Bayesian shape-constrained curve-fitting with Gaussian processes

Eduardo Adame Salles & Luiz Max Carvalho
School of Applied Mathematics, Getulio Vargas Foundation, Brazil.

12th December 2024

Abstract

In many applications, one is interested in reconstructing a function f when only few (potentially very noisy) evaluations $f(x)$ are available, usually due to budget restrictions. When information about the "shape" of f is available, e.g., whether it is monotonic, convex/concave, etc., it is desirable to include this information into the curve-fitting procedure. Here we build on the Gaussian process literature to propose a comprehensive framework for flexibly modeling f and its first two derivatives given evaluations of f, f', f'' at potentially irregular grids. We show how to include shape-constraints in a principled way through the prior and apply the developed methods to function emulation for noisy Markov chain Monte Carlo. In the context of the Normalized Power Prior (NPP), we obtain some intuition about the information that the derivative processes bring to the regression, although this may propagate uncertainty and harm the quality of the fit. Furthermore, we evaluate how the proposed method compares to SCAM, one of the main frequentist methods for function emulation with shape constraints. .

Key-words: Bayesian Curve Fitting; Gaussian Processes; Shape Constraints; Function Emulation; Noisy Markov Chain Monte Carlo.

1 Background

In many scenarios, evaluating a computationally intensive function $f : \mathcal{D}(f) \to \mathbb{R}^d$ across an extensively large collection of points S is necessary. A typical strategy in such instances is to construct a more cost-effective estimator $\hat{f}(q) \approxeq f(q)\ \forall\ q \in Q \subseteq S$, and then use $\hat{f}(q)$ for approximation over $q \in (\mathcal{D}(f) \setminus Q)$, covering the remaining points in the domain. There are numerous established techniques that aim for the same goal. Two of the most prevalent strategies are:

1. Restricting the set of potential functions to a predefined parametric family (e.g., linear, exponential, etc.); or

2. Introducing uncertainty across all feasible functions within the space containing f, and preferentially weighting those considered more probable (e.g., a particular class of functions).

Our study concentrates on **Gaussian processes (GPs)**, which facilitate direct uncertainty quantification from our estimator, similar to some models from the second approach. This enables the computation of probabilities such as $\mathbb{P}[\hat{f}(a) \in A]$ for each $a \in \mathcal{D}(\hat{f})$ and every $A \subseteq \mathrm{Im}(\hat{f})$. A proper Gaussian processes discussion is reserved to **??**, as well as its foundation in Bayesian statistics.

Furthermore, we aim to improve the assignment of prior probabilities by incorporating shape constraints on f, like monotonicity and convexity. The concept involves introducing

derivative information of $\hat{f}$, considering its correlation with $\hat{f}$ itself. On the other hand, a derivative of a GP is still a GP, which help us to operate within a Gaussian Process framework. This methodology, initially explored in Rasmussen and Williams (2005); Riihimäki and Vehtari (2010); and Wang and Berger (2016), will be discussed in subsection 2.3.

1.1 Related work and contributions

The initial contribution towards incorporating shape constraints in a Gaussian process was discussed in a section of Rasmussen and Williams (2005), focusing on the probability distributions of GP derivatives. In contrast, Riihimäki and Vehtari (2010) made a significant stride by establishing a framework for shape constraints, leveraging first derivative data for GP fitting and devising a method to enforce monotonicity through virtual points.

Subsequently, Wang and Berger (2016) extended these ideas to n-th order derivatives, paving the way for more robust constraints. Furthermore, this study introduced advanced techniques within the Bayesian framework to effectively implement these constraints.

2 Gaussian Processes

Introducing the main focus of this work, consider the model:

$$Y = f(X) + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \sigma^2 I), \tag{1}$$

where you aim to find a function f that best fits your data according to some criterion. As discussed in section 1, there are many ways to approach this problem, but from a parametric perspective, one would assume a specific parametric form for f. This implies that there exists a map $T : \Theta \to \mathcal{F}$ such that for every $\theta \in \Theta$, we can obtain a function f_θ from its function space $\mathcal{F}$, then find a parameter $\hat{\theta}$ that minimizes a certain loss function $L(\theta, \hat{\theta})$, preferably convex. The most traditional approaches include maximum likelihood estimation (frequentist) and maximum a posteriori estimation (Bayesian), which have some favorable statistical properties.

In contrast to this approach, we introduce a Gaussian Process as a non-parametric method for modeling f. Here, f is modeled as a random function, rather than a deterministic one, which allows us to quantify the uncertainty associated with f (and its potential derivatives). Generally, a Gaussian Process (GP) is described as:

$$Y \mid (\boldsymbol{f}, \boldsymbol{X}) \sim \mathcal{N}(\boldsymbol{f}, \sigma^2 I), \tag{2}$$

$$\boldsymbol{f} \mid \boldsymbol{X} \sim \mathcal{GP}(\mu(\boldsymbol{X}), K(\boldsymbol{X}, \boldsymbol{X})), \tag{3}$$

where μ is called the mean function and K is the covariance function. We denote that $\boldsymbol{f}$ represents f applied to each entry of $\boldsymbol{X}$, the same goes for $\mu(\boldsymbol{X})$, and $K(\boldsymbol{X}, \boldsymbol{X})$ denotes applying K to each pairwise combination of entries in X.

However, to support our goal of extending the model to include shape constraints, we need to formally define and review some basic properties of Gaussian Processes.

2.1 Definition and Key Properties

For our purposes, we can define a Gaussian Process as follows:

Definition 2.1 (Gaussian Process (GP))**.** *A Gaussian Process is a collection of random variables, such that any finite number of them has a joint Normal distribution. For a function* $f : \mathcal{X} \to \mathbb{R}$*, modeled as* $f(\boldsymbol{X}) \mid \boldsymbol{X} \sim \mathcal{GP}(m(\boldsymbol{X}), k(\boldsymbol{X}, \boldsymbol{X}'))$*, where* $m(\boldsymbol{X})$ *and* $k(\boldsymbol{X}, \boldsymbol{X}')$ *represent the mean and kernel (or covariance) functions applied to each input* $X \in \boldsymbol{X}$*, respectively. The kernel function must be symmetric and satisfy* $k(X, X) > 0$ *for each* X*.*

Considering the model $\boldsymbol{Y} \mid (\boldsymbol{f}, \boldsymbol{X}) \sim \mathcal{N}(\boldsymbol{f}, \sigma^2 I)$ with $\boldsymbol{f} \mid \boldsymbol{X} \sim \mathcal{GP}(m(\boldsymbol{X}), k(\boldsymbol{X}, \boldsymbol{X}'))$, where $\sigma^2 > 0$ is a known constant, $\boldsymbol{Y}, \boldsymbol{X} \in \mathbb{R}^n$ are random vectors, and I denotes the identity matrix in $\mathbb{R}^{n \times n}$, we can take a Bayesian approach and compute a posterior distribution $\boldsymbol{f} \mid \boldsymbol{X}, \boldsymbol{Y}$ for a sample size of n. In this case, we can sample from any unobserved $\boldsymbol{Y}_\star$ for a new observation $\boldsymbol{x}_\star$ via the posterior predictive distribution $p(Y_\star \mid \boldsymbol{y}, \boldsymbol{x}_\star, \boldsymbol{x})$, which is also normally distributed (see appendix A).

2.1.1 Kernel Functions

Also called covariance functions, these are fundamental to Gaussian Process modeling.

The choice of kernel function determines the assumed characteristics of the latent function. For example, the squared exponential kernel encodes the assumption that the latent function is smooth, whereas the Matérn kernel encodes the assumption that the latent function is not smooth.

The following properties are necessary for a function to be considered a kernel:

1. Symmetry: $k(x, x') = k(x', x)$;
2. Positive semi-definiteness: $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$ for any $n \in \mathbb{N}$, $c_1, \cdots, c_n \in \mathbb{R}$, and $x_1, \cdots, x_n \in \mathbb{R}$;
3. Stationarity: $k(x, x') = k(x - x')$;
4. Isotropy: $k(x, x') = k(|x - x'|)$.

For each type of problem, one kernel may be more suitable than another. Some common examples of kernels are:

1. Radial Basis Function (RBF): $k(x, x';\ \sigma^2, l) = \sigma^2 \exp(-\frac{1}{2l^2} d(x, x'))$, which is often the default choice;
2. Matérn Kernel: $k(x, x';\ \sigma^2, l, \nu) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} (\sqrt{2\nu} \frac{d(x,x')}{l})^\nu K_\nu (\sqrt{2\nu} \frac{d(x,x')}{l})$, where K_ν is the modified Bessel function of the second kind and is often a less generic alternative to the RBF;
3. Linear Kernel: $k(x, x'; \sigma^2) = \sigma^2 x^T x'$, commonly used in linear regression problems;
4. Periodic Kernel: $k(x, x'; \sigma^2, l, p) = \sigma^2 \exp(-\frac{2}{l^2} \sin^2(\frac{\pi}{p} d(x, x')))$, well-suited for problems with seasonality;
5. Cosine Kernel: $k(x, x'; \sigma^2, p) = \sigma^2 \cos(\frac{\pi}{p} d(x, x'))$, another less smooth alternative to the periodic kernel.

where $x, x' \in \mathcal{X}$, $d(x, x')$ is the Euclidean distance between x and x', σ^2 is the variance, l is the length scale, ν is the smoothness parameter, and p is the period.

2.1.2 Optimizing Hyperparameters

To optimize the hyperparameters of a Gaussian Process, we can maximize the marginal likelihood (also called *type II maximum likelihood*).

This involves maximizing the expected likelihood under the prior, i.e., $p(Y \mid f(X)) = \mathbb{E}_{f \sim GP(0,k)}[\mathcal{N}(Y \mid f(X), \sigma^2 I)]$. More specifically, with the kernel function's parameters denoted generically by ω, the optimal parameters can be calculated as follows:

$$\begin{aligned}\hat{\omega} &= \arg\max_{\omega \in \Omega} \log p(Y \mid f(X)) \\ &= \arg\max_{\omega \in \Omega} \log \mathcal{N}(Y \mid 0, k(X,X) + \sigma^2 I) \\ &= -\frac{1}{2} \log\det\left(k(X,X) + \sigma^2 I\right) - \frac{1}{2} Y^T \left(k(X,X) + \sigma^2 I\right)^{-1} Y \end{aligned} \tag{4}$$

It's also important to note that the procedure assumes the observational noise σ^2 is constant. In this case, we could estimate the kernel parameters for each value of σ^2 using the training set and choose the σ^2 that results in the highest evidence. Another approach is to optimize σ^2 jointly with ω, as shown below:

$$\sigma^2, \hat{\omega} = \arg\max_{\omega \in \Omega, \sigma^2 \in \mathbb{R}} -\frac{1}{2} \log\det\left(k(X,X) + \sigma^2 I\right) - \frac{1}{2} Y^T \left(k(X,X) + \sigma^2 I\right)^{-1} Y \tag{5}$$

However, joint optimization of σ^2 and ω can be more challenging, and various heuristics can be used to find a local optimum.

2.2 Variations of Gaussian Processes

Beyond the traditional form of Gaussian Processes, there are some variations that may be useful for different types of problems, such as changing the problem from regression to classification, or adding shape constraints to the model. In this section, we will discuss some of these variations.

2.2.1 Fully Bayesian Gaussian Processes

A method known as *fully Bayesian Gaussian Processes (FBGP)* (as discussed in Riis et al. (2022)) for hyperparameter optimization treats the hyperparameters as random variables and assigns a prior distribution to them.

In this case, fitting the Gaussian Process is done jointly with the optimization of the hyperparameters. That is:

$$p(\boldsymbol{f}, \boldsymbol{\theta} \mid Y, X) \propto p(Y \mid \boldsymbol{f}) p(\boldsymbol{f} \mid \boldsymbol{\theta}, X) p(\boldsymbol{\theta}), \tag{6}$$

and thus the posterior predictive distribution for a new point $X_\star$ is given by:

$$p(Y_\star \mid Y) = \int \int p(Y_\star \mid \boldsymbol{f}_\star) p(\boldsymbol{f}_\star \mid \boldsymbol{\theta}, Y) p(\boldsymbol{\theta} \mid Y) \, \mathrm{d}\boldsymbol{f}_\star \, \mathrm{d}\boldsymbol{\theta},$$

Note that the above integral is taken over all possible values of $f_\star$ and θ, which has infinite dimensions. However, one way to approximate this integral is to recognize that we can reuse the posterior predictive distribution of a traditional GP, reducing the problem to:

$$p(Y_\star \mid Y) = \int p(Y_\star \mid Y, \theta_\star) p(\theta \mid Y) \, \mathrm{d}\theta,$$

With this, we can use the following approximation by sampling from $p(\theta \mid Y)$ using MCMC:

$$p(Y_\star \mid Y) \approx \frac{1}{S} \sum_{s=1}^{S} p(Y_\star \mid Y, \theta_s), \ \theta_s \sim p(\theta \mid Y).$$

2.2.2 Gaussian Processes for Classification

Although Gaussian Processes are primarily suited for regression problems, they can be adapted for classification tasks. In the case of binary classification, we can model the likelihood as a Bernoulli distribution, with the link function being the sigmoid function. That is:

$$p(Y_i \mid f_i) = \sigma(f_i)^{Y_i} (1 - \sigma(f_i))^{1 - Y_i}. \tag{7}$$

It follows that:

$$\begin{aligned} p(f \mid Y, X) &\propto \log p(Y \mid f) + \log p(f \mid X), \\ &= p(Y \mid f) - \frac{1}{2} f^T k(X, X)^{-1} f - \frac{1}{2} \log |k(X, X)| - \frac{N}{2} \log 2\pi \end{aligned}$$

However, to facilitate fitting and using the posterior predictive distribution, it is common to use a Laplace approximation for the posterior distribution of f.

Laplace Method The Laplace method is an optimization technique that approximates the posterior distribution of a Gaussian Process by a Normal distribution. The idea is to approximate the posterior distribution $p(f \mid Y, X)$ with a Normal distribution $q(f)$, obtained by maximizing the marginal likelihood $p(Y \mid f, X)$ (Riihimäki and Vehtari, 2013).

Latent and Approximate Gaussian Processes In many cases, exact inference of a Gaussian Process can be computationally expensive, especially when the number of observations is large. To address this issue, it is common to use approximation methods, such as the Laplace method, the latent variable method, or numerical integration methods (Williams and Barber, 1998).

2.3 Introducing Shape Constraints

The main result that allows us to introduce the approach of shape-constrained Gaussian Processes (SCGPs) was also introduced by Rasmussen and Williams (2005), which is:

Property 2.1 (Derivative of a GP is a GP). *Let* $(f \mid X) \sim \mathcal{GP}(m(X), k(X, X'))$*, then for each partial derivative* $\frac{\partial f}{\partial X_d}$*, we have:*

$$\operatorname{Cov}\left(f_i, \frac{\partial f_j}{\partial X_{dj}}\right) = \frac{\partial}{\partial X_{dj}} k(X_i, X_j) \qquad \text{and} \qquad \operatorname{Cov}\left(\frac{\partial f_i}{\partial X_{di}}, \frac{\partial f_j}{\partial X_{ej}}\right) = \frac{\partial^2}{\partial X_{di} \partial X_{ej}} k(X_i, X_j). \tag{8}$$

Linearity also holds for the expected values, so:

$$\begin{bmatrix} f \\ f' \end{bmatrix} \mid X \sim \mathcal{GP}\left(\begin{bmatrix} m(X) \\ \frac{\partial}{\partial X} m(X) \end{bmatrix}, \begin{bmatrix} k(X, X') & \frac{\partial}{\partial X'} k(X, X') \\ \frac{\partial}{\partial X} k(X, X') & \frac{\partial^2}{\partial X \partial X'} k(X, X') \end{bmatrix}\right). \tag{9}$$

This holds because the Normal distribution is fully characterized by its mean and covariance matrix.

Considering the posterior predictive distribution and equation (9), we are able to perform shape-constrained Gaussian Process regression using observations of $((x_1, y), (x_2, y'))$ that do not need to be at the same input values.

2.3.1 Conditional SCGPs

In addition to the proposed method of fitting an SCGP through the joint distribution between f and its derivatives, there are other ways to fit an SCGP. One such approach is by Wang and Berger (2016), which suggests fitting a Gaussian Process $Z(\cdot)$ by conditioning on an extension of its derivative process $Z'^{+}(t) = Z'(t) \vee 0$, which results in the following distribution:

$$Z(t) \mid \{Z'^{+}(t)\}_{s=1}^{n} \sim \mathcal{N}(\mu + K^{01}(t, s) K^{11}(s, s)^{-1} Z'^{+}(s), K^{\triangle}(t, t)),$$

where K^{01} and K^{11} are the covariance matrices between Z and Z'^{+} and between Z'^{+}, respectively, and $K^{\triangle} := K^{00} - K^{01}(K^{11})^{-1} K^{01}$.

3 Other shape-constrained models

3.1 Generalised additive models

4 Experiments

To understand in which scenarios the inclusion of shape constraints is beneficial, a series of experiments was conducted.

The final implementation was done in the programming language `Python` using the `GPyTorch` library (Gardner et al., 2021), and it is available at https://github.com/adamesalles/shape-constrained-gaussian-processes, which also contains the results of the experiments conducted.

The choice of `GPyTorch` was due to its ability to perform operations efficiently on a GPU (using Nvidia CUDA), which is essential for large-scale experiments and numerical hyperparameter optimization. Initial prototypes were developed in `Julia` with the `GaussianProcesses.jl` library (Fairbrother et al., 2022) and in R R Core Team (2024) with the `GPfit` library (MacDonald et al., 2015).

The experiments were conducted on a Ryzen 7 5800H (8 cores, 16 threads) with 24GB of RAM and an Nvidia RTX 3050 GPU with 4GB of video memory.

4.1 Normalized Power Prior

The initial application planned for the developed methods is the emulation of functions in noisy Monte Carlo Markov Chain (MCMC) settings. In this context, a relevant example in the literature is the emulation of models using the *Normalized Power Prior* (NPP) (Carvalho and Ibrahim, 2021).

Consider a tempered posterior distribution of the form $p_\alpha(\theta \mid z) \propto l(z \mid \theta)^\alpha \pi(\theta)$ for $\alpha \in [0, 1]$. Our goal is to emulate $f_z(\alpha) = \int_\Omega l(z \mid t)^\alpha \pi(t)\, \mu(\mathrm{d}t)$ with $M = 20$ noisy evaluations, which are computationally costly. In the case where the prior distribution is an NPP and also proper, the function $f_z(\alpha)$ is strictly convex in α.

The datasets used in the experiments are the four combinations of:

- HC and MC (*high curvature and medium curvature*): representing the curvature of the function $f_z(\alpha)$.
- *uniform* and *adaptive*: the evaluation grid is either regular or irregular, using the algorithm 1.

4.2 Comparison between Gaussian Process and SCGP

In this section, we compare the fit of the traditional Gaussian Process with that of the SCGP in high and medium curvature scenarios, evaluated on regular and irregular grids. There are two experiments, assessing the fit capabilities of the proposed methods under variations in curvature and the position of observed values, while reducing the number of observed points.

4.2.1 Variations in Curvature and Position of Observed Values

Our first experiment consists of evaluating the fitting ability of the proposed methods under variations in the curvature and position of observed values.

For the regressions in these $M = 20$ evaluations of $f_z(\alpha)$, we used the Radial Basis Function (RBF) as the kernel function k and $m(x) = 0$ as the mean function. Additionally, for hyperparameter optimization, we used the Adam optimization method (Kingma and Ba, 2017) with a learning rate of 10^{-2}, 1000 iterations for the GP, and 2000 iterations for the SCGP. The MSE is evaluated on a grid of 1000 uniformly spaced points.

In the case where curvature is medium and evaluations are done on a regular grid, figure 1 shows the comparison between Gaussian Process with and without shape constraints under medium curvature.

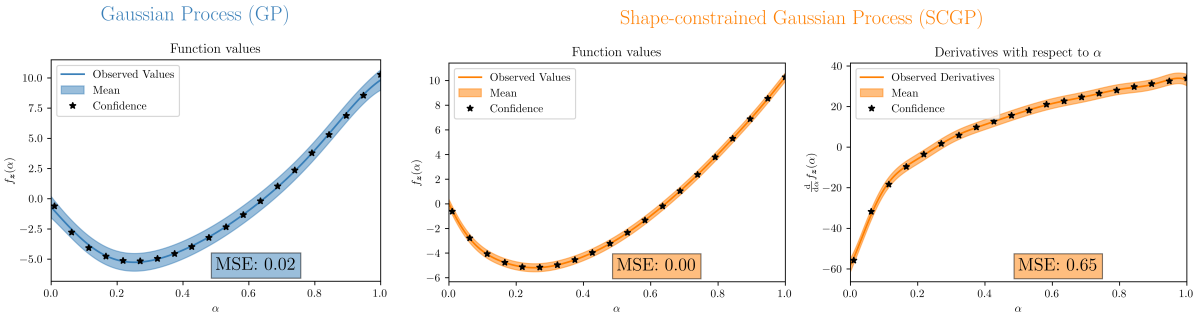

Figure 1: Comparison between Gaussian Process with and without shape constraints under medium curvature and regular grid evaluation. Source: author.

It can be observed that the Gaussian Process with shape constraints is able to capture the curvature of the function $f_z(\alpha)$ more accurately, while the Gaussian Process without shape constraints tends to underestimate the curvature.

Observation 4.1. *Note that the derivative process fitting was satisfactory, which will be important in future experiments. The comparison, however, should be made between the functions $f_z(\alpha)$ rather than between the functions $f_z'(\alpha)$.*

In the case where evaluations are done on an irregular grid, figure 2 shows the comparison between the methods. In this scenario, the SCGP correctly captures the shape constraint, while the traditional GP disregards the convexity of the function. However, both methods show a less satisfactory fit.

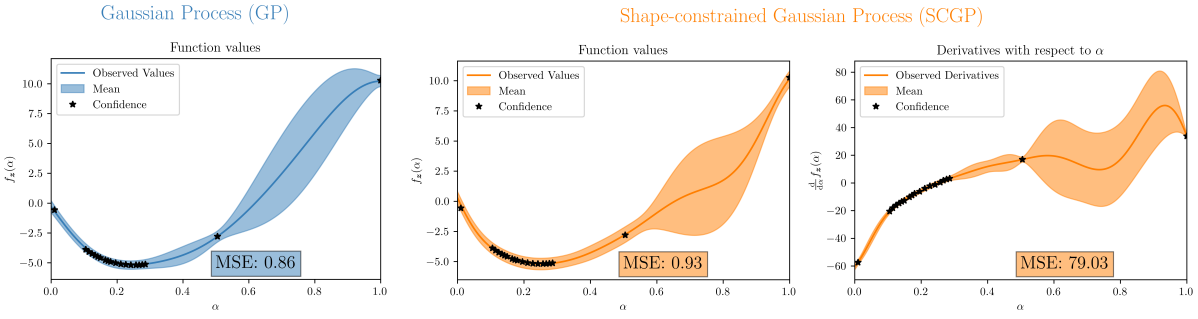

Figure 2: Comparison between Gaussian Process with and without shape constraints under medium curvature and irregular grid evaluation. Source: author.

Evaluating the high curvature scenario, figure 3 shows the comparison in the high curvature scenario with regular grid evaluation. In this case, the SCGP is much better at capturing the curvature of the function $f_z(\alpha)$ compared to the traditional Gaussian Process.

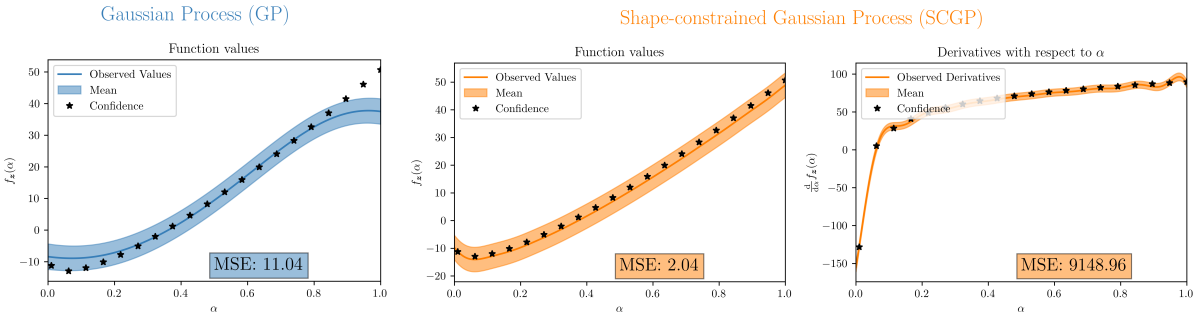

Figure 3: Comparison between Gaussian Process with and without shape constraints under high curvature and regular grid evaluation. Source: author.

In figure 4, both models perform poorly in the adaptive grid scenario with high curvature. In the case of the SCGP, the unsatisfactory fitting of the derivative process is a possible explanation for the model not performing as well as in previous scenarios.

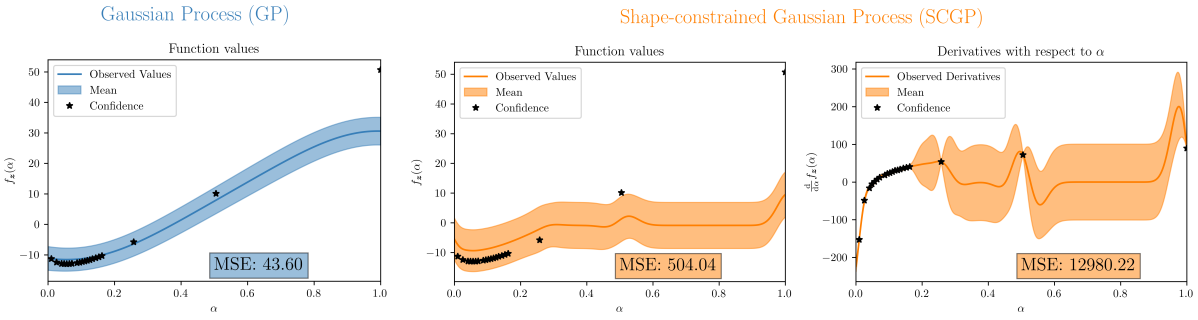

Figure 4: Comparison between Gaussian Process with and without shape constraints under high curvature and irregular grid evaluation. Source: author.

4.2.2 Reducing the Number of Observed Points

The second experiment consists of evaluating the fitting ability of the proposed methods under variations in the number of observed points. For this, we will perform the fitting under medium curvature evaluated on a regular grid (which proved to be the best scenario in section 4.2.1), but with $M = \{6, 8, 10, 12, 14, 16, 18\}$ evaluations of $f_z(\alpha)$.

Observing figure 5, it is possible to see that the traditional Gaussian Process is outperformed starting from 8 observed points, with an MSE approximately 18 times higher than the SCGP. Furthermore, in figure 6, we see that the SCGP is able to fit the derivative process satisfactorily from 12 observed points, outperforming the traditional Gaussian Process in all scenarios.

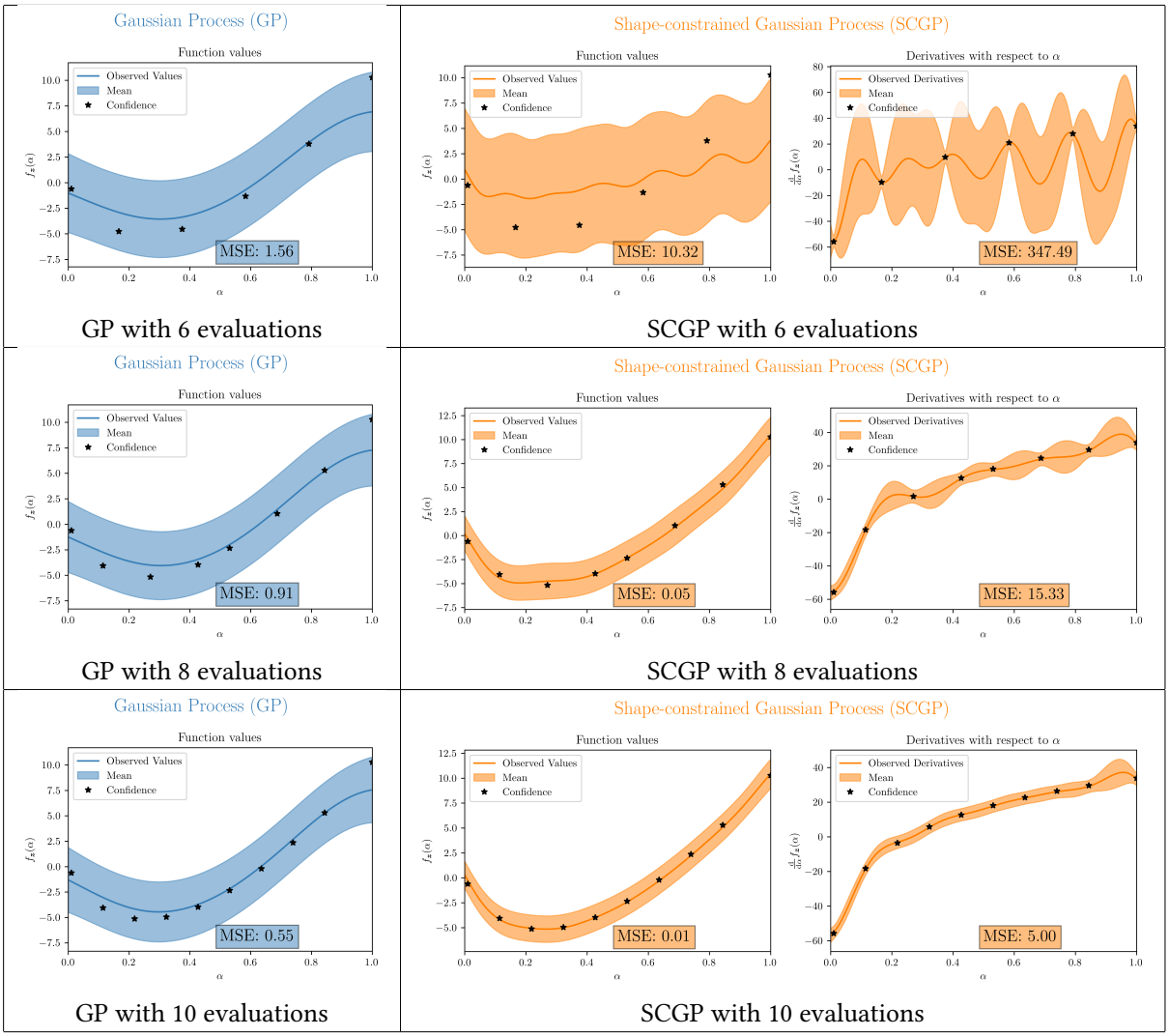

Figure 5: Comparison between Gaussian Process with and without shape constraints under medium curvature and regular grid evaluation (6 to 10 points). Source: author.

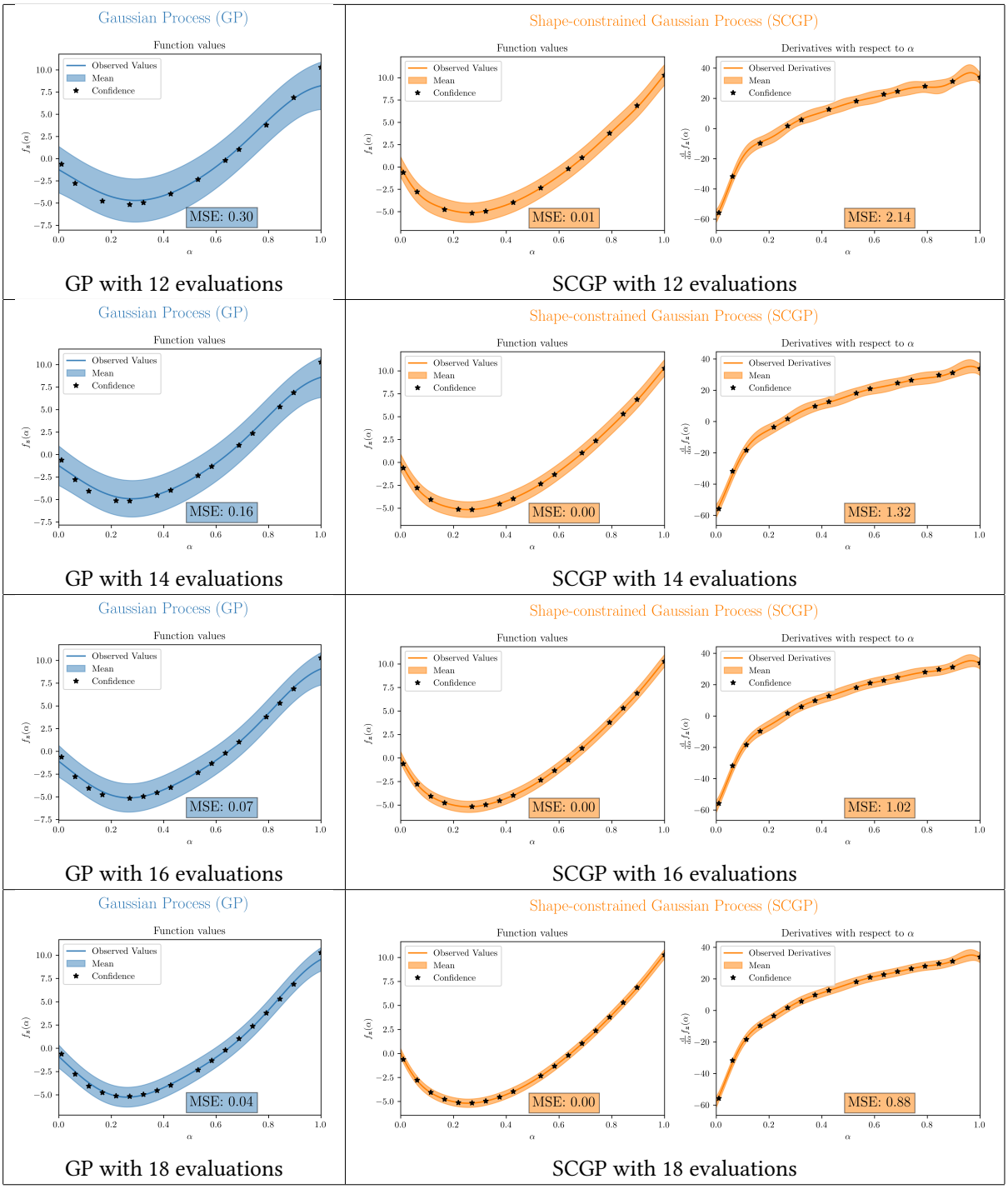

Figure 6: Comparison between Gaussian Process with and without shape constraints under medium curvature and regular grid evaluation (12 to 18 points). Source: author.

4.3 Comparison with SCAM

Shape Constrained Additive Models (SCAM) Pya and Wood (2014) provide an alternative for incorporating shape constraints in function emulation processes. They extend the concept of GAMs (mentioned in section 3.1) to include shape constraints. The SCAM implementation used here was performed in R with the `scam` library Pya (2012).

There is no direct uncertainty quantification in SCAM, which is a disadvantage compared to SCGP. In this case, approximate confidence intervals are used through estimators for the standard deviation of each prediction. Note that these confidence intervals cannot be interpreted as Bayesian guarantees and vice versa.

4.3.1 Variations in Curvature and Position of Observed Values

We compared the SCGP with the SCAM in the same scenarios of high and medium curvature, evaluated on both regular and irregular grids.

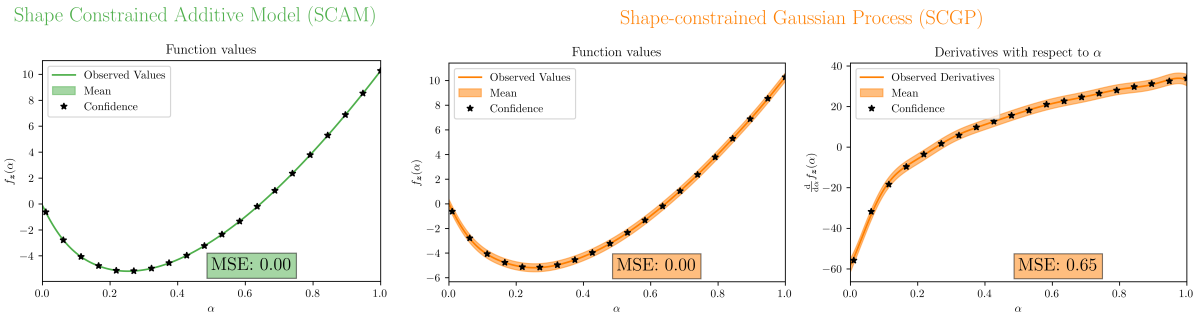

Figure 7: Comparison between SCAM and SCGP under medium curvature and regular grid evaluation. Source: author.

Observing figure 7, we see that both methods performed well in capturing the curvature of the function $f_z(\alpha)$.

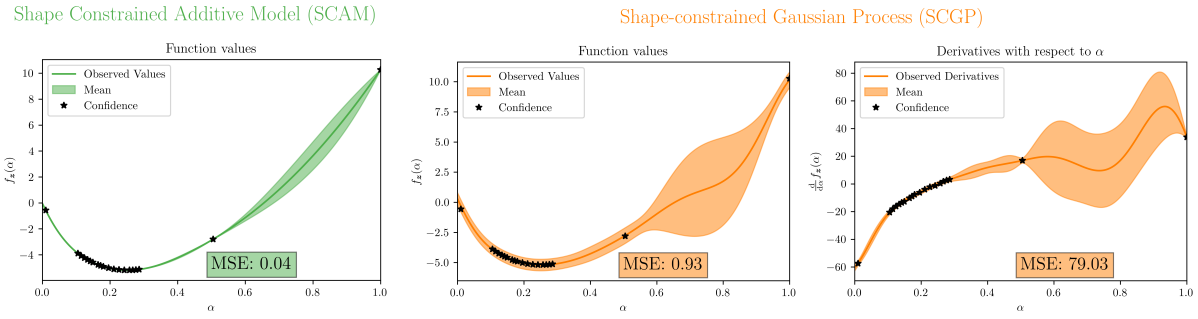

Figure 8: Comparison between SCAM and SCGP under medium curvature and irregular grid evaluation. Source: author.

In the case of irregular grid evaluation, figure 8 shows that SCAM was effective in capturing the curvature of the function $f_z(\alpha)$, whereas the SCGP did not provide a satisfactory fit.

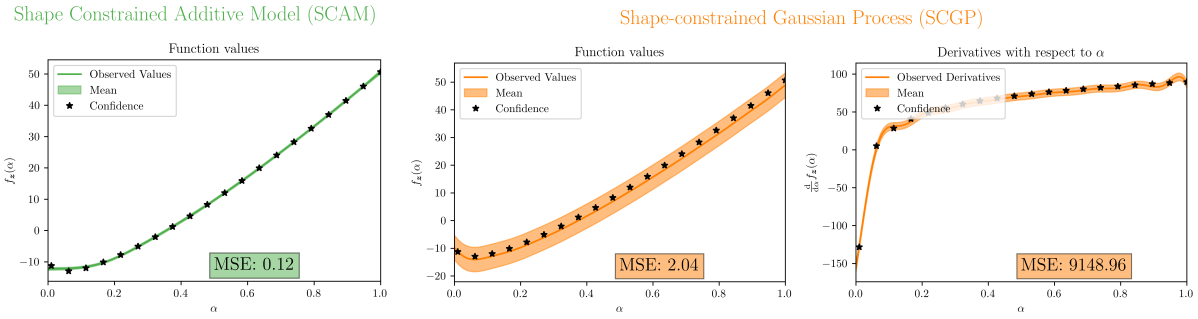

Figure 9: Comparison between SCAM and SCGP under high curvature and regular grid evaluation. Source: author.

In figure 9, although the SCGP manages to adequately recover the target curvature, it is outperformed by the SCAM, which nearly interpolates the function.

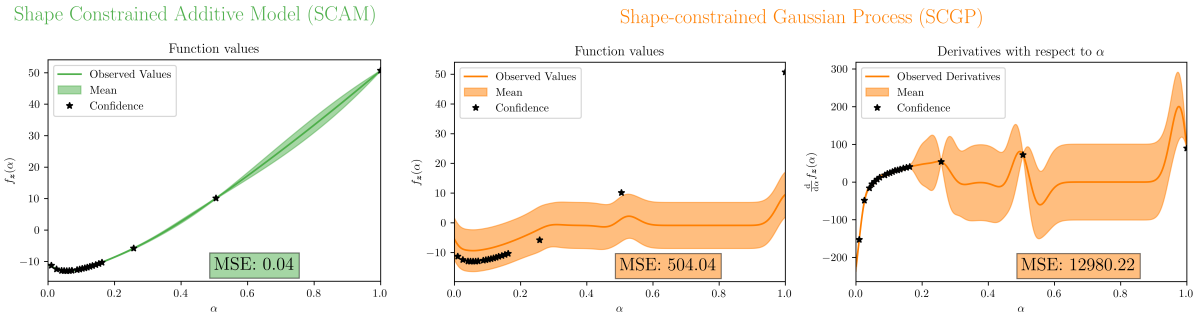

Figure 10: Comparison between SCAM and SCGP under high curvature and irregular grid evaluation. Source: author.

In scenarios where the SCGP fails to capture the curvature of the function $f_z(\alpha)$, the SCAM still provides a satisfactory fit, as observed in figure 10.

4.3.2 Reducing the Number of Observed Points

Our final experiment consists of replicating the experiment from section 4.2.2, but comparing the SCGP with the SCAM. However, there is an interesting detail: the SCAM is unable to fit the function $f_z(\alpha)$ with fewer than 10 observed points, which poses a problem in scenarios where the number of evaluations is limited. Furthermore, as seen in figure 11, the SCAM is able to capture the curvature of the function $f_z(\alpha)$ with 10 observed points, while the SCGP provides an unsatisfactory fit.

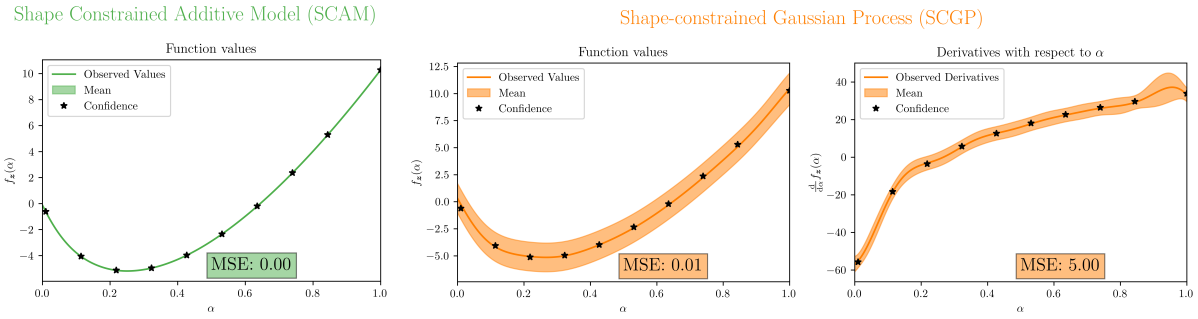

Figure 11: Comparison between SCAM and SCGP under medium curvature and regular grid evaluation with 10 observations. Source: author.

5 Future Work

Through the work developed in this study, we explored the propagation of shape constraints from the derivatives of a function to the function itself, using a regression model based on Gaussian Processes. Gaussian Process theory is a powerful tool for modeling functions and their derivatives, allowing direct quantification of uncertainty and the handling of shape constraints. Moreover, the lack of implementation for shape constraints in Gaussian Processes is a problem that was addressed in this work, proposing a method to incorporate shape constraints in Gaussian Processes, utilizing state-of-the-art tools such as the `Adam` optimization method and the `GPyTorch` library for Gaussian Processes.

The experiments conducted showed that propagating shape constraints from the derivatives of a function to the function itself is effective, at the cost of increased uncertainty, which can be detrimental in high-curvature scenarios with irregular grids. However, in other scenarios, the method proved promising compared to traditional Gaussian Processes, especially considering that derivative evaluations may be available at no additional cost in certain applications. It was observed that the quality of the Gaussian Process fit on the derivatives and the distribution of the function's evaluation points were crucial factors for the propagation of shape constraints to the function itself.

Considering the computational limitations of using Gaussian Processes in large-scale problems, the proposed method may be a viable alternative for emulating functions with shape constraints when only a few evaluations are available. Compared to SCAM, the proposed method stands out for its direct uncertainty quantification, fitting capability without a minimum number of observations, and the ability to handle shape constraints through observations of the function's derivatives. However, SCAM excels in the quality of function fitting, especially in high-curvature scenarios with irregular grids. In the context of function emulation, however, it is plausible to arbitrarily select the points that will be observed to perform the fit, highlighting the SCGP's fitting capability.

As future steps, it is possible to explore fitting through conditional SCGPs (section 2.3.1), which still lacks implementation in the literature, and to develop or adapt metrics to assess the quality of function fitting, thereby minimizing the need for visual evaluation.

Code availability

The code is available at https://github.com/adamesalles/shape-constrained-gaussian-processes.

Acknowledgements

References

Luiz Max Carvalho and Joseph G Ibrahim. On the normalized power prior. *Statistics in Medicine*, 40(24):5251–5275, 2021.

Jamie Fairbrother, Christopher Nemeth, Maxime Rischard, Johanni Brea, and Thomas Pinder. Gaussianprocesses.jl: A nonparametric bayes package for the julia language. *Journal of Statistical Software*, 102(1):1–36, 2022. doi: 10.18637/jss.v102.i01.

J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Blake MacDonald, Pritam Ranjan, and Hugh Chipman. GPfit: An R package for fitting a gaussian process model to deterministic simulator outputs. *Journal of Statistical Software*, 64(12):1–23, 2015. URL http://www.jstatsoft.org/v64/i12/.

K. B. Petersen and M. S. Pedersen. The matrix cookbook, October 2008. Version 20081110.

Natalya Pya. scam: Shape constrained additive models, January 2012. URL http://dx.doi.org/10.32614/CRAN.package.scam.

Natalya Pya and Simon N. Wood. Shape constrained additive models. *Statistics and Computing*, 25(3):543–559, February 2014. ISSN 1573-1375. doi: 10.1007/s11222-013-9448-7. URL http://dx.doi.org/10.1007/s11222-013-9448-7.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2024. URL https://www.R-project.org/.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2005. ISBN 9780262182539.

J. Riihimäki and A. Vehtari. Gaussian processes with monotonicity information. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 645–652, Chia Laguna Resort, Sardinia, Italy, 5 2010. PMLR.

Jaakko Riihimäki and Aki Vehtari. Laplace approximation for logistic gaussian process density estimation and regression, 2013. URL https://arxiv.org/abs/1211.0174.

Christoffer Riis, Francisco Antunes, Frederik Boe Hüttel, Carlos Lima Azevedo, and Francisco Camara Pereira. Bayesian active learning with fully bayesian gaussian processes. *ArXiv*, abs/2205.10186, 2022. URL https://api.semanticscholar.org/CorpusID:248965380.

X. Wang and J. O. Berger. Estimating shape constrained functions using gaussian processes. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):1–25, 2016. doi: 10.1137/140955033.

C.K.I. Williams and D. Barber. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998. doi: 10.1109/34.735807.

A Proofs

A.1 Predictive Posterior Distribution of a GP

Considering a model given by

$$\begin{aligned} Y \mid (\boldsymbol{f}, \boldsymbol{X}) &\sim \mathcal{N}(\boldsymbol{f}, \sigma^2 I), \\ \boldsymbol{f} \mid \boldsymbol{X} &\sim \mathcal{GP}(0, k) \equiv \mathcal{N}(0, k(\boldsymbol{X}, \boldsymbol{X})), \end{aligned}$$

where $\boldsymbol{X} = X_1, \cdots, X_N$, $\boldsymbol{Y} = Y_1, \cdots, Y_N$ e $\boldsymbol{f} = f(X_1), \cdots, f(X_N)$. And so $\tilde{\boldsymbol{X}} = (\boldsymbol{X}_\star, \boldsymbol{X})$ e $\tilde{\boldsymbol{f}} = (f(\boldsymbol{X}_\star), \boldsymbol{f})$ as notation. We now can find $p(\boldsymbol{Y}, \tilde{\boldsymbol{f}} | \tilde{\boldsymbol{X}})$.

It is actually simple, once we notice that $Y_i = f_i + \varepsilon$ and, by consequence, $\mathrm{Cov}(Y_i, f_j) = \mathrm{Cov}(f_i + \varepsilon_i, f_j) = \mathrm{Cov}(f_i, f_j) = k(X_i, X_j)$. Which makes us able to find its joint distribution.

$$(\boldsymbol{Y}, \tilde{\boldsymbol{f}}) \mid \tilde{\boldsymbol{X}} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma^2 I + k(\boldsymbol{X}, \boldsymbol{X}) & k(\boldsymbol{X}, \tilde{\boldsymbol{X}}) \\ k(\tilde{\boldsymbol{X}}, \boldsymbol{X}) & k(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{X}}) \end{bmatrix} \right).$$

We now marginalise $\boldsymbol{f}$, so we can find the joint distribution of $(\boldsymbol{Y}, \boldsymbol{f}_\star)$. Again, using Gaussian distributions closed operations, we get:

$$(\boldsymbol{Y}, \boldsymbol{f}_\star, \boldsymbol{f}) \mid \tilde{\boldsymbol{X}} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma^2 I + k(\boldsymbol{X}, \boldsymbol{X}) & k(\boldsymbol{X}, \boldsymbol{X}_\star) & k(\boldsymbol{X}, \boldsymbol{X}) \\ k(\boldsymbol{X}_\star, \boldsymbol{X}) & k(\boldsymbol{X}_\star, \boldsymbol{X}_\star) & k(\boldsymbol{X}_\star, \boldsymbol{X}) \\ k(\boldsymbol{X}, \boldsymbol{X}) & k(\boldsymbol{X}, \boldsymbol{X}_\star) & k(\boldsymbol{X}, \boldsymbol{X}) \end{bmatrix} \right).$$

And now:

$$(\boldsymbol{Y}, \boldsymbol{f}_\star) \mid \tilde{\boldsymbol{X}} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma^2 I + k(\boldsymbol{X}, \boldsymbol{X}) & k(\boldsymbol{X}, \boldsymbol{X}_\star) \\ k(\boldsymbol{X}_\star, \boldsymbol{X}) & k(\boldsymbol{X}_\star, \boldsymbol{X}_\star). \end{bmatrix} \right)$$

So as to find the distribution of $(\boldsymbol{f}_\star \mid \boldsymbol{Y}, \tilde{\boldsymbol{X}})$, we can do it by noticing that $(\boldsymbol{f}_\star \mid \boldsymbol{Y}, \tilde{\boldsymbol{X}}) \sim \mathcal{N}(\boldsymbol{\mu}_\star, \Sigma_\star)$ and therefore:

$$\begin{aligned} \boldsymbol{\mu}_\star &= k(\boldsymbol{X}_\star, \boldsymbol{X})(k(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 I)^{-1} \boldsymbol{Y} \\ \Sigma_\star &= k(\boldsymbol{X}_\star, \boldsymbol{X}_\star) - k(\boldsymbol{X}_\star, \boldsymbol{X})(k(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 I)^{-1} k(\boldsymbol{X}, \boldsymbol{X}_\star) \end{aligned}$$

All conditioning and marginalizing operations are found on Petersen and Pedersen (2008).

Finishing, finding $\mathbb{E}_{(\boldsymbol{f}_\star | \boldsymbol{Y}, \tilde{\boldsymbol{X}})} \left[p(\boldsymbol{Y}_\star | \boldsymbol{f}_\star, \tilde{\boldsymbol{X}}) \right]$ is simple by remembering that $\boldsymbol{Y}_\star = \boldsymbol{f}_\star + \varepsilon$ and so we get:

$$(\boldsymbol{Y}_\star \mid \boldsymbol{Y}, \tilde{\boldsymbol{X}}) \sim \mathcal{N}(\boldsymbol{\mu}_\star, \Sigma_\star + \sigma^2 I)$$

.

B Algorithms

B.1 Adaptive grid-building

Algoritimo 1 Adaptive grid-building

Require: $m > 0, M > m, v_1, v_2 > 0, J \in \mathbb{N}$.

procedure INITIALISE

$\boldsymbol{x} = \{x_1, \ldots, x_J\}$, $\boldsymbol{z} = \{z_1, \ldots, z_J\}$, $\boldsymbol{z'} = \{z'_1, \ldots, z'_J\}$, and $\boldsymbol{z''} = \{z''_1, \ldots, z''_J\}$.

end procedure

$x_1 \leftarrow m$ and $x_J \leftarrow M$

$z_1 \leftarrow l(m)$ and $z_J \leftarrow l(M)$

$z'_1 \leftarrow l'(m)$ and $z'_J \leftarrow l'(M)$

$z''_1 \leftarrow l''(m)$ and $z''_J \leftarrow l''(M)$

$J \leftarrow J - 2$

if $\operatorname{sgn}(z'_1) = \operatorname{sgn}(z'_J)$ **then**

for $k \leq J$ **do** ▷ Build regular grid

$x_k \leftarrow x_{k-1} + (M - m)/J$;

$z_k \leftarrow l(x_k)$;

$z'_k \leftarrow l'(x_k)$;

$z''_k \leftarrow l''(x_k)$;

end for

else if $\operatorname{sgn}(z'_1) \neq \operatorname{sgn}(z'_J)$ **then** ▷ Adaptive step

procedure INITIALISE

$\boldsymbol{L} = \{L^{(1)}, \ldots, L^{(J)}\}$, $\boldsymbol{U} = \{U^{(1)}, \ldots, U^{(J)}\}$, $\boldsymbol{\delta} = \{\delta^{(1)}, \ldots, \delta^{(J)}\}$;

$L^{(1)} = m, U^{(1)} = M$;

end procedure

for $1 < j$ **do**

$x_j \leftarrow (U^{(j)} + L^{(j)})/2$;

$z_j \leftarrow l(x_j)$;

$z'_j \leftarrow l'(x_j)$;

$z''_j \leftarrow l''(x_j)$;

if $\operatorname{sgn}(z'_k) = \operatorname{sgn}(z'_1)$ **then**

$L^{(j)} \leftarrow z_j$ and $U^{(j)} \leftarrow U^{(k-1)}$;

else $L^{(j)} \leftarrow L^{(j-1)}$ and $U^{(j)} \leftarrow z_j$;

end if

$\delta^{(j)} = |x_j - x_{j-1}|$;

$j \leftarrow j - 1$

if J=0 **then** Stop

else if $\delta^{(j)} < v_1 m$ **then** Stop

else Find $x_i \in \big(\max(0, x_j - v_2 m), \min(x_k + v_2 m, M)\big)$ s. t. $|x_i - x_{i+1}|$ is largest;

$x_j = |x_i - x_{i+1}|/2$;

$z_j \leftarrow l(x_j)$;

$z'_j \leftarrow l'(x_j)$;

$z''_j \leftarrow l''(x_j)$;

end if

end for

end if