processes for regression and classification*. PhD thesis, Department of Physics, University of Cambridge, 1997.

[11] Robert B. Gramacy and Herbert K. H. Lee. Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130, 2008.

[12] Peter J. Green. Reversible jump markov chain monte carlo computation and Bayesian model determination. *Biometrika*, 82:711–732, 1995.

[13] N. Haji-Ghassemi and M.P. Deisenroth. Approximate inference for long-term forecasting with periodic Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2014.

[14] Dave Higdon, James Gattiker, Brian Williams, and Maria Rightley. Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583, 2008.

[15] Bertrand Iooss and Paul Lemaître. A review on global sensitivity analysis methods. In *Uncertainty Management in Simulation-Optimization of Complex Systems: Algorithms and Applications*. Springer, 2015.

[16] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[17] Marc C. Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

[18] Brian A. Lockwood and Mihai Anitescu. Gradient-enhanced universal kriging for uncertainty propagation. *Nuclear Science and Engineering*, 170(2):168–195, 2012.

[19] D. J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, pages 133–166. Springer-Verlag, 1998.

[20] Radford M. Neal. Regression and classification using Gaussian process priors. pages 475–501. Bayesian Statistics 6, Oxford University Press, 1998.

[21] H. B. Nielsen, S. N. Lophaven, and J. Søndergaard. DACE – A Matlab kriging toolbox – version 2.0. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2002.

[22] Jeremy Oakley. Estimating percentiles of uncertain computer code outputs. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 53(1):83–93, 2004.

[23] Jeremy E. Oakley and Anthony O'Hagan. Probabilistic sensitivity analysis of complex models: a Bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(3):751–769, 2004.

[24] A. O'Hagan. Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering & System Safety*, 91(10-11):1290–1300, 2006.

[25] C. A. Pope, J. P. Gosling, S. Barber, J. Johnson, T. Yamaguchi, G. Feingold, and P. Blackwell. Modelling spatial heterogeneity and discontinuities using Voronoi tessellations. *ArXiv e-prints*, 2018.

[26] Luc Pronzato and Werner G. Müller. Design of computer experiments: space filling and beyond. *Statistics and Computing*, 22(3):681–701, 2012.

[27] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning (adaptive computation and machine learning)*. The MIT Press, 2005.

[28] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

[29] Matthias Seeger. *Bayesian Gaussian process models: PAC-Bayesian generalisation error bounds and sparse approximations*. PhD thesis, University of Edinburgh, 2003.

[30] Edward Snelson, Carl Edward Rasmussen, and Zoubin Ghahramani. Warped Gaussian processes. In *Advances in Neural Information Processing Systems (NIPS)*, pages 337–344. MIT Press, 2004.

[31] Christopher K. I. Williams. Computing with infinite networks. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 295–301. MIT Press, 1997.

[32] Ying Xiong, Wei Chen, Daniel Apley, and Xuru Ding. A non-stationary covariance-based kriging method for metamodelling in engineering design. *International Journal for Numerical Methods in Engineering*, 71(6):733–756, 2007.

Appendix A   Covariance functions/kernels

Covariance kernels are positive definite (PD) functions. The symmetric function $k : \mathcal{D} \times \mathcal{D} \longmapsto \mathbb{R}$ is PD if

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j k(\mathbf{x}^i, \mathbf{x}^j) \geq 0$$

for any $N \in \mathbb{N}$ points $\mathbf{x}^1, \ldots, \mathbf{x}^N \in \mathcal{D}$ and $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^\top \in \mathbb{R}^N$. If k is a PD function, then the $N \times N$ matrix $\mathbf{K}$ whose elements are $\mathbf{K}_{ij} = k(\mathbf{x}^i, \mathbf{x}^j)$ is a positive semidefinite matrix because $\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j \mathbf{K}_{ij} \geq 0$.

Checking the positive definiteness of a function is not easy. One can combine the existing kernels to make a new one. For example, if k_1 and k_2 are two kernels, the function k obtained by the following operations is a valid covariance kernel:

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$$
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') \times k_2(\mathbf{x}, \mathbf{x}')$$
$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}'), \ c \in \mathbb{R}^+$$
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + c, \ c \in \mathbb{R}^+$$
$$k(\mathbf{x}, \mathbf{x}') = g(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')g(\mathbf{x}') \text{ for any function } g(.).$$

We refer the reader to [27, 8] for a detailed discussion about the composition of covariance functions. It is also possible to compose kernels with a function as explained in Section 5.

Usually, a covariance function depends on some parameters $\mathbf{p}$ which are unknown and need to be estimated from data. In practice, a parametric family of k is chosen first. Then the parameters are estimated via maximum likelihood estimation (MLE), cross-validation or (full) Bayesian approaches [27]. In the sequel, we describe the MLE method as is used in this paper.

The likelihood function measures the adequacy between a probability distribution and the data; a higher likelihood function means that observations are more consistent with the assumed distribution. In the GP framework, as observations are presumed to have the normal distribution, the likelihood function is

$$p\left(\mathbf{y}|\mathbf{X}, \mathbf{p}, \mu\right) = \frac{1}{(2\pi)^{n/2}|\mathbf{K}|^{1/2}} \exp\left(-\frac{(\mathbf{y} - \mu\mathbf{1})^\top \mathbf{K}^{-1} (\mathbf{y} - \mu\mathbf{1})}{2}\right), \tag{22}$$

where $|\mathbf{K}|$ is the determinant of the covariance matrix. In the above equation, if μ is unknown, it is replaced with its estimate given by Equation (5).

Usually for optimisation, it is more convenient to work with the natural logarithm of the likelihood (log-likelihood) function which is

$$\ln p\left(\mathbf{y}|\mathbf{X}, \mathbf{p}, \mu\right) = -\frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathbf{K}| - \frac{(\mathbf{y} - \mu\mathbf{1})^\top \mathbf{K}^{-1} (\mathbf{y} - \mu\mathbf{1})}{2}. \tag{23}$$

Maximising (23) is a challenging task as the log-likelihood function is often nonconvex with multiple maxima. To do so, numerical optimisation algorithms are often applied. We refer the reader to [21, 9] for further information.