GRADIENT-ENHANCED DEEP GAUSSIAN PROCESSES FOR MULTIFIDELITY MODELLING

A PREPRINT

Viv A. Bone
Department of Electrical and Electronic Engineering
The University of Melbourne
Parkville, VIC, 3010
viv.bone@unimelb.edu.au

Chris van der Heide
Department of Electrical and Electronic Engineering
The University of Melbourne
Parkville, VIC, 3010
chris.vanderheide@unimelb.edu.au

Kieran Mackle
School of Mechanical and Mining Engineering
The University of Queensland
St Lucia, QLD, 4072
m.kearney@uq.edu.au

Ingo H. J. Jahn
School of Engineering
The University of Southern Queensland
Springfield, QLD, 4300
ingo.jahn@unisq.edu.au

Peter M. Dower
Department of Electrical and Electronic Engineering
The University of Melbourne
Parkville, VIC, 3010
pdower@unimelb.edu.au

Chris Manzie
Department of Electrical and Electronic Engineering
The University of Melbourne
Parkville, VIC, 3010
pdower@unimelb.edu.au

February 27, 2024

ABSTRACT

Multifidelity models integrate data from multiple sources to produce a single approximator for the underlying process. Dense low-fidelity samples are used to reduce interpolation error, while sparse high-fidelity samples are used to compensate for bias or noise in the low-fidelity samples. Deep Gaussian processes (GPs) are attractive for multifidelity modelling as they are non-parametric, robust to overfitting, perform well for small datasets, and, critically, can capture nonlinear and input-dependent relationships between data of different fidelities. Many datasets naturally contain gradient data, especially when they are generated by computational models that are compatible with automatic differentiation or have adjoint solutions. Principally, this work extends deep GPs to incorporate gradient data. We demonstrate this method on an analytical test problem and a realistic partial differential equation problem, where we predict the aerodynamic coefficients of a hypersonic flight vehicle over a range of flight conditions and geometries. In both examples, the gradient-enhanced deep GP outperforms a gradient-enhanced linear GP model and their non-gradient-enhanced counterparts.

1 Introduction

Across science and engineering, we often seek to model some underlying process from sampled data. This data is often available from several sources; accurate data is usually expensive to obtain, while biased or noisy data is more readily available. Multifidelity methods aim to fuse high- and low-fidelity data to produce a single predictor that is valid across all fidelities [Peherstorfer et al., 2018]. These methods use densely-sampled lower-fidelity data to reduce interpolation error and sparsely-sampled higher-fidelity data to systematically compensate for the bias and noise that corrupt the lower-fidelity data. This approach can significantly increase the accuracy of the surrogate for a given computational budget [Kennedy and O'Hagan, 2000, Perdikaris et al., 2017].

Gaussian processes (GPs) are a family of stochastic processes that are completely specified by their mean and covariance functions Williams and Rasmussen [2006]. Powerful and principled multifidelity models [Le Gratiet and Garnier, 2014, Rokita and Friedmann, 2018, Liu et al., 2018, Forrester et al., 2007] can be constructed by representing the latent function associated with each fidelity level as a realization of GP. GPs are attractive for regression and function approximation problems for several reasons: their probabilistic construction naturally yields uncertainty predictions; they are non-parametric, so their complexity grows with the size of the dataset; they are both highly flexible and robust to overfitting; empirically, they perform well for small datasets; and they can incorporate prior modelling assumptions and gradient information [Williams and Rasmussen, 2006]. These characteristics have led to increasing use of Gaussian process regression (GPR) across many areas of science and engineering [Deisenroth et al., 2013, Ray and Myer, 2019, Deringer et al., 2021, Gelfand and Schliep, 2016, Wu et al., 2014, Lukaczyk, 2015, Brevault et al., 2020].

Many existing GPR-based multifidelty methods [Helterbrand and Cressie, 1994, Goovaerts, 1998] are special cases of the so-called linear model of coregionalization (LMC) [Bourgault and Marcotte, 1991, Alvarez et al., 2012], which assumes that the output for each fidelity is represented by a linear combination of some underlying latent functions. The seminal 'AR1' autoregressive model [Kennedy and O'Hagan, 2000] follows a similar approach, but relates only successive fidelity levels directly using a bias correction term. For both these methods, the global linear relationship between fidelity levels admits solutions governed by a single joint Gaussian process, facilitating exact computation of the posterior predictions and the marginal likelihood [Alvarez et al., 2012, Kennedy and O'Hagan, 2000].

However, the performance of classical multifidelity techniques degrades when the relationship between fidelity levels is not linear or when the correlation between fidelities varies across the input space. The nonlinear autoregressive GP (NARGP) [Perdikaris et al., 2017] and multifidelity deep GP [Cutajar et al., 2019] were developed to address these issues. These methods generalize the AR1 model and assume that the outputs at fidelity level ℓ are predicted by nonlinearly transforming the outputs from fidelity level $\ell - 1$ and adding a correction term that is a function of the original inputs. NARGP further imposes that the training data are nested and that measurements are noise-free, allowing the marginal likelihood to be explicitly evaluated [Perdikaris et al., 2017]. The multifidelity deep GP [Cutajar et al., 2019] leverages recent advances in sparse variational inference to relax these assumptions and construct an approximate non-Gaussian posterior and marginal likelihood [Damianou and Lawrence, 2013, Salimbeni and Deisenroth, 2017]. This structure affords the multifidelity deep GP additional flexibility and generality, but adds computational cost and cedes some theoretical properties of standard GP models.

A further advantage of GPs is that they can naturally incorporate gradient data, which leads to improved predictive accuracy and uncertainty estimates [Bouhlel and Martins, 2019, Han et al., 2013, Lukaczyk, 2015]. Gradient data is particularly common when approximating the output of an expensive high-fidelity computational model from a small set of data. In this setting, gradient data can often be cheaply and conveniently obtained via automatic differentiation tools [Baydin et al., 2018]. Moreover, when considering solutions to discretized systems of differential equations, cheap approximate gradients can instead be obtained with lower memory consumption via adjoint methods [Jameson, 1988, Nadarajah and Jameson, 2000]. Multifidelity data also naturally arises in this context, often corresponding to increasing levels of simulation accuracy [Peherstorfer et al., 2018]. When simulating physical systems, data of different fidelities can be generated by employing models with different levels of simplifying assumptions. For example, in computational fluid dynamics, these fidelity levels might correspond to inviscid simulations, Reynolds averaged Navier Stokes simulations, and large eddy simulations [Versteeg and Malalasekera, 2007] (in order of increasing accuracy). Moreover, when simulating any system of discretized partial differential equations (PDEs), multifidelity data may be generated by utilizing different mesh discretizations [Peherstorfer et al., 2018].

While classical multifidelity GPR techniques (i.e., those that assume linear relationships between fidelity levels) can be routinely extended to incorporate gradient data, this extension is less clear for the modern multifidelity approaches. Gradient data has been incorporated into a nonlinear multifidelity kriging method, but this method assumes a specific form for the bridging function that relates fidelity levels [Han et al., 2013]. This work focuses on extending the multifidelity deep GP, which is a more general method, to incorporate gradient information. We first illustrate this technique on the multifidelity branin function, which is a common analytic test problem [Perdikaris et al., 2017], then we consider a challenging aerospace application, where we predict the aerodynamic coefficients of hypersonic flight vehicle over a range of flight conditions and vehicle geometries. The techniques presented herein also apply to deep GPs trained on only single-fidelity data.

This paper is structured as follows: Sec. 2 provides relevant background on GPR, Sec. 3 presents the gradient-enhanced deep GP, Sec. 4 summarizes the implementation of the GPR models, Sec. 5.1 presents numerical results on an analytical test problem, Sec. 5.2 treats the representative aerospace test problem, and Sec. 6 concludes the paper.

2 Review of Gaussian process regression

This section reviews the elements of GPR theory that are relevant to the construction of our method, including background to GPs, gradient-enhancement, sparse variational inference, and deep GPs. A GP is a stochastic process f on $\mathbb{R}^d$ characterized by the property that given any finite collection of points $x_1, \ldots, x_n \in \mathbb{R}^d$, the vector $[f(x_1), \ldots, f(x_n)]^\intercal$ has a multivariate Gaussian distribution. The mean and covariance functions $m : \mathbb{R}^d \to \mathbb{R}$ and $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ completely specify the GP: if, given $x, x' \in \mathbb{R}^d$, we have $\mathbb{E}f(x) = m(x)$ and $\mathbb{E}(f(x) - m(x))(f(x') - m(x')) = k(x, x')$, that is $(f(x_1), \ldots, f(x_n)) \sim N(m_X, K)$ where $(m_X)_i = m(x_i)$ and $K_{ij} = k(x_i, x_j)$ for $i, j = 1, \ldots, n$, then we say that $f \sim \mathcal{GP}(m, k)$. We assume that the kernel is Gaussian with variance and lengthscale parameters σ^2 and l_i for $i = 1, ..., d$, i.e. $k(x, x') \doteq \sigma^2 \exp\left(\frac{1}{2}\sum_{i=1}^d |x_i - x_i'|^2/l_i\right)$. Other choices of kernel can be used in our construction provided they are C^2.

When using GPs for regression tasks, inference is performed using Bayes' rule. Given a set $(X_i, Y_i)_{i=1}^n$ of independent and identically distributed input-output data pairs, the assumption that each $Y_i = \tilde{f}(X_i) + \varepsilon_i$ for a latent function $\tilde{f}$ with $\varepsilon_i \sim N(0, \sigma_n)$ induces a Gaussian likelihood. That is, $p(Y \mid f, X) = (2\pi\sigma_n^2)^{-n/2} \exp(-\frac{1}{2\sigma^2}\|Y - \tilde{f}(X)\|^2)$, with σ_n^2 capturing model and output noise. While the case where $m \equiv 0$ is most commonly considered, the hierarchical structure of both multifidelity models and deep GPs admit natural prior means. The covariance structure is encoded in the gram matrix K, which has elements $K_{ij} = k(X_i, X_j)$. When considering a collection of N new input points $X^\star = \{x_1^\star, \ldots, x_N^\star\}$, we write $f(X^\star) = (f(x_1^\star), \ldots, f(x_N^\star))$ and denote by $K^{\star\star} \in \mathbb{R}^{N\times N}$ and $k^\star \in \mathbb{R}^{n\times N}$ the matrices with entries $K_{ij}^{\star\star} = k(x_i^\star, x_j^\star)$ and $K_{ij}^\star = k(X_i, x_j^\star)$. In this construction, our choice of GP prior $f \sim \mathcal{GP}(m, k)$ conditioned on the data specifies a posterior distribution $p(f \mid X, Y)$ that satisfies Williams and Rasmussen [2006] $f(X^\star) \mid X, Y \sim \mathcal{N}(f^\star(X^\star), \Sigma(X^\star))$, where

$$\begin{aligned} f^\star(X^\star) &= m_{X^\star} - K^{\star\top}(K + \sigma_n^2 I)^{-1}(Y - m_X), \quad \text{for} \quad (m_{X^\star})_i \doteq m(x_i^\star) \\ \Sigma(X^\star) &= K^{\star\star} - K^{\star\top}(K + \sigma_n^2 I)^{-1}K^\star \end{aligned} \tag{2.1}$$

and Y denotes the vector with i-th component Y_i. The model hyperparameters θ, which include kernel parameters l_1, ..., l_d and noise variance σ_n^2, are then typically optimized in an empirical Bayes procedure to (locally) maximize the log-marginal likelihood

$$L(Y \mid X, \theta) = -Y^\top(K + \sigma_n^2 I)^{-1}Y - \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{n}{2}\log(2\pi), \tag{2.2}$$

which is equivalent to the free energy principle. This Bayesian approach inherently balances model complexity and prediction accuracy; the first term in (2.2) penalizes prediction error while the second penalizes model complexity.

2.1 Gradient enhancement

Many datasets in the physical sciences and engineering, especially when generated with computational models, contain gradient information in addition to output values. This information can be incorporated into GP models to improve their prediction accuracy and uncertainty estimates [Williams and Rasmussen, 2006]. We consider datasets where the output samples are augmented with gradients — i.e., $(Y_\nabla)_i = [Y_i, \nabla^\intercal Y_i]^\intercal$ — where ∇Y denotes the d-dimensional sample of gradient information. We assume independent noise on each derivative measurement [Lukaczyk, 2015], so each $(Y_\nabla)_i \in \mathbb{R}^{1+d}$ is a sample from

$$\mathcal{N}\left(\begin{bmatrix}\tilde{f}(X_i) \\ \nabla\tilde{f}(X_i)\end{bmatrix}, \text{diag}\left(\sigma_1^2, ..., \sigma_{d+1}^2\right)\right). \tag{2.3}$$

Since the kernel is assumed to be smooth, joint predictions for f and ∇f can be generated using using the gradient kernel $k_\nabla : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{(d+1)\times(d+1)}$

$$k_\nabla(x_p, x_q) = \begin{bmatrix} k(x_p, x_q) & \nabla_q^\intercal k(x_p, x_q) \\ \nabla_p k(x_p, x_q) & \nabla_p\nabla_q^\intercal k(x_p, x_q)\end{bmatrix}, \tag{2.4}$$

for all $x_p, x_q \in \mathbb{R}^d$, where ∇_p (∇_q) denotes the derivative taken with respect to x_p (x_q), and $k_\nabla(x_p, x_q)$ is the covariance of $f_\nabla(x_p)$ and $f_\nabla(x_q)$ for $f_\nabla \doteq [f, \nabla^\top f]^\top$. The mean vector of f_∇ is denoted $m_\nabla \doteq [m, \nabla^\top m]^\top$. The form of (2.4) follows from linearity of differentiation and is positive definite for stationary kernels – for details see [Williams and Rasmussen, 2006]. The gradient kernel can be used to generate predictions in exactly the same way as the gradient-free case, with each entry of K, $K^\star$ and $K^{\star\star}$ being replaced by blocks of the form given in (2.4). Samples from $f_\nabla \mid Y_\nabla, X$ now correspond to selecting only functions that (approximately) pass through the data at the observed gradients. This process improves the predictions generated by the GP, particularly for sparse training data.

2.2 The linear model of coregionalization

Multifidelity GPR methods are data fusion techniques that approximate a family of related latent functions $\tilde{f}^1$, ..., $\tilde{f}^L$ from iid samples that contain information from each of the L fidelities, $(X_i^\ell, Y_i^\ell)_{\ell,i=1}^{L,n^\ell}$, where n^ℓ is the number of datapoints for fidelity ℓ. Many classical multifidelity techniques are special cases of the LMC scheme, which models each output as a linear combination of some underlying latent functions [Bourgault and Marcotte, 1991]. This assumption induces a product kernel which is separable in its vector-valued point-wise arguments x_p, x_q and integer-valued layer-wise indices i, j, which represent model fidelities:

$$\text{cov}(f^i(x_p),\, f^j(x_p)) = k(x_p,\, x_q) \cdot k_{\text{I}}(i,\, j) \tag{2.5}$$

where $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a standard (or gradient-enhanced) GP kernel and $k_{\text{I}} : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ is an index kernel. The kernel matrix K_I corresponding to the kernel k_{I} is positive semidefinite, typically parameterized as a Cholesky decomposition, i.e., $k_{\text{I}}(i,\, j) = [BB^\intercal]_{ij}$, where $B \in \mathbb{R}^{L\times L}$ is an upper triangular matrix, and the i, j-th entry gives the scaling of the shared kernel between model fidelities i and j. LMC can be extended to consider a sum of T separable kernels, each with different hyperparameters:

$$\text{cov}(f^i(x_p),\, f^j(x_p)) = \sum_{t=1}^{T} \left(k_t(x_p,\, x_q) \cdot k_{\text{I}t}(i,\, j)\right). \tag{2.6}$$

The common 'AR1' approach [Kennedy and O'Hagan, 2000] can be shown to be a special case of LMC with number of separable kernels set to the number of fidelities (i.e., $T = L$).

LMC models all the latent functions, each of which corresponds to a different fidelity level, as a single multi-output GP $f_{MF} = [f^1,\, ...,\, f^L]^\top$. For a finite collections of points, the gram matrix is

$$K_{LMC} = \begin{bmatrix} K_{11} & ... & K_{1L} \\ \vdots & \ddots & \vdots \\ K_{L1} & ... & K_{LL} \end{bmatrix}, \tag{2.7}$$

where each submatrix K_{ij} is the kernel matrix evaluated with the covariance function in (2.6) for input sets X^i and X^j and indices i and j for $i, j \in 1, ..., L$. We remark that in the case where training data for all fidelity levels contain every input point, this can be computed as the Kronecker product $K_I \otimes K$, where $(K_I)_{i,j} = k_I(i, j)$ and K is the gram matrix with entries $K_{p,q} = k(X_p, X_q)$. LMC can naturally be extended to consider gradient information by using gradient kernels from (2.4) for each k_t. Assuming equal measurement noise σ_n^2 for each fidelity level, the log-marginal likelihood (2.2) and posterior predictive mean and covariance terms (2.1) can be evaluated in the same way as the single-fidelity case.

2.3 Multifidelity deep Gaussian processes

An emerging technique that can capture nonlinear relationships between different model fidelities is the multifidelity deep GP [Cutajar et al., 2019]. The deep GP defines the prior recursively, with the inputs to layer ℓ being the outputs from layer $\ell - 1$, augmented with the base-layer input [Damianou and Lawrence, 2013]. Under this construction, layer one is a standard GP with deterministic inputs, then from the second layer onwards, we assume the model has the following autoregressive form [Cutajar et al., 2019]:

$$f^\ell(x) = g^\ell(f^{\ell-1}(x),\, x) + \gamma^\ell(x), \tag{2.8}$$

where $g^\ell : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ and $\gamma^\ell : \mathbb{R}^d \to \mathbb{R}$. To represent this model as a GP, we assume that the mapping g^ℓ decomposes as [Cutajar et al., 2019]

$$f^\ell(x) = g_f^\ell(f^{\ell-1}(x)) \cdot g_x^\ell(x) + \gamma^\ell(x). \tag{2.9}$$

Note that (2.9) can be represented by combining three GPs with mean functions $m_{gx}^\ell : \mathbb{R}^d \to \mathbb{R}$, $m_{gf}^\ell : \mathbb{R} \to \mathbb{R}$, $m_{\gamma x}^\ell : \mathbb{R}^d \to \mathbb{R}$ and covariance functions $k_{gx}^\ell : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, $k_{gf}^\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $k_{\gamma x}^\ell : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. The compositional structure of the deep GP results in a model that is not a GP itself, since its values evaluated at finitely many points cannot be described by a multivariate normal distribution [Damianou and Lawrence, 2013].

We set g_x^ℓ to be everywhere one and assume a zero mean prior for $m_{\gamma x}^\ell$, reducing the mean function $m^\ell : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ to $m^\ell(f^{l-1}(x),\, x) = m_{gf}^\ell(f^{l-1}(x)) = \mathbb{E}f^\ell(x)$. We then choose an affine mean for m_{gf}^ℓ, i.e.,

$$m^\ell(f^{l-1}(x),\, x) = m_{gf}^\ell(f^{\ell-1}(x)) \doteq \kappa f^{\ell-1}(x) + c, \tag{2.10}$$

with the learned parameters κ and c initialzed to unity and zero respectively. That is, as a prior, we assume the outputs for fidelity level ℓ are equal to those for fidelity level $\ell - 1$. The layer-wise covariance functions are of the form

$$\begin{aligned}\operatorname{cov}(f^\ell(x_p),\, f^\ell(x_q)) &= \mathbb{E}[(f^\ell(x_p) - m^\ell(f^{\ell-1}(x_p),\, x_p))(f^\ell(x_q) - m^\ell(f^{\ell-1}(x_q),\, x_q))] \\ &\doteq k^\ell((f_p^{\ell-1},\, x_p),\, (f_q^{\ell-1},\, x_q)),\end{aligned} \tag{2.11}$$

where we write $f_p^{\ell-1} = f^{\ell-1}(x_p)$, $f_q^{\ell-1} = f^{\ell-1}(x_q)$, and $k^\ell : \mathbb{R} \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$. Following [Salimbeni and Deisenroth, 2017], in each layer's covariance function, we include internal noise (which propagates through all subsequent layers), giving

$$k^\ell((f_p^{\ell-1},\, x_p),\, (f_q^{\ell-1},\, x_q)) = k_{gx}^\ell(x_p,\, x_q) \cdot k_{gf}^\ell(f_p^{\ell-1},\, f_q^{\ell-1}) + k_{\gamma x}^\ell(x_p,\, x_q) + \sigma_k^2\, \delta_{p,q} \tag{2.12}$$

where σ_k^2 is the variance of the kernel noise, and $\delta_{p,q}$ is the Kronecker delta. We write $X^\ell = (X_i^\ell)_{i=1}^n$ and fix the first-layer inputs as the union of all input data $X^0 = \bigcup_{\ell=1}^L X^\ell$. In contrast to the NARGP [Perdikaris et al., 2017], this approach does not require training data to be nested across fidelity levels ($X^1 \supseteq X^2 \supseteq ... \supseteq X^L$).

We have seen that as observed in Damianou and Lawrence [2013], the compositional nature of deep GPs results in a stochastic process that is not itself a GP. Since the prior is no longer Gaussian, the posterior distribution and marginal likelihood are no longer analytically or computationally tractable. However, in order to perform inference using these models, variational approximations can be used.

2.4 Variational inference

The fundamental challenge when fitting deep GPs is the evaluation of the marginal likelihood and posterior distribution. Direct numerical approximation of the marginal likelihood via Monte Carlo sampling is only tractable for extremely small problems. However, deep GPs can be made computationally tractable using variational inference (VI) techniques [Damianou and Lawrence, 2013, Salimbeni and Deisenroth, 2017], which were previously used to scale GP models to large datasets [Titsias and Lawrence, 2010]. These so-called sparse VI methods mitigate the $\mathcal{O}(n^3)$ computational bottleneck of inference by circumventing the need to invert the full gram matrix and by enabling subsampling.

Sparse VI techniques introduce a family of *inducing inputs* $Z = \{z_i\}_{i=1}^m$ with corresponding inducing points $U = \{u\}_{i=1}^m$ with each $z_i \in \mathbb{R}^d$ and $u_i \in \mathbb{R}$ assumed to satisfy $u_i = f(z_i)$. The joint density $p(f, U)$ is then a Gaussian with a common prior mean and covariance function. Thus, the joint distribution of Y, f and U decompose into the prior and likelihood, giving

$$p(Y, f, U) = p(f \,|\, U; X, Z)p(U; Z) \prod_{i=1}^n p(Y_i \,|\, f_i), \tag{2.13}$$

where we have factored the joint prior $p(Y, f, U \,|\, X, Z)$ as $U \sim \mathcal{N}(m_Z, K_{ZZ})$ and conditional likelihood $f \,|\, U \sim \mathcal{N}(\tilde{\mu}, \tilde{\Sigma})$. Here, the mean and covariance of U satisfy $(m_Z)_i \doteq m(z_i)$ and $(K_{ZZ})_{ij} \doteq k(z_i, z_j)$. The conditional mean and covariance $\tilde{\mu}$ and $\tilde{\Sigma}$ are given by

$$\tilde{\mu}_i \doteq m(X_i) + \alpha(X_i)^\top (U - m_Z) \qquad \text{and} \qquad \tilde{\Sigma}_{ij} \doteq k(X_i, X_j) - \alpha(X_i)^\top K_{ZZ} \alpha(X_j),$$

for $i, j = 1, \ldots, n$ and $\alpha(X_i) \doteq K_{ZZ}^{-1} K_{ZX_i}$, where $(K_{ZX_i})_j = k(z_j, X_i)$.

For training, a lower bound on the marginal likelihood (the evidence lower bound, or 'ELBO') is introduced as a surrogate cost in lieu of computing the log-marginal likelihood. This cost is derived by introducing a free-form variational posterior

$$q(f, U) \approx p(f, U \,|\, Y). \tag{2.14}$$

Writing the marginal likelihood as $p(Y) = \frac{p(f,U,Y)}{p(f,U \,|\, Y)}$ (where dependence on X and U is suppressed), then taking the log and expectation over $q(f, U)$, gives

$$\begin{aligned}\log p(Y) &= \mathbb{E}_{q(f,U)} \log\left(\frac{p(f, U, Y)}{q(f, U)}\right) + \mathbb{E}_{q(f,U)} \log\left(\frac{q(f, U)}{p(f, U \,|\, Y)}\right) \\ &\doteq \mathcal{L}_{\mathrm{ELBO}}\left(q(f, U)\right) + \mathcal{D}_{\mathrm{KL}}\left(q(f, U) \,\|\, p(f, U \,|\, Y)\right) \geq \mathcal{L}_{\mathrm{ELBO}}\left(q(f, U)\right)\end{aligned}$$

where $\mathcal{D}_{\mathrm{KL}}$ denotes the (non-negative) Kullback-Leibler divergence (KL) from $q(f, U)$ to $p(f, U \,|\, Y)$. Exploiting (2.13), the ELBO can be decomposed and marginalized to obtain

$$\begin{aligned}\mathcal{L}_{\mathrm{ELBO}}\left(q(f, U)\right) &= \mathbb{E}_{q(f,U)} \log\left(\frac{p(f, U, Y)}{q(f, U)}\right) = \mathbb{E}_{q(f,U)} \log\left(\frac{p(Y \,|\, f, U)p(f \,|\, U)p(U)}{p(f \,|\, U)q(U)}\right) \\ &= \mathbb{E}_{q(f)} \log p(Y \,|\, f) - \mathcal{D}_{\mathrm{KL}}\left(q(U) \,\|\, p(U)\right).\end{aligned} \tag{2.15}$$

For analytic and computational tractability, we set the form of the variational posterior to be $q(f, U) = p(f \mid U)q(U)$, where $q(U) = \mathcal{N}(m_q, S_q)$ is a Gaussian with variational parameters $m_q \in \mathbb{R}^m$ and $S_q \in \mathbb{R}^{m\times m}$ [Salimbeni and Deisenroth, 2017]. Maximization of the ELBO can be viewed as maximization of the expected marginal likelihood under the choice of variational posterior, penalized by the KL divergence $q(U)$ to the true prior mean. As the variational posterior is a product of two Gaussians, U can be marginalized out analytically to obtain the Gaussian predictive distribution $q(f) = \mathcal{N}(\mu, \Sigma)$ with mean and covariance given by

$$\mu_i \doteq m(X_i) + \alpha(X_i)^\top (m_q - m_Z) \qquad \text{and} \qquad \Sigma_{ij} \doteq k(X_i, X_j) - \alpha(X_i)^\top \left(K_{ZZ} - S_\phi\right) \alpha(X_j).$$

Inducing point methods can also be used to define a variational posterior for deep GPs by introducing inducing inputs $Z^\ell = \left\{z_i^\ell\right\}_{i=1}^{m_\ell}$ and inducing variables $\{u^\ell\}_{i=1}^{m_\ell}$ for each layer. The inducing inputs correspond to each layer's inputs, so for the first layer each $z_i^1 \in \mathbb{R}^d$, and for subsequent layers, the inducing inputs z_i^ℓ match the previous layer's output dimension. In the multifidelity setting, we consider single-width hidden layers, so each $z_i^\ell \in \mathbb{R}$ for $l \geq 2$ has a corresponding base-layer inducing input z_i^1. For moderately sized datasets, we choose a full-rank variational posterior where $m_\ell = n_\ell$. In this case, we fix the base-layer inducing inputs at the input data $Z^1 = X^0$ and set the remaining inducing inputs as the output data for the previous layer, so $z_i^\ell = \tilde{y}_i^{\ell-1} \approx \tilde{f}^{\ell-1}(x_i)$ for $l = 2, ..., L$ [Cutajar et al., 2019]. For larger datasets we set $m_\ell \leq n_\ell$ and including the inducing inputs as free variational parameters. To construct an analogous ELBO for a deep GP with L layers, we suppress dependence on data and intermediate layers' internal dimensions to decompose the joint density as the likelihood and prior

$$p(Y, \{f^\ell, U^\ell\}_{\ell=1}^L) \doteq \prod_{i=1}^n p(Y_i \mid f_i^L) \prod_{\ell=1}^L p(f^\ell \mid U^\ell; f^{\ell-1}, Z^\ell) p(U^\ell; Z^\ell),$$

which can be compared with (2.13).

Doubly stochastic variational inference can be used to compute approximate deep Gaussian process posteriors that maintain correlations both within and between layers [Salimbeni and Deisenroth, 2017]. Under this setup, the variational distribution of U^ℓ factorizes between layers as a Gaussian with mean m_ϕ^ℓ and covariance S_ϕ^ℓ, admitting a variational posterior of the form

$$q\Big(\left\{f^\ell, U^\ell\right\}_{\ell=1}^L\Big) \doteq \prod_{\ell=1}^L p\big(f^\ell \mid U^\ell; f^{\ell-1}, Z^\ell\big) q\big(U^\ell\big). \tag{2.16}$$

Since the distributions are Gaussian, the inducing variables can be marginalized out to obtain a Gaussian predictive distribution

$$q(\{f^\ell\}_{\ell=1}^L) \doteq \prod_{\ell=1}^L q(f^\ell \mid m_\phi^\ell, S_\phi^\ell; f^{\ell-1}, Z^\ell) = \prod_{\ell=1}^L \mathcal{N}(\mu^\ell, \Sigma^\ell), \tag{2.17}$$

whose mean vectors and covariance matrices have entries

$$\begin{aligned} \mu_i^\ell &\doteq m^\ell(f_i^{l-1}, X_i) + \alpha^\ell(f_i^{\ell-1}, X_i)^\top (m_q^\ell - m_Z^\ell) \\ \Sigma_{ij}^\ell &\doteq k^\ell(f_i^{\ell-1}, f_j^{\ell-1}) - \alpha^\ell(f_i^{\ell-1}, X_i)^\top \big(K_{ZZ}^\ell - S_q\big)\alpha^\ell(f_j^{\ell-1}, X_j), \end{aligned} \tag{2.18}$$

where $(m_Z^\ell)_i = m^\ell(Z_i^\ell, Z_i^1)$ for the mean function (2.10), $(K_{ZZ}^\ell)_{ij} = k^\ell((Z_i^\ell, Z_i^1), (Z_j^\ell, Z_j^1))$ for the covariance function (2.11), and $\alpha^\ell(f_i^{\ell-1}, x_i) \doteq (K_{ZZ}^\ell)^{-1} K_{ZX_i}^\ell$ with $(K_{ZX_i}^\ell)_j = k^\ell((Z_j^\ell, Z_j^1), (f_i^{\ell-1}, x_i))$. The i-th marginal of any layer M of the posterior depends only on the i-th marginal of the previous layers, that is,

$$q(f_i^M) = \int \prod_{\ell=1}^{M-1} q(f_i^\ell \mid m_\phi^\ell, S_\phi^\ell; f_i^{\ell-1}, Z^\ell)\, df_i^\ell, \tag{2.19}$$

which enables tractable computation of the posterior. The integral appearing in (2.19) can be approximated by first sampling $\varepsilon_i^\ell \sim \mathcal{N}(0, I_{d^\ell})$, then recursively drawing

$$\hat{f}_i^\ell \sim q(f_i^\ell \mid m^\ell, S^\ell; f_i^{\ell-1}, Z^\ell) \tag{2.20}$$

for $\ell = 1, \ldots, L-1$ via $\hat{f}_i^\ell = \mu_i^\ell + \varepsilon_i^\ell \odot \sqrt{\Sigma_{ii}^\ell}$, where d^ℓ is the output dimension of the ℓ-th layer, and $\hat{f}_i^0 = X_i$. This process is equivalent to representing each layer's posterior as a mixture of N_s Gaussian distributions $\{\hat{q}^j(f^\ell)\}_{j=1}^{N_s}$,

where each distribution corresponds to a sample $(\hat{f}^{\ell-1})^j = \{(\hat{f}_i^{\ell-1})^j\}_{i=1}^{n^l}$ drawn from the previous layer's corresponding output distribution, i.e.,

$$\hat{q}^j(f^\ell) = q(f_i^\ell \,|\, m_\phi^\ell, S_\phi^\ell; \hat{f}_i^{\ell-1}, Z^\ell). \tag{2.21}$$

Thus, the posterior at layer ℓ is approximated by the Gaussian mixture

$$q(f^\ell) \approx \hat{q}(f^\ell) \doteq \frac{1}{N_s}\sum_{j=1}^{N_s} \hat{q}^j(f^\ell), \tag{2.22}$$

where samples can be drawn from $\hat{q}^j(f_i^\ell)$ for $i = 1, ..., n^\ell$ independently. We denote the full set of samples over all layers as $\hat{f} \doteq \{(\hat{f}^\ell)^j\}_{\ell,j=1}^{L,N_s}$ and the corresponding approximate posteriors as $\hat{q} \doteq \{\hat{q}(f^\ell)\}_{\ell=1}^L$.

The ELBO for the deep GP under doubly stochastic variational inference can be computed analogously to the single-layer case (2.15). This work focuses on multifidelity modelling, so we evaluate the ELBO over all layers [Cutajar et al., 2019], giving

$$\begin{aligned}\mathcal{L}_{\text{DGP}-\text{ELBO}}\left(q(f,u)\right) &= \mathbb{E}_{q(\{f^\ell, U^\ell\}_{\ell=1}^L)} \log\left(\frac{p\left(Y, \{f^\ell, U^\ell\}_{\ell=1}^L\right)}{q\left(\{f^\ell, U^\ell\}_{\ell=1}^L\right)}\right) \\ &= \sum_{\ell=1}^{L}\sum_{i=1}^{n^\ell} \mathbb{E}_{q(f_i^\ell)} \log p(Y_i \,|\, f_i^\ell) - \beta \sum_{\ell=1}^{L} \mathcal{D}_{\text{KL}}\left(\phi(U^\ell) \,\|\, p(U^\ell)\right),\end{aligned} \tag{2.23}$$

where a β scaling term has been introduced to modulate the regularizing effect of the KL divergence term. The expectations in (2.23) can be approximated using the samples over all layers $\hat{f}$.

More recently, an alternative training objective that restores full symmetry between the objective itself and the predictive posterior has been introduced [Jankowiak et al., 2020b]. This objective, known as the predictive log likelihood (PLL), is obtained by directly minimizing the KL divergence from the empirical output distribution to the predictive distribution, then adding the KL divergence regularizing term from the ELBO [Jankowiak et al., 2020b]:

$$\begin{aligned}\mathcal{L}_{\text{DGP}-\text{PLL}} &= \mathbb{E}_{p_{\text{data}}\left(\{Y^l\}_{\ell=1}^L, X\right)}\left(\log q(\{Y^\ell\}_{\ell=1}^L \,|\, X)\right) - \beta \sum_{\ell=1}^{L} \mathcal{D}_{\text{KL}}\left(q(U^\ell) \parallel p(U^\ell)\right), \\ &= \sum_{\ell=1}^{L}\sum_{i=1}^{n_\ell}\left(\log q(Y^\ell \,|\, X_i)\right) - \beta \sum_{\ell=1}^{L} \mathcal{D}_{\text{KL}}\left(q(U^\ell) \parallel p(U^\ell)\right).\end{aligned} \tag{2.24}$$

This objective gives good performance for single-layer GPs and can also be applied to deep GPs [Jankowiak et al., 2020a]. However, for deep GPs, the expectation over the latent function values occurs inside the log, so approximation of $\mathcal{L}_{\text{DGP}-\text{PLL}}$ using finite samples from $q(f^{l-1} \,|\, x)$ results in a biased estimator. Recent work developed the deep sigma point process to address this issue [Jankowiak et al., 2020a], but here we found that approximating the expectations with sufficiently many samples gives adequate performance.

3 Gradient-enhanced deep GP

In view of the methods introduced in Sec. 2, we construct our method for extending deep GPs to incorporate gradient data. Deep GPs can be readily extended to incorporate gradient information by predicting and conditioning on f_∇^ℓ each layer, where

$$f_\nabla^\ell = [f^\ell, \nabla^\top f^\ell]^\top \in \mathbb{R}^{d+1} \tag{3.1}$$

and ∇f^ℓ is the gradient of f^ℓ with respect to the base inputs x. Gradients can be predicted by modifying both the deep GP kernel function and variational posterior and passing the function values and gradients (3.1) through each layer. The first layer takes only the base inputs x and thus uses a standard gradient-enhanced kernel (2.4). For the subsequent layers $\ell \geq 2$, the corresponding gradient kernels $k_\nabla^\ell : \mathbb{R}^{d+1} \times \mathbb{R}^d \times \mathbb{R}^{d+1} \times \mathbb{R}^d \to \mathbb{R}^{(d+1)\times(d+1)}$ satisfy

$$k_\nabla^\ell\left((f_{\nabla p}^{\ell-1}, x_p), (f_{\nabla q}^{\ell-1}, x_q)\right) = \begin{bmatrix} k^\ell\left((f_p^{\ell-1}, x_p), (f_q^{\ell-1}, x_q)\right) & \nabla_q^\top k^\ell\left((f_p^{\ell-1}, x_p), (f_q^{\ell-1}, x_q)\right) \\ \nabla_p k^\ell\left((f_p^{\ell-1}, x_p), (f_q^{\ell-1}, x_q)\right) & \nabla_p \nabla_q^\top k^\ell\left((f_p^{\ell-1}, x_p), (f_q^{\ell-1}, x_q)\right) \end{bmatrix}. \tag{3.2}$$

Due to the separable structure of the kernel (2.12), the gradient terms can be explicitly computed as

$$\begin{aligned}\nabla_p k^\ell\big((f^{\ell-1}_{\nabla p},\, x_p),\,(f^{\ell-1}_{\nabla q},\, x_q)\big) &= \nabla_p\left(k^\ell_{gx}(x_p,\, x_q)\cdot k^\ell_{gf}(f^{\ell-1}_p,\, f^{\ell-1}_q)\right) + \nabla_p\, k^\ell_{\gamma x}(x_p,\, x_q) \\ &= \nabla_p k^\ell_{gx}(x_p,\, x_q)\cdot k^\ell_{gf}(f^{\ell-1}_p,\, f^{\ell-1}_q) + k^\ell_{gx}(x_p,\, x_q)\cdot \nabla_p k^\ell_{gf}(f^{\ell-1}_p,\, f^{\ell-1}_q) \\ &\quad + \nabla_p\, k^\ell_{\gamma x}(x_p,\, x_q).\end{aligned} \tag{3.3}$$

Explicitly accounting for the functional dependence of $f^{\ell-1}_p$ on x_p — i.e., $f^{\ell-1}_p = f^{\ell-1}(x_p)$ — using the chain rule gives

$$\nabla_p k^\ell_{gf}(f^{\ell-1}_p,\, f^{\ell-1}_q) = \nabla_{f^{\ell-1}_p} k^\ell_{gf}(f^{\ell-1}_p,\, f^{\ell-1}_q)\cdot \nabla f^{\ell-1}(x_p). \tag{3.4}$$

By denoting gradient kernels with respect to their inputs as $k^\ell_{gx\nabla}$, $k^\ell_{gf\nabla}$, and $k^\ell_{\gamma\nabla}$ and defining matrices

$$\hat{F}^\ell_{\nabla p} = \begin{bmatrix}1 & 0\\ \mathbf{0} & \nabla f^{\ell-1}(x_p)\end{bmatrix} \qquad \text{and} \qquad \hat{F}^\ell_{\nabla q} = \begin{bmatrix}1 & 0\\ \mathbf{0} & \nabla f^{\ell-1}(x_q)\end{bmatrix}, \tag{3.5}$$

(3.2) becomes

$$\begin{aligned}k^\ell_\nabla\big((f^{\ell-1}_{\nabla p},\, x_p),\,(f^{\ell-1}_{\nabla q},\, x_q)\big) &= k^\ell_{gx\nabla}(x_p,\, x_q)\cdot k^\ell_{gf}(f^{\ell-1}_p,\, f^{\ell-1}_q) \\ &\quad + k^\ell_{gx}(x_p,\, x_q)\cdot \hat{F}^\ell_{\nabla p}\, k^\ell_{gf\nabla}(f^{\ell-1}_p,\, f^{\ell-1}_q)\,(\hat{F}^\ell_{\nabla q})^\top \\ &\quad + k^\ell_{\gamma x\nabla}(x_p,\, x_q).\end{aligned} \tag{3.6}$$

The choice of affine mean used in the deep GP (2.10) means the gradient-enhanced mean function $m^l_\nabla : \mathbb{R}^{d+1}\times\mathbb{R}^d \to \mathbb{R}^{d+1}$ is simply

$$m^\ell_\nabla(f^{\ell-1}_\nabla(x),\, x) = \begin{bmatrix}\kappa f^{\ell-1}(x) + c\\ \kappa\nabla f^{\ell-1}(x)\end{bmatrix}. \tag{3.7}$$

We now define a variational posterior where each inducing input is augmented with additional 'inducing gradients'. For each base-layer inducing input z^1_i, we define the corresponding inducing input $(z^\ell_\nabla)_i \in \mathbb{R}^{d+1}$ for each layer $\ell \geq 2$ and write $Z^\ell_\nabla = \big\{(z^\ell_\nabla)_i\big\}_{i=1}^{m_\ell}$. When using a full-rank variational posterior, we fix the inducing points to the training data locations and set these additional d inducing values for each gradient data point, so we abuse notation and write $z^\ell_i = [Y^{\ell-1}_i, \nabla^\top Y^{\ell-1}_i]^\top$. Analogously to (2.16), we set the form of the gradient-enhanced variational posterior as

$$q\Big(\big\{f^\ell_\nabla,\, U^\ell_\nabla\big\}_{\ell=1}^L\Big) = \prod_{\ell=1}^L p\big(f^\ell_\nabla \,|\, U^\ell_\nabla\big)\, q\big(U^\ell_\nabla\big), \tag{3.8}$$

where $q\big(U^\ell_\nabla\big) = \mathcal{N}\big(m^\ell_{\nabla q},\, S^\ell_{\nabla q}\big)$, with mean $m^\ell_{\nabla q} \in \mathbb{R}^{m^\ell(d+1)}$ and covariance $S^\ell_{\nabla q} \in \mathbb{R}^{m_\ell(d+1)\times m_\ell(d+1)}$. As these terms factorize layer-wise and all distributions are Gaussian, the inducing values can be marginalized out to obtain

$$q\Big(\big\{f^\ell_\nabla\big\}_{\ell=1}^L\Big) = \prod_{\ell=1}^L q\big(f^\ell_\nabla\big) = \prod_{l=1}^L \mathcal{N}\big(\tilde{\mu}^\ell_\nabla,\, \tilde{\Sigma}^\ell_\nabla\big). \tag{3.9}$$

Analogously to the single-layer case, the mean and covariance terms are

$$\begin{aligned}(\tilde{\mu}^\ell_\nabla)_i &= m^\ell_\nabla(f^{\ell-1}_{\nabla i},\, X_i) + \alpha_\nabla(f^{\ell-1}_{\nabla i},\, X_i)^\top(m^\ell_{\nabla q} - m^\ell_\nabla(z^\ell_{\nabla i},\, z^1_i)) \\ (\tilde{\Sigma}^l_\nabla)_{ij} &= k^\ell_\nabla((f^{\ell-1}_{\nabla i},\, X_i),\,(f^{\ell-1}_{\nabla j},\, X_j)) - \alpha_\nabla(f^{\ell-1}_{\nabla i},\, X_i)^\top(K_{Z^\ell_\nabla Z^\ell_\nabla} - S^\ell_{\nabla q})\,\alpha_\nabla(f^{\ell-1}_{\nabla j},\, X_j),\end{aligned} \tag{3.10}$$

where $\alpha_\nabla(f^{\ell-1}_{\nabla i},\, X_i) = K^{-1}_{Z^\ell_\nabla Z^\ell_\nabla} K_{Z^\ell_\nabla f^{l-1}_{\nabla i}}$ and the matrices $K_{Z^l_\nabla Z^l_\nabla}$ and $K_{Z^\ell_\nabla f^{l-1}_{\nabla i}}$ have entries

$$\begin{aligned}(K_{Z^l_\nabla Z^l_\nabla})_{pq} &= k^\ell_\nabla\big((z^\ell_{\nabla p},\, z^1_p),\,(z^l_{\nabla q},\, z^1_q)\big) \\ \big(K_{Z^\ell_\nabla f^{l-1}_{\nabla i}}\big)_{pq} &= k^\ell_\nabla\big((f^{l-1}_{\nabla p},\, X_p),\,(z^l_{\nabla q},\, z^1_q)\big).\end{aligned}$$

Under this construction, $f^\ell_{\nabla i}$ depends only on $f^k_{\nabla i}$ for $k < \ell$, so we can sample from $q\big(f^l_{\nabla i}\big)$ by iterative computation from the base layer to layer ℓ, using samples from the marginal distributions $q\big(f^k_{\nabla i}\big)$ as inputs to the next layer. We remark that for most choices of base kernel, and in particular the squared-exponential, the off-diagonal terms in $k^\ell_\nabla\big((f^{\ell-1}_{\nabla i},\, x_i),\,(f^{\ell-1}_{\nabla i},\, x_i)\big)$ vanish, enabling independent sampling from each $q\big(f^\ell_{\nabla i}\big)$ and thus mitigating scaling issues associated with high input dimensions d. Under our choice of variational distribution, the multifidelity gradient-enhanced ELBO is

$$\mathcal{L}_{\nabla DGP-ELBO}\Big(q\big(\{f^\ell_\nabla,\, U^\ell_\nabla\}_{\ell=1}^L\big)\Big) = \sum_{\ell=1}^L\sum_{i=1}^{n_\ell}\mathbb{E}_{q\left(f^\ell_{\nabla i}\right)}\log p\big(y^\ell_{\nabla i}\,|\,f^\ell_{\nabla i}\big) - \sum_{\ell=1}^L \mathcal{D}_{\mathrm{KL}}\Big(q\big(U^\ell_\nabla\big)\big\|p\big(U^\ell_\nabla\big)\Big), \tag{3.11}$$

where expectations over each $q(f^\ell_{\nabla i})$ are approximated by sampling from the previous layer's distribution $q(f^{\ell-1}_{\nabla i})$. The gradient-enhanced PLL objective is defined from (2.24) analogously.

4 Implementation

This section briefly summarizes the implementation of the multifidelity GPR models presented in Sec. 2 and 3. For fidelities $\ell = 1, ..., L$, we compute the output data $(Y_i^\ell)_{i=1}^{n^\ell}$ by sampling the model $\tilde{f}^\ell$ at the chosen input locations $(X_i^\ell)_{i=1}^{n^\ell}$. We denote the input and output data over all layers as $\mathcal{X} = \{(\ell, X^\ell)\}_{\ell=1}^L$ and $\mathcal{Y} = \{Y^\ell\}_{\ell=1}^L$. Moreover, we denote the desired prediction locations paired with the highest fidelity level as $\mathcal{X}^\star = (L, X^\star)$. Initial values for the hyperparameters and variational parameters are denoted θ_0 and ψ_0 respectively.

Alg. 1, `ComputeLMCPredictions`, describes how the LMC models are constructed from the data. `ComputeKLMC`($\mathcal{X}_1$, $\mathcal{X}_2$, θ) computes the matrix K_{LMC} for the fidelity-input pairs $\mathcal{X}_1$, $\mathcal{X}_2$ and hyperparameters θ using the LMC kernel function (2.6); `ComputeMLL`(K, Y, θ) computes the marginal log likelihood for the gram matrix K, output data Y, and hyperparameters θ using (2.2); and `ComputeFStar`(K, $K^\star$, $K^{\star\star}$, Y) computes the marginal predictive mean and covariance for gram matrices K, $K^\star$, $K^{\star\star}$ and output data Y using (2.1); `OptimizerUpdate`(L, θ) updates the parameters θ using a gradient-based optimizer to reduce the loss L. For a gradient-enhanced LMC model, the gradient kernel k_∇ (2.4) is used in place of a standard kernel when evaluating `ComputeKLMC` and gradient data is included in output data $\mathcal{Y}$.

Alg. 2, `ComputeDeepGPPosterior`, shows how to compute the approximate deep GP variational posterior from input points X, hyperparameters θ, variational parameters ψ, and number of samples N_S. `ComputeMDGP`(X, f, θ) evaluates the deep GP mean function (2.10) for base inputs X, previous-layer outputs f, and hyperparameters θ; `ComputeKDGP`(X_1, X_2, f_1, f_2, θ) evaluates the deep GP kernel function (2.12) for base inputs X_1 and X_2, previous-layer outputs f_1 and f_2, and hyperparameters θ; `ComputeQ`(m_Z, m_X, K_{ZZ}, K_{ZX}, ψ) computes the variational posterior (2.18) for inducing mean m_Z, mean m_X, gram matrices K_{ZZ} and K_{ZX} hyperparameters ψ; `SampleQ`(q) samples the multivariate distribution q by sampling from each variable independently (2.20); `ComputeMixture`($q_1, ..., q_N$) computes the mixture model from the input distributions according to (2.22).

Alg. 3 describes how the deep GP models are built from the data. `EvaluateDGPObjective`($\mathcal{L}$, $\hat{f}$, $\mathcal{Y}$) evaluates the chosen objective function $\mathcal{L}$ with posterior samples $\hat{f}$ and data $\mathcal{Y}$. $\mathcal{L}$ is chosen as either the ELBO (2.23) or PLL objective (2.24). To build a gradient-enhanced deep GP model, the gradient deep GP kernel k_∇^ℓ (3.2) and mean m_∇^ℓ (3.7) are used when evaluating `ComputeKDGP` and `ComputeMDGP`, the gradient-enhanced variational posterior is used when evaluating `ComputeQ` (3.10), gradient data is included in the output data $\mathcal{Y}$, and a gradient objective function is used (see (3.11) for the ELBO).

The models are implemented using `gpytorch` [Gardner et al., 2018], which is built on the `pytorch` [Paszke et al., 2019] framework. Training is performed using the Adam optimizer [Kingma and Ba, 2014], with the required derivatives computed using `pytorch`'s automatic differentiation tools. We found that a four-stage training procedure on normalized data was typically sufficient to converge the optimization problems. These stages consisted of $N_T = 800$ iterations with learning rates of 0.03, 0.01, 0.003, and 0.001 respectively.

Data: $\mathcal{X}^\star, \mathcal{X}, \mathcal{Y}, N_T, \theta_0$
Result: $f^L(X^\star) \,|\, \mathcal{X}, \mathcal{Y}$ `// high-fidelity predictive distribution`
$\theta \leftarrow \theta_0$ `// initialize hyperparameters`
for t *in* $1, ..., N_T$ **do**
  $K_{LMC} \leftarrow$ `ComputeKLMC`$(\mathcal{X}, \mathcal{X}, \theta)$ `// evaluate gram matrix`
  $L \leftarrow$ `ComputeMLL`$(K_{LMC}, \mathcal{Y}, \theta)$ `// evaluate log-marginal likelihood`
  $\theta \leftarrow$ `OptimizerUpdate`(L, θ) `// update hyperparameters`
end
$K_{LMC} \leftarrow$ `ComputeKLMC`$(\mathcal{X}, \mathcal{X}, \theta)$, $K_{LMC}^\star \leftarrow$ `ComputeKLMC`$(\mathcal{X}^\star, \mathcal{X}, \theta)$, $K_{LMC}^{\star\star} \leftarrow$ `ComputeKLMC`$(\mathcal{X}^\star, \mathcal{X}^\star, \theta)$
$\mu_{f^\star}, \Sigma_{f^\star} \leftarrow$ `ComputeFStar`$(K_{LMC}, K_{LMC}^\star, K_{LMC}^{\star\star}, \mathcal{Y})$ `// posterior mean and covariance`
$f^L(X^\star) \,|\, \mathcal{X}, \mathcal{Y} \leftarrow \mathcal{N}(\mu_{f^\star}, \Sigma_{f^\star})$
return $f^L(X^\star) \,|\, \mathcal{X}, \mathcal{Y}$ `// predictive distribution`

Algorithm 1: `ComputeLMCPredictions`

Data: X, θ, ψ, N_S
Result: $\hat{q}$, $\hat{f}$ // variational posterior and corresponding function samples
// Compute first-layer output distribution
$m_Z^1 \leftarrow \texttt{ComputeMDGP}(Z^1, \texttt{None}, \theta)$
$m_X^1 \leftarrow \texttt{ComputeMDGP}(X, \texttt{None}, \theta)$
$K_{ZZ}^1 \leftarrow \texttt{ComputeKDGP}(Z^1, Z^1, \texttt{None}, \texttt{None}, \theta)$
$K_{ZX}^1 \leftarrow \texttt{ComputeKDGP}(Z^1, X, \texttt{None}, \texttt{None}, \theta)$
$q(f^1) \leftarrow \texttt{ComputeQ}(m_Z^1, m_X^1, K_{ZZ}^1, K_{ZX}^1, \psi)$
// Compute N_S posterior samples through all layers
for ℓ *in* 2, ..., L **do**
  $m_Z^\ell \leftarrow \texttt{ComputeMDGP}(Z^1, Z^\ell, \theta)$
  $K_{ZZ}^\ell \leftarrow \texttt{ComputeKDGP}(Z^1, Z^1, Z^\ell, Z^\ell, \theta)$
  for j *in* 1, ..., N_S **do**
    $(\hat{f}^1)^j \leftarrow \texttt{SampleQ}(q(f^1))$
    $m_X^\ell \leftarrow \texttt{ComputeMDGP}(X, (\hat{f}^{\ell-1})^j, \theta)$
    $K_{ZX}^\ell \leftarrow \texttt{ComputeKDGP}(Z^1, X, Z^\ell, (\hat{f}^{\ell-1})^j, \theta)$
    $\hat{q}^j(f^\ell) \leftarrow \texttt{ComputeQ}(m_Z^\ell, m_X^\ell, K_{ZZ}^\ell, K_{ZX}^\ell, \psi)$
    $(\hat{f}^\ell)^j \leftarrow \texttt{SampleQ}(\hat{q}^j(f^\ell))$
  end
end
$\hat{q} \leftarrow \texttt{ComputeMixture}(\hat{q}^1(f^1), ..., \hat{q}^{N_S}(f^L))$ // make approximate posterior
return $\hat{q}$, $\hat{f}$

Algorithm 2: `ComputeDeepGPPosterior`

Data: $\mathcal{X}^\star$, $\mathcal{X}$, $\mathcal{Y}$, N_T, N_S, $N_S^\star$, θ_0, ψ_0, $\mathcal{L}$
Result: $f^L(X^\star) \,|\, \mathcal{X}, \mathcal{Y}$ // high-fidelity predictive distribution
$\theta \leftarrow \theta_0$, $\psi \leftarrow \psi_0$ // initialize hyperparameters and variational parameters
$X_0 \leftarrow \bigcup \mathcal{X}$ // evaluate DGP at all points
for t *in* 1, ..., N_T **do**
  $\hat{q}, \hat{f} \leftarrow \texttt{ComputeDGPPosterior}(X_0, \theta, \psi, N_S)$ // compute approximate posterior
  $L \leftarrow \texttt{EvaluateDGPObjective}(\mathcal{L}, \hat{f}, \mathcal{Y})$ // evaluate objective using samples
  $\theta, \psi \leftarrow \texttt{OptimizerUpdate}(L, (\theta, \psi))$ // update hyperparameters and variational parameters
end
$X_0^\star \leftarrow \bigcup \mathcal{X}^\star$
$\hat{q}, \hat{f} \leftarrow \texttt{ComputeDGPPosterior}(X_0^\star, \theta, \psi, N_S^\star)$ // predictive distribution
$f^L(X^\star) \,|\, \mathcal{X}, \mathcal{Y} \leftarrow \hat{q}^L$
return $f^L(X^\star) \,|\, \mathcal{X}, \mathcal{Y}$ // approximate predictive distribution

Algorithm 3: `ComputeDeepGPPredictions`

5 Numerical examples

5.1 Results: Test problem

We begin by testing our method on a benchmark test problem, the multifidelity Branin function (Eq. 5.1) [Perdikaris et al., 2017], to highlight the differences between each of the GP models. The high-fidelity data is sampled from the Branin function and the medium- and low-fidelity data are generated by nonlinearly transforming the high-fidelity function. The multifidelity Branin function is shown in Fig. 1; note how the relationships between the different fidelities vary over the input space.

$$
\begin{aligned}
f_3(x) &= \left(\frac{-1.275x_1^2}{\pi^2} + \frac{5x_1}{\pi} + x_2 - 6\right)^2 + \left(10 - \frac{5}{4\pi}\right)\cos(x_1) + 10 \\
f_2(x) &= 10\sqrt{f_3(x-2)} + 2(x_1 - 0.5) - 3(3x_2 - 1) - 1 \\
f_1(x) &= f_2(1.2(x+2)) - 3x_2 + 1 \\
x &= [x_1,\, x_2]^\intercal, \quad -5 \le x_1 \le 10, \quad 0 \le x_2 \le 15
\end{aligned}
\tag{5.1}
$$

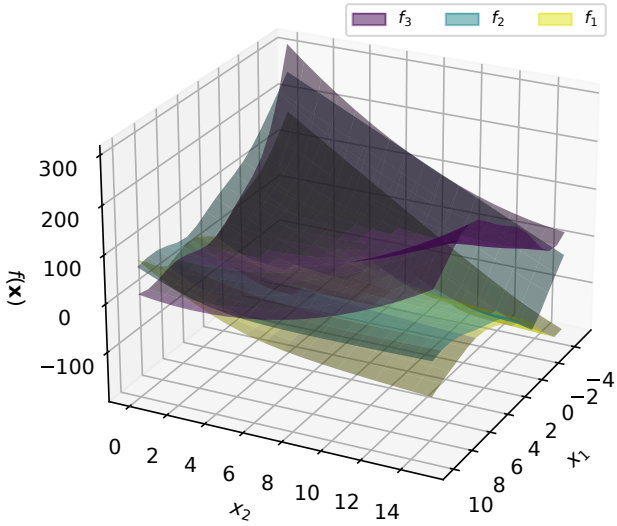


Figure 1: Multifidelity Branin function

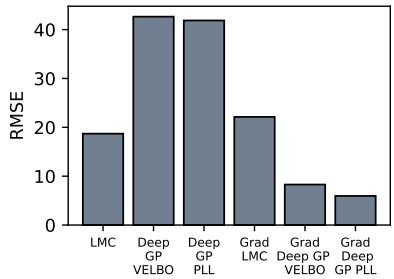


Figure 2: Prediction errors on multifidelity Branin function, medium sampling

We investigate the predictive performance of the GP models for three sampling densities — dense, medium, and sparse — which are detailed in Table 1. Here, N_{f_1}, N_{f_2}, and N_{f_3} denote the number of sample points for each function in (5.1). For the medium and sparse datasets, we use a full-rank variational posterior with the number of inducing points equal to the number of data points. For the dense dataset, we set $m^l = 40$ and use the inducing points as free variational parameters. We evaluate the corner points and centre point of the domain, then add the additional points by uniformly sampling the input space. The predictions for the LMC and deep GP models are generated using Alg. 1 and Alg. 3 respectively. See Sec. 4 for implementations details pertaining to the gradient-enhanced models.

The table gives root mean square prediction error (RMSE) and mean absolute prediction error (MAE), evaluated on 100 unseen randomly distributed test points for f_3.

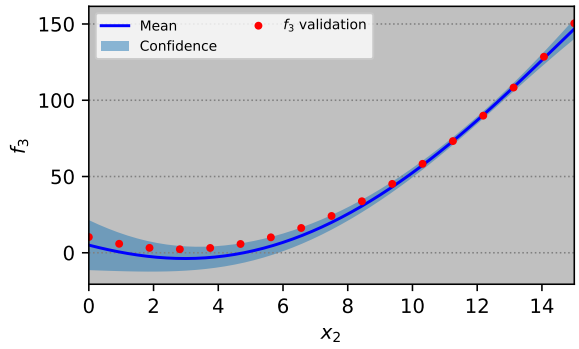
The LMC models use three separable kernels ($R = 3$) and thus these models are generalizations of AR1. We also compare performance for both choices of deep GP training objective: the variational ELBO and the PLL. We set $\beta = 1$ for both choices and use 30 Monte Carlo samples to approximate the expectations. For all cases, the gradient-enhanced deep GP methods outperform the LMC analogue, while the standard LMC model outperforms both the standard deep GP models. Fig. 2 summarizes the performance of the models in terms of RMSE for the 'medium' sampling density. The choice of deep GP objective function does not significantly affect the mean prediction errors of the deep GP models for most cases, although the gradient-enhanced model trained with the PLL objective performance sufffers in the sparse sampling case.

To further inspect the differences between the multifidelity techniques, Fig. 3 shows slices of the gradient-enhanced models for the 'medium' sampling density, with validation samples from the true function f_3 shown in red. This

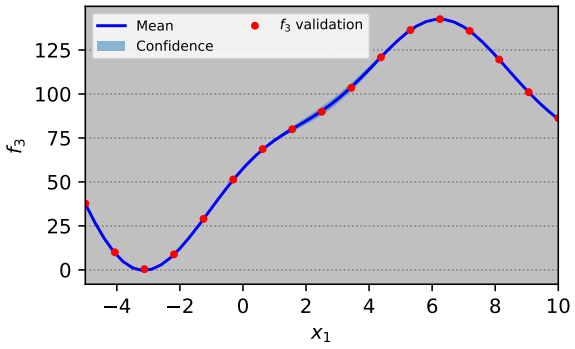
Table 1: Multifidelity Branin results

Sampling	$N_{f_1}, N_{f_2}, N_{f_3}$	**Model**	**Objective**	**RMSE**	**MAE**
Sparse	20, 10, 5	LMC		27.956	20.756
		LMC grad		19.359	15.950
		DGP	ELBO	58.843	51.018
		DGP grad	ELBO	**11.236**	**8.396**
		DGP	PLL	48.712	40.139
		DGP grad	PLL	20.671	15.450
Medium	40, 20, 10	LMC		18.704	9.488
		LMC grad		22.131	6.681
		DGP	ELBO	41.471	24.100
		DGP grad	ELBO	8.287	**2.366**
		DGP	PLL	41.873	23.425
		DGP grad	PLL	**5.956**	2.475
Dense	80, 40, 20	LMC		7.196	4.576
		LMC grad		6.749	2.753
		DGP	ELBO	28.395	18.808
		DGP grad	ELBO	**2.377**	**1.049**
		DGP	PLL	8.652	4.722
		DGP grad	PLL	3.244	1.081

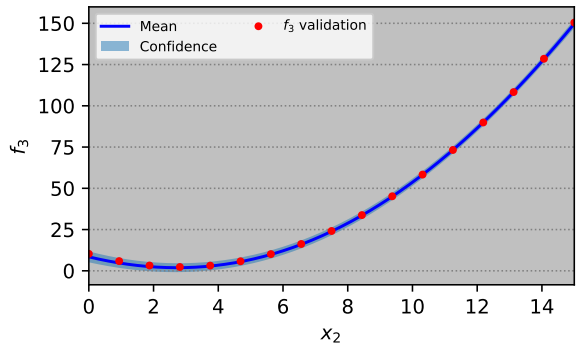
figure shows the mean predictions and uncertainty bounds, given by the mean plus or minus two standard deviations, generated by the GP models. With the ELBO objective, the deep GP model uses output noise to fit the data, resulting in uniform uncertainty bounds. As discussed in [Jankowiak et al., 2020b], with the PLL objective, the model uses internal kernel noise and is better able to account for input-dependent uncertainty.



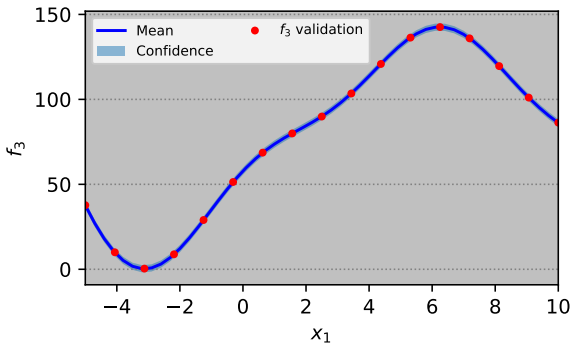
(a) Gradient LMC GP [$x_1 = 2.5$]



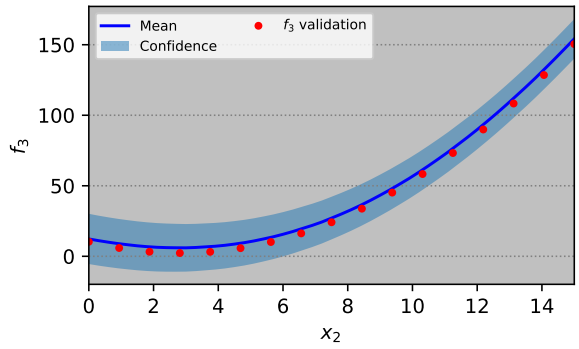
(b) Gradient LMC GP [$x_2 = 12$]



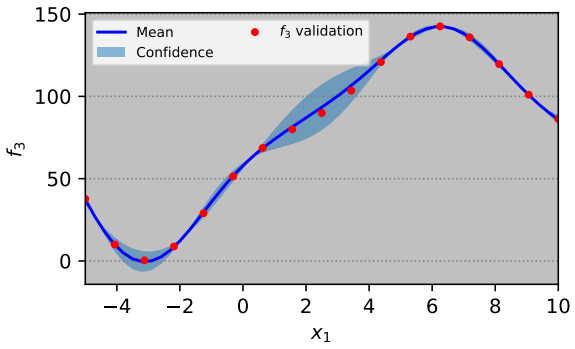
(c) Gradient deep GP, ELBO [$x_1 = 2.5$]



(d) Gradient deep GP, ELBO [$x_2 = 12$]



(e) Gradient deep GP, PLL [$x_1 = 2.5$]



(f) Gradient deep GP, PLL [$x_2 = 12$]

Figure 3: Performance of gradient-enhanced GP models on the multifidelity Branin test problem, 'medium' sampling density

5.2 Results: Aerospace PDE problem

We now consider a practical application, where we use various GP models presented herein to approximate the mapping between boundary conditions for a system of PDEs and integrated output quantities. Specifically, we seek to predict the coefficients of lift C_L, drag C_D, and pitching moment C_M for a hypersonic waverider vehicle geometry (see Fig. 4) over three inputs: free stream Mach number Ma, angle of attack AoA, and body curvature G_c. The body curvature input controls the rate of curvature along the vehicle's longitudinal axis, as illustrated in Fig. 5. The training data is generated by evaluating the flow solver at different levels of mesh discretization, which naturally leads to approximations of the quantities of interest at different fidelity levels.

The flow field around the vehicle is given by the steady-state solution to the Euler equations over a domain Ω enclosing the geometry, closed with the equation of state for air. These solutions to the steady-state Euler equations $u : \Omega \to \mathbb{R}^3$ satisfy

$$\begin{aligned} \nabla \cdot (\rho u) &= 0 \\ \rho(u \cdot \nabla)u + \nabla p &= 0 \\ \nabla \cdot (\rho u E + p u) &= 0 \end{aligned} \tag{5.2}$$

in Ω, subject to the boundary conditions $u = g(\text{Ma}, \text{AoA}, G_c)$ on $\partial\Omega$. Here, $\rho : \Omega \to \mathbb{R}$ is the fluid density, $p : \Omega \to \mathbb{R}$ is pressure, u is the velocity vector, and $E : \Omega \to \mathbb{R}$ is total energy.

The aerodynamic coefficients are computed from the pressure field as

$$\begin{aligned} C_L &= \frac{1}{\frac{1}{2}\rho u_\infty^2 S} \oint (p - p_\infty) n_y \, dS \\ C_D &= \frac{1}{\frac{1}{2}\rho u_\infty^2 S} \oint (p - p_\infty) n_x \, dS \\ C_M &= \frac{1}{\frac{1}{2}\rho u_\infty^2 S L} \oint (p - p_\infty)(x - x_{ac}) n_y \, dS \end{aligned} \tag{5.3}$$

where p_∞ and U_∞ are the free-stream (boundary) pressure and velocity magnitude, S is the planform area of the vehicle, L is the vehicle length, x_{ac} is the coordinate of the centre of pressure, and n_x and n_y arethe components of the unit normal vector to the surface.

As this example considers supersonic flight conditions, the inflow boundary is specified by the free-stream air conditions consisting of pressure p_∞ and temperature T_∞, as well as the inlet Mach number and angle of attack. We use a zero-gradient outflow boundary condition and a no-penetration wall boundary condition.

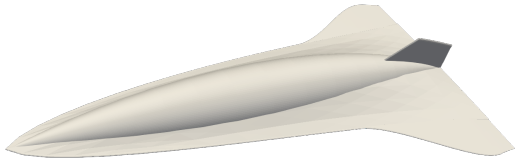


Figure 4: Hypersonic waverider geometry

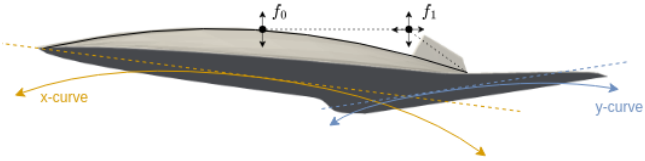


Figure 5: Body curvature (exaggerated)

We use NASA `CART3D` to compute approximate solutions to (5.2) and evaluate the aerodynamic coefficients for each set of boundary conditions. `CART3D` features adjoint-based adaptive mesh refinement using cartesian cut cells [Nemec et al., 2008]. We generate multifidelity data by using this feature to automatically compute solutions for up to seven mesh refinement levels for each set of inputs. We employ three fidelity levels: low, medium, and high, which correspond to three, five, and seven mesh refinements respectively. Table 2 lists the nominal runtime for each fidelity level; the medium and low fidelity data are $5\times$ and $25\times$ cheaper respectively than the high fidelity data. When using a flow solver without automatic mesh refinement, multifidelity information may instead be obtained by externally generating several meshes with different refinement factors, then computing the flow solution for each mesh. The 37 training points this example, shown in Fig. 6, span $2 \leq \text{Ma} \leq 10$, $-3^\circ \leq \text{AoA} \leq 10^\circ$, and $5.0 \leq G_c \leq 5.3$. The corner points and centre point are evaluated to high fidelity (seven levels of mesh refinement) and the remaining points are evaluated to medium or low fidelity. The deep GP variants use a full-rank variational posterior, where all data points used as inducing points. We use 30 Monte Carlo samples during training and 300 Monte Carlo samples when evaluating the prediction errors and generating the results plots. For the gradient-enhanced deep GP, we set the KL divergence scaling factor to $\beta = 2$ for both the ELBO and PLL objectives. For the standard deep GP, we use the usual setting of

Table 2: Data fidelities

Fidelity	Mesh refinements	Runtime [sec]	Speed increase
low	3	230	24.88
medium	5	1230	5.005
high	7	4200	1.0

$\beta = 1$. Again, the predictions for the LMC and deep GP models are generated using Alg. 1 and Alg. 3 respectively, with Sec. 4 detailing implementation of the gradient-enhanced models.

Fig. 7 compares the RMSE prediction error of the multifidelity GP models and the single-fidelity reference model across the three outputs — C_L, C_D, and C_M — on a test set of 50 points. The prediction errors are small, highlighting the effectiveness of GPR models for this application. The gradient-enhanced methods outperform the standard methods in all cases, clearly demonstrating the value of incorporating gradients into GP models. Moreover, for the gradient-enhanced deep GP, models trained with the PLL objective outperform the ELBO objective in all cases, corroborating the findings in [Jankowiak et al., 2020b]. Similarly, using the PLL objective, the gradient-enhanced deep GP outperforms the its LMC analogue, further demonstrating that the deep GP's ability to capture input-dependent function correlations makes them well-suited for multifidelity modelling.

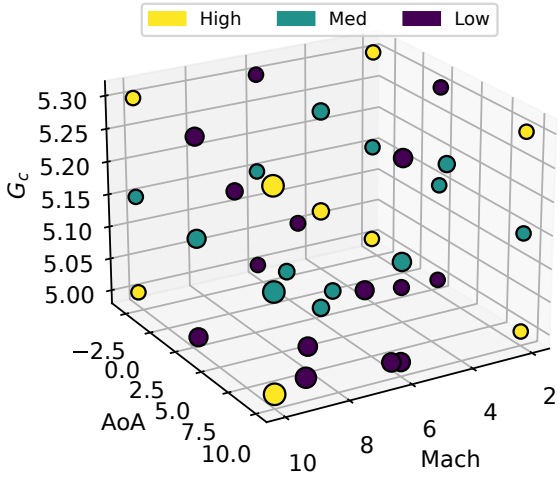


Figure 6: Sampling points for aerospace example

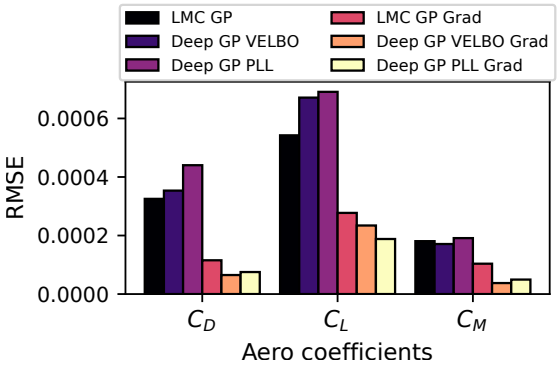
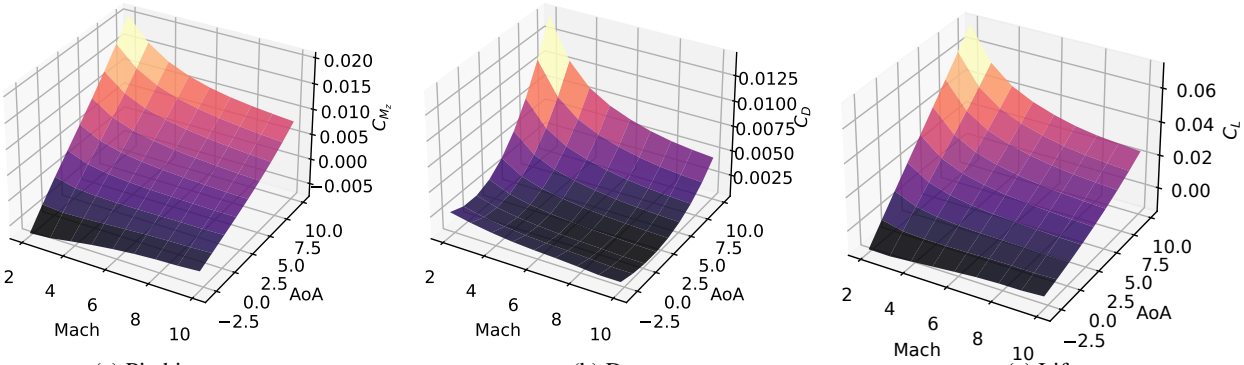


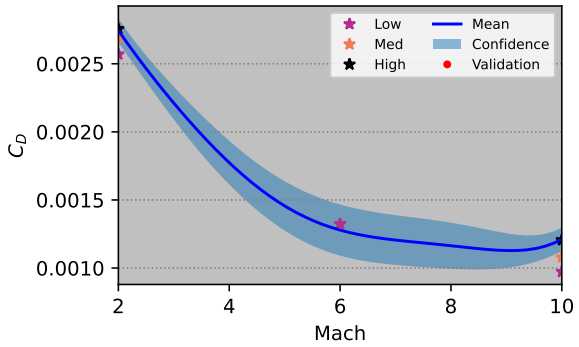
Figure 7: RMSE for aerospace example

Fig. 8 visualizes the aerodynamic coefficient surfaces as a function of Mach number and angle of attack, with the body curvature parameter fixed to 5. This reference surface is generated using a single-fidelity GP model, trained using high fidelity data at all training and test points. Fig. 9 displays three one-dimensional slices with error bounds for the drag coefficient surface. Where applicable, training and test data from the high-fidelity result are also shown.

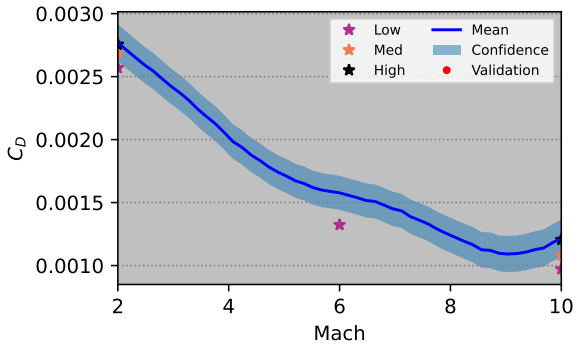


(a) Pitching moment (b) Drag (c) Lift

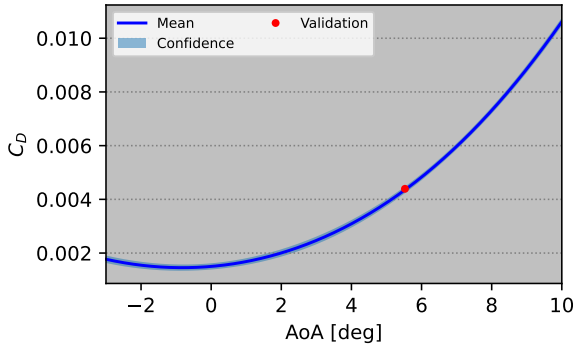
Figure 8: Aerodynamic coefficient maps



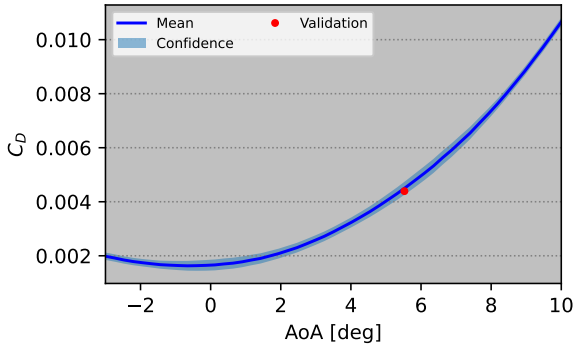
(a) Gradient LMC [AoA=3°, Gc=5.0]



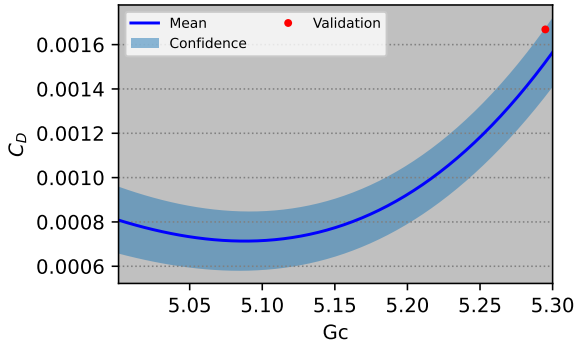
(b) Gradient deep GP (PLL) [AoA=3°, Gc=5.0]



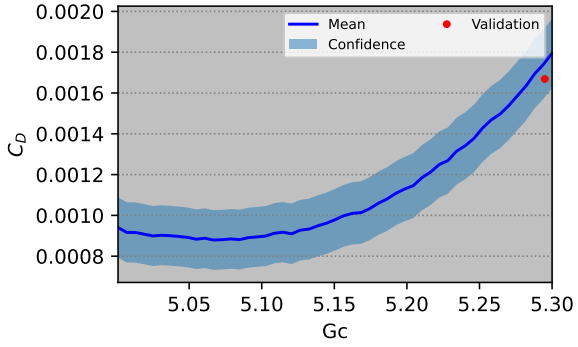
(c) Gradient LMC [Mach=3.495, Gc=5.043]



(d) Gradient deep GP (PLL) [Mach=3.495, Gc=5.043]



(e) Gradient LMC [Mach=7.744, AoA=-1.51°]



(f) Gradient deep GP (PLL) [Mach=7.744, AoA=-1.51°]

Figure 9: Comparison of multifidelity GPR methods

5.3 Discussion

The results presented herein show that (i) deep GPs can be extended to incorporate gradient information and (ii) that this information is useful for improving their performance in the multifidelity setting. For all test cases that we consider, gradient-enhanced deep GPs outperform gradient-enhanced LMC models. In particular, the improved performance of gradient-enhanced deep GPs on the aerospace PDE problem highlights their utility for challenging realistic examples. The comparative performance advantage of gradient-enhanced deep GP models may further increase with for problems with higher dimensional inputs, where the restrictive assumptions that underpin the linear multifidelity models become harder to satisfy, and accurately capturing the input-dependent relationships between different fidelities becomes critical. Furthermore, training on gradient information enables the GPR models to more accurately *predict* gradient information, enabling their deployment when solving parametric optimization problems, such as shape optimization, codesign, and Bayesian optimization.

However, a major drawback of gradient-enhanced GPs is their increased computational cost when compared to standard GPs, whose bottleneck results from inverting the covariance matrix of dimension $(1 + d) \times n$. This increase in computational cost carries over to the gradient-enhanced deep GP, requiring inversion of a $(1 + d) \times m_\ell$ dimensional covariance matrix at each layer. We have focused on relatively small datasets, which permit the use of comparatively many inducing points and data points without excessive computational cost. Future work should consider how gradient-enhanced deep GPs perform on larger datasets, which necessitate the use of many less inducing points than data points and minibatch sampling [Salimbeni and Deisenroth, 2017]. Scaling issues may be further alleviated by compressing the gradient-enhanced covariance matrix via directional derivatives [Padidar et al., 2021] or enhanced optimization techniques [Hebbal et al., 2021], which may reduce the number of required training iterations.

Finally, as discussed in [Jankowiak et al., 2020b,a], the choice of training objective significantly impacts the predictive distributions generated by variational GP models. Most notably, the PLL objective yields significantly richer uncertainty bounds than the standard ELBO objective, which may be useful in Bayesian or robust optimization settings. Future work should extend the gradient-enhancement techniques presented here to the deep sigma point process [Jankowiak et al., 2020a], which does not result in a biased estimator when coupled with the PLL objective.

6 Conclusion

This work presented a new deep GP model which incorporates gradient data, then applied this model in the multifidelity setting. This capability is particularly relevant when constructing surrogate models for discretized PDE solvers, where gradient information can be obtained cheaply via adjoint solutions. For the examples presented herein, the gradient-enhanced deep GP significantly outperforms a standard deep GP. In contrast to existing multifidelity gradient-enhanced methods, the gradient-enhanced deep GP can handle nonlinear input-dependent relationships between fidelity levels. This method leverages sparse VI and can thus be scaled to relatively large datasets.

References

References

Mauricio A Alvarez, Lorenzo Rosasco, Neil D Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

Mohamed A Bouhlel and Joaquim RRA Martins. Gradient-enhanced kriging for high-dimensional problems. *Engineering with Computers*, 35(1):157–173, 2019.

Gilles Bourgault and Denis Marcotte. Multivariable variogram and its application to the linear model of coregionalization. *Mathematical Geology*, 23:899–928, 1991.

Loïc Brevault, Mathieu Balesdent, and Ali Hebbal. Overview of gaussian process based multi-fidelity techniques with variable relationship between fidelities, application to aerospace systems. *Aerospace Science and Technology*, 107: 106339, 2020.

Kurt Cutajar, Mark Pullin, Andreas Damianou, Neil Lawrence, and Javier González. Deep gaussian processes for multi-fidelity modeling. *arXiv preprint arXiv:1903.07320*, 2019.

Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):408–423, 2013.

Volker L Deringer, Albert P Bartók, Noam Bernstein, David M Wilkins, Michele Ceriotti, and Gábor Csányi. Gaussian process regression for materials and molecules. *Chemical Reviews*, 121(16):10073–10141, 2021.

Alexander IJ Forrester, András Sóbester, and Andy J Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the royal society a: mathematical, physical and engineering sciences*, 463(2088):3251–3269, 2007.

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31, 2018.

Alan E Gelfand and Erin M Schliep. Spatial statistics and gaussian processes: A beautiful marriage. *Spatial Statistics*, 18:86–104, 2016.

Pierre Goovaerts. Ordinary cokriging revisited. *Mathematical Geology*, 30:21–42, 1998.

Zhong-Hua Han, Stefan Görtz, and Ralf Zimmermann. Improving variable-fidelity surrogate modeling via gradient-enhanced kriging and a generalized hybrid bridge function. *Aerospace Science and technology*, 25(1):177–189, 2013.

Ali Hebbal, Loïc Brevault, Mathieu Balesdent, El-Ghazali Talbi, and Nouredine Melab. Bayesian optimization using deep gaussian processes with applications to aerospace system design. *Optimization and Engineering*, 22:321–361, 2021.

Jeffrey D Helterbrand and Noel Cressie. Universal cokriging under intrinsic coregionalization. *Mathematical Geology*, 26:205–226, 1994.

Antony Jameson. Aerodynamic design via control theory. *Journal of scientific computing*, 3:233–260, 1988.

Martin Jankowiak, Geoff Pleiss, and Jacob Gardner. Deep sigma point processes. In *Conference on Uncertainty in Artificial Intelligence*, pages 789–798. PMLR, 2020a.

Martin Jankowiak, Geoff Pleiss, and Jacob Gardner. Parametric gaussian process regressors. In *International Conference on Machine Learning*, pages 4702–4712. PMLR, 2020b.

Marc C Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Loic Le Gratiet and Josselin Garnier. Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. *International Journal for Uncertainty Quantification*, 4(5), 2014.

Haitao Liu, Yew-Soon Ong, Jianfei Cai, and Yi Wang. Cope with diverse data structures in multi-fidelity modeling: a gaussian process method. *Engineering Applications of Artificial Intelligence*, 67:211–225, 2018.

Trent William Lukaczyk. *Surrogate modeling and active subspaces for efficient optimization of supersonic aircraft*. Stanford University, 2015.

Siva Nadarajah and Antony Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. In *38th Aerospace sciences meeting and exhibit*, page 667, 2000.

Marian Nemec, Michael Aftosmis, and Mathias Wintzer. Adjoint-based adaptive mesh refinement for complex geometries. In *46th AIAA Aerospace Sciences Meeting and Exhibit*, page 725, 2008.

Misha Padidar, Xinran Zhu, Leo Huang, Jacob Gardner, and David Bindel. Scaling gaussian processes with derivative information using variational inference. *Advances in Neural Information Processing Systems*, 34:6442–6453, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *Siam Review*, 60(3):550–591, 2018.

Paris Perdikaris, Maziar Raissi, Andreas Damianou, Neil D Lawrence, and George Em Karniadakis. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2198):20160751, 2017.

Anandaroop Ray and David Myer. Bayesian geophysical inversion with trans-dimensional gaussian process machine learning. *Geophysical Journal International*, 217(3):1706–1726, 2019.

Tomer Rokita and Peretz P Friedmann. Multifidelity cokriging for high-dimensional output functions with application to hypersonic airloads computation. *AIAA Journal*, 56(8):3060–3070, 2018.

Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep gaussian processes. *Advances in neural information processing systems*, 30, 2017.

Michalis Titsias and Neil D Lawrence. Bayesian gaussian process latent variable model. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 844–851. JMLR Workshop and Conference Proceedings, 2010.

Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson education, 2007.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Yue Wu, José Miguel Hernández-Lobato, and Zoubin Ghahramani. Gaussian process volatility model. *Advances in neural information processing systems*, 27, 2014.