IOP Publishing Mach. Learn.: Sci. Technol. 5 (2024) 045015 https://doi.org/10.1088/2632-2153/ad7ad5

MACHINE LEARNING
Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
23 April 2024

REVISED
29 July 2024

ACCEPTED FOR PUBLICATION
13 September 2024

PUBLISHED
15 October 2024

Original Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence.

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

PAPER

Multi-fidelity Gaussian process surrogate modeling for regression problems in physics

Kislaya Ravi[1,4,*], **Vladyslav Fediukov**[1,2,4,*], **Felix Dietrich**[1], **Tobias Neckel**[1], **Fabian Buse**[2], **Michael Bergmann**[3] **and Hans-Joachim Bungartz**[1]

[1] School of Computation, Information and Technology, Technische Universität München, Garching b. München, Germany
[2] Institute of System Dynamics and Control, German Aerospace Center (DLR), Weßling, Germany
[3] Max-Planck-Institut für Plasmaphysik, Garching b. München, Germany
[4] These authors contributed equally to this work.
[*] Authors to whom any correspondence should be addressed.

E-mail: kislayaravi@tum.de and vladyslav.fediukov@dlr.de

Keywords: multi-fidelity, machine learning, Gaussian processes, physical simulations

Abstract

One of the main challenges in surrogate modeling is the limited availability of data due to resource constraints associated with computationally expensive simulations. Multi-fidelity methods provide a solution by chaining models in a hierarchy with increasing fidelity, associated with lower error, but increasing cost. In this paper, we compare different multi-fidelity methods employed in constructing Gaussian process surrogates for regression. Non-linear autoregressive methods in the existing literature are primarily confined to two-fidelity models, and we extend these methods to handle more than two levels of fidelity. Additionally, we propose enhancements for an existing method incorporating delay terms by introducing a structured kernel. We demonstrate the performance of these methods across various academic and real-world scenarios. Our findings reveal that multi-fidelity methods generally have a smaller prediction error for the same computational cost as compared to the single-fidelity method, although their effectiveness varies across different scenarios.

1. Introduction

Complex simulations are often computationally expensive and time-consuming. A surrogate model is a simpler and faster model that emulates the output of a complex model as a function of input parameters. Surrogates, also known as digital twins, have various applications in design optimization [1, 2], uncertainty quantification [3, 4], real-time predictions [5, 6], etc. The amount of training data is one of the crucial factors governing the quality of the surrogate. Generating an extensive training data set is computationally infeasible for expensive simulations. In this work, we tackle the data availability limitation using Multi-fidelity [7].

The advancement of computational capabilities significantly increased the use of numerical simulation in almost every field of application. This led to the development of various simulation methods offering different levels of approximation quality and computational costs. Low-fidelity models provide estimations with reduced accuracy but demand fewer resources, whereas high-fidelity models yield precise predictions at a higher cost. These distinct fidelity levels arise from differences in physical modeling or simulation of the same phenomena, the linearization of complex physical processes, or the application of the same modeling approach with different discretization levels. The trade-off between simulation accuracy and computational cost is an ever-present challenge. Relying solely on high-fidelity models for applications that require multiple evaluations of the model is impractical due to their computational cost. Conversely, utilizing only low-fidelity models may compromise the results' accuracy. Multi-fidelity methods address this challenge by leveraging low-fidelity models to alleviate computational load and incorporating occasional high-fidelity evaluations to control result accuracy.

© 2024 The Author(s). Published by IOP Publishing Ltd

Multi-fidelity methods are widely used in applications like surrogate modeling, uncertainty quantification [8–11], bayesian inference [12–14], optimization [15–20], etc. They are helpful in scenarios where resource considerations and accurate estimations are important simultaneously.

This work focuses on using multi-fidelity models to build a Gaussian Process (GP) [21] surrogate. We use GP because of its probabilistic nature and additional functionalities like Bayesian optimization and uncertainty quantification. There are multiple ways to incorporate multi-fidelity in GP. The first method developed was the linear auto-regressive models [22, 23]. The linear modeling was not sufficient for more complex problems. This led to the development of non-linear auto-regression methods [24, 25]. However, the non-linear methods cannot be extended to more than two levels of fidelity and require low-fidelity evaluations during prediction. One can use Deep Gaussian Process (DGP) [26] to tackle the limitations.

This paper is organized as follows. We provide the basic theoretical background on GP, kernels, and calibration in section 2. Then, we review different multi-fidelity GP surrogate models in section 3. We also suggest modifications in existing non-linear autoregressive to incorporate more than two fidelities without DGP. In the same section, we suggest using delay terms in multi-fidelity DGP. Then, we compare the performance of different methods on the academic problem and two real-world problems, namely Terramechanical problems and Plasma microturbulence simulations in section 4. Finally, we end the paper with concluding remarks in section 5.

2. Background

2.1. Gaussian process regression

Gaussian processes (GP) are an efficient and flexible tool for probabilistic predictions [21]. They provide reliable statistical inference as well as uncertainty predictions. A GP is a stochastic process in which each finite subset of variables forms a multivariate Gaussian distribution. GPs are defined by their covariance (kernel) function k and define a probability distribution over functions,

$$f(\cdot,\cdot) \sim \mathcal{GP}\left(0, k(\cdot,\cdot)\right). \tag{1}$$

The mean function is usually assumed to be zero for simplicity. In practice, the base space of the process is sampled with a finite number of N points $X = \{x_i\}_{i=1}^{N}$. Then, the GP is given as a finite-dimensional, multivariable Gaussian distribution of points y,

$$y \sim \mathcal{N}\left(\mu, K\right), \tag{2}$$

where μ is a vector of means, usually zeros, and $K = (K_{ij})_{i,j=1}^{N} = (k(x_i, x_j))_{i,j=1}^{N}$ is a covariance matrix of size $N \times N$. Gaussian distributions can be conveniently employed in a Bayesian framework because they are closed under condition and conjugate distributions, which means marginal and conditional distributions of multivariate Gaussians are again Gaussian.

GP is typically used in regression, also called the 'Kriging method' [27]. In this case, each training point $x_i \in X$ is assigned an additional function value y_i, and the goal is to construct function values y^* for unobserved test points $X^* = \{x^*\}$. Then, we can construct a joint distribution

$$\begin{bmatrix} f(X) \\ f(X^*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix}\right). \tag{3}$$

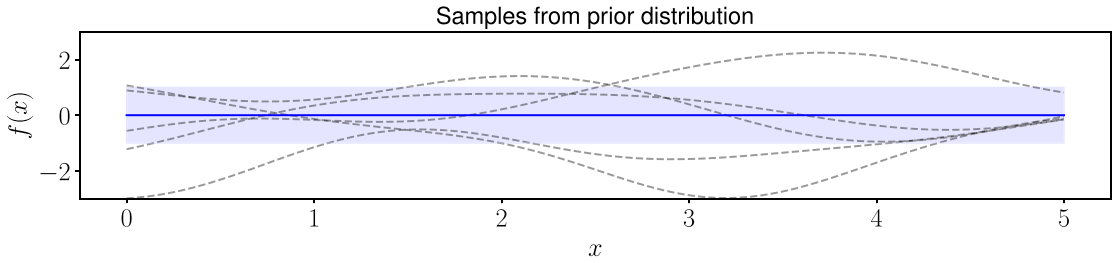
After we observe the training data, the posterior consists of a narrowed distribution of functions, illustrated in figure 1(b), initially defined by the prior distribution as observed in figure 1(a). The inference is done then using posterior mean and posterior covariance defined as

$$\mathbb{E}\left(f(X^*)\right) = K(X,X^*)^T K(X,X)^{-1} f(X), \tag{4}$$

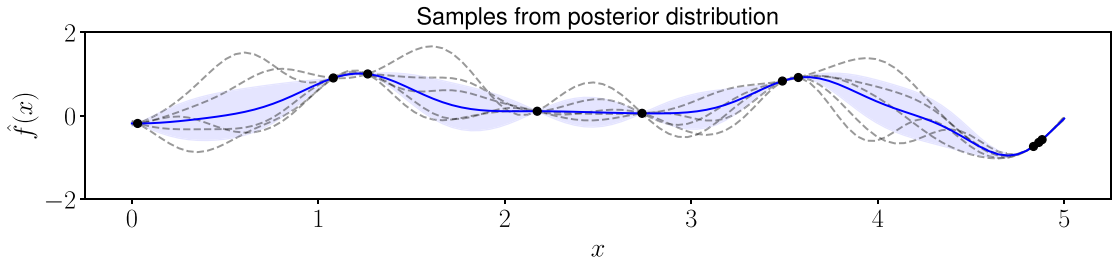
$$cov\left(f(X^*)\right) = K(X^*,X^*) - K(X,X^*)^T K(X,X)^{-1} K(X^*,X). \tag{5}$$

2.2. Kernels

K denotes a matrix built with the help of a *kernel*, which is a function used to define the covariances between the data points. This covariance is usually interpreted as a measure of similarity between data points, and thus, points x and x' that are considered similar in the input space will have target values y and y' that are also similar.



(a) Visualization of Gaussian process prior.



(b) Visualization of Gaussian process posterior.

Figure 1. A kernel function defines the family of the functions with which we are approximating the target function. In this example, we use squared-exponential or RBF kernel and visualize (a) a prior distribution over the functions. (b) In the posterior distribution in the second plot, newly observed training data points, depicted here in black, narrow the space of potential functions from the initial family, defined by the RBF kernel.

A valid covariance function must be positive semidefinite [21]. The product and sum of two valid covariance functions generate a valid covariance function. Using this property, one can generate more sophisticated covariance functions.

There are multiple options for covariance functions like squared-exponential, exponential, trigonometric, etc. The task of choosing the right kernel is complicated. There is no standardized algorithm to do so except for the random search and the researcher's intuition regarding modeling. However, some authors are making promising progress by creating interesting heuristics and algorithms [28–31]. In this work, we only work with squared-exponential covariance functions, because it yields smooth, i.e. mean-square differentiable, sample paths from the resulting GP, while others, like exponential kernel, do not [21].

After choosing the kernel family, we evaluate the hyperparameters of the kernel by maximizing the log-likelihood. The log-likelihood L as a function of observed values of data points Y, parameterized by the hyperparameters θ as

$$\log\left(p\left(y|X,\theta\right)\right) = -\frac{1}{2}y^{T}{K_{y}}^{-1}y - \frac{1}{2}\log|K_{y}| - \frac{n}{2}\log 2\pi. \tag{6}$$

We will use a gradient-based method (LBFGS [32] with multiple starting points) to find the maximum likelihood.

2.3. Calibration

As shown in equation (6), hyperparameter optimization for Gaussian process kernels is usually performed by maximizing the marginal likelihood. This often leads to an acceptable mean squared error [21], but the posterior variance tends to be lower than the variance of the actual distribution [33, 34]. This problem requires an additional post-processing step known as calibration. There are various calibration techniques, but in all cases, they aim to adjust the predicted variance towards the actual variance in the data. Calibration is an established concept in the classification context, but in the case of regression, it is relatively new [35]. In general, calibration can be split into three categories: *quantile*, *distribution*, and *variance* calibration.

The regression is *quantile calibrated*, if

$$\frac{1}{N}\sum_{n=1}^{N}\mathbf{I}\left\{y_{n} < F_{n}^{-1}\left(p\right)\right\} \to p \text{ for all } p \in [0,1] \text{ as } N \to \infty \tag{7}$$

for the set of data $(x_n, y_n)_{n=1}^{N}$, where F_n stands for predicted cumulative distribution at x_n and F_n^{-1} the corresponding quantile function. Intuitively, $X\%$ of the confidence interval should capture the ground truth

values in $X\%$ predictions [36, 37]. Quantile calibration is considered a global calibration, as it concentrates only on the marginal distribution without considering a distribution calibration around each exact prediction.

Distribution calibration resembles the calibration approach in classification tasks because it is also focused on local calibration. In the classification case, instances are grouped by their predicted probability and considered calibrated only if each group matches the indicated probability. Assume X as input variables, Y as a target variable, and the regression model $f: \mathbb{X} \rightarrow \mathbb{S}_{\mathbb{Y}}$ or intuitively, the regression model is mapping from input features into a space of distributions and for each test instance predicted a particular distribution. Denoting s as an arbitrary distribution predicted by the regression model, the regression model f is distribution calibrated if

$$p\left(Y = y | f_{Y|X} = s\right) = s\left(y\right) \text{for all } y \in Y, s \in S, \tag{8}$$

or intuitively, for all predicted instances with a particular distribution s, all instances on average with such prediction should follow this distribution s [36].

Variance calibration aims to make the model predict its own error, i.e. for all instances where variance $\sigma^2(x)$ takes a certain value ω, the squared predicted error $\mathbb{E}_{X,Y}[\mu(x) - y]$ will match this predicted variance [38]. Given $\mu(x)$ is a predicted mean and $\sigma^2(x)$ is a predicted variance, then the regression model is variance calibrated if

$$\mathbb{E}_{X,Y}\left[\left(\mu\left(x\right) - y\right)^2 | \sigma^2\left(x\right) = \omega\right] = \omega. \tag{9}$$

The first method performs a global calibration, while the last two perform a local one. Global calibration is a better choice in case we need only to rescale the marginalized distribution, but in the case when variance should be rescaled only in certain places, it is better to use local calibration. Knowing *a priori* the required type of rescaling is hard for real-world data, so if we do not have any prior assumptions on the real variance, we must empirically find the most suited calibration method.

3. Methodology

3.1. Multi-fidelity

In this paper, our focus is on exploring various multi-fidelity Gaussian process surrogate modeling methods. In the upcoming sub-sections, we delve into the motivation behind each method, elucidate their formulation, and critically evaluate their limitations. Additionally, we propose enhancements for some methods within the corresponding sub-sections to address identified shortcomings.

3.1.1. Linear auto-regressive model

We start with the Auto-Regressive Model (AR1), a straightforward linear auto-regressive model introduced by Kennedy and O'Hagan [22]. In this model, we formulate the joint distribution of all fidelities, building on the fundamental concept of expressing the model for a particular fidelity as the sum of a linear scaling of the previous fidelity and an additive correction term.

We explain AR1 using a two-fidelity case. This process can be easily extended to cases with more than two fidelities. Let us consider the following random variables:

$$\begin{aligned} u_l \sim \mathcal{N}\left(0, K_l\right) \Rightarrow \mathbb{E}\left[u_l\right] = 0 \quad \text{and} \quad \mathbb{E}\left[u_l u_l'\right] = K_l\left(X, X'\right), \\ u_\delta \sim \mathcal{N}\left(0, K_\delta\right) \Rightarrow \mathbb{E}\left[u_\delta\right] = 0 \quad \text{and} \quad \mathbb{E}\left[u_\delta u_\delta'\right] = K_\delta\left(X, X'\right). \end{aligned}$$

We are modeling all the surrogates and the additive correction term using GP. u_l and u_δ represent the samples drawn from the low-fidelity and the additive correction GP. Let u_h represent a realization of the GP that approximates a high-fidelity function f_h. $\rho \in \mathbb{R}$ is a linear-scaling parameter. We assume an additive relationship between the consecutive fidelities so that

$$u_h = \rho u_l + u_\delta. \tag{10}$$

The expected value of the surrogate of the high-fidelity model is assumed to be zero,

$$\mathbb{E}\left[u_h\right] = \mathbb{E}\left[\rho u_l + u_\delta\right] = \rho \mathbb{E}\left[u_l\right] + \mathbb{E}\left[u_\rho\right] = 0. \tag{11}$$

The covariance of the surrogate of the high-fidelity model is then given by

$$\begin{aligned} Cov(u_h, u_h') &= \mathbb{E}[u_h u_h'] - \underbrace{\mathbb{E}[u_h]\mathbb{E}[u_h']}_{=0} \\ &= \mathbb{E}[(\rho u_l + u_\delta)(\rho u_l') + u_\delta'] = \rho^2 \mathbb{E}[u_l u_l'] + \mathbb{E}[u_\delta u_\delta'] = \rho^2 K_l + K_\delta. \end{aligned}$$

The covariance between u_l and u_h is

$$\begin{aligned} Cov(u_h, u_l') &= \mathbb{E}[u_h u_l'] - \underbrace{\mathbb{E}[u_h]\mathbb{E}[u_l']}_{=0} = \mathbb{E}[(\rho u_l + u_\delta) u_l'] \\ &= \rho \mathbb{E}[u_l u_l'] + \mathbb{E}[u_\delta u_l'] = \rho K_l + \underbrace{\mathbb{E}[u_\delta]\mathbb{E}[u_l']}_{=0} = \rho K_l. \end{aligned}$$

The joint distribution of samples drawn from the low-fidelity surrogate and the high-fidelity surrogate is written

$$\begin{bmatrix} u_h \\ u_l \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_\delta + \rho^2 K_l & \rho K_l \\ \rho K_l & K_l \end{bmatrix} \right). \tag{12}$$

Using the maximum likelihood method, one can calculate the value of ρ and the kernel hyperparameters. We can also extend the joint distribution in equation (12) to incorporate more than two fidelities that will lead to covariance matrices with sparse blocks. Le Gratiet [23] took advantage of this block structure and decreased the complexity of the method from $\mathcal{O}((\sum_{l=1}^{L} N_l)^3)$ to $\sum_{l=1}^{L} \mathcal{O}(N_l^3)$. He also improved the performance using polynomial terms for scaling (ρ) instead of a constant term.

Nevertheless, the efficacy of the Auto-Regressive Model (AR1) is limited by the assumption of a linear dependency between the low-fidelity and high-fidelity functions. This assumption might prove inadequate in scenarios characterized by non-linear transformations.

3.1.2. Non-linear auto-regressive models

Let us consider a two-fidelity case. A non-linear transformation on the low-fidelity function to obtain the high-fidelity function is written as

$$f_h(x) = g(x, f_l(x)), \tag{13}$$

where $g : \mathbb{R}^{d+1} \to \mathbb{R}$ is a function representing the non-linear transformation applied on the low-fidelity function to obtain the high-fidelity model. When attempting to model both the low-fidelity and high-fidelity functions using GPs, writing the joint distribution of the surrogates, as presented in equation (12), becomes analytically challenging for a general transformation g. To overcome this difficulty, a common assumption in studies such as [24, 25] is that the low-fidelity function can be evaluated at zero cost. This eliminates the need for a surrogate for the low-fidelity function. Consequently, the non-linear transformation depicted in equation (13) transforms into creating a surrogate in a $d+1$ dimensional space. In our work, we delve into the modeling of g using a Gaussian Process (GP). This approach enables us to effectively navigate the challenges associated with the joint distribution of surrogates, providing a viable solution for incorporating non-linear transformations within the multi-fidelity surrogate modeling framework.

Let us consider the following pedagogical examples where the high-fidelity model f_h and the lower-fidelity model f_l are defined as

$$\begin{aligned} f_h(x) &= \left(\sqrt{2} - x\right) \sin^3(8\pi x), \\ f_l(x) &= \sin(8\pi x). \end{aligned}$$

Figure 2 illustrates the relationship between f_l and f_h. Modeling the transformation g as described in equation (13) is synonymous with learning the manifold depicted in figure 2. Crucially, this manifold lacks oscillations, signifying that learning it necessitates relatively fewer high-fidelity function evaluations. This observation underscores the advantage of multi-fidelity modeling in this example, as it allows for a learning process by leveraging the complicated feature to be captured in the low-fidelity function, thereby reducing the requirement of high-fidelity function evaluations.

A refinement of the method involves incorporating a well-structured kernel into the GP model, as suggested by Peridikaris *et al* [25]. The proposed kernel structure is given by:

$$k(x, x') = k_\rho(x, x'; \theta_\rho)\, k_f\left(f_l(x), f_l(x'); \theta_f\right) + k_\delta(x, x'; \theta_\delta). \tag{14}$$

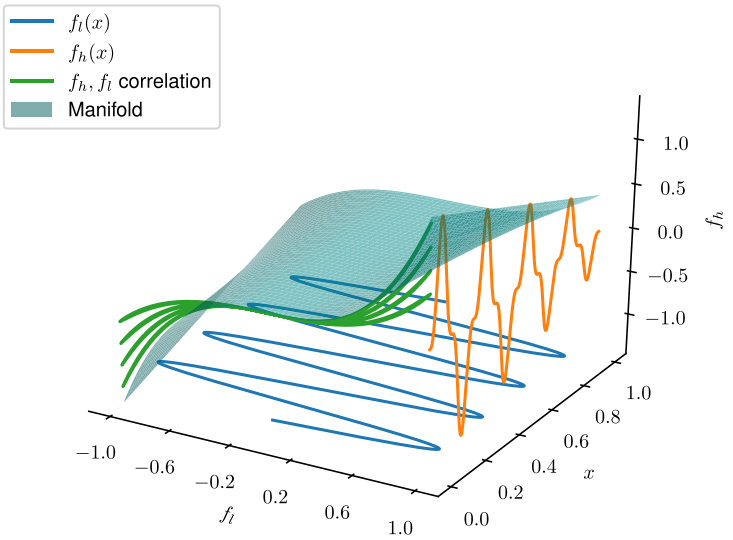


Figure 2. It is difficult to approximate the function $f_h(x)$ (orange) using only x as an input, because it is highly oscillatory. Including the function $f_l(x)$ (blue) as an additional input means only the green, smooth manifold must be approximated, which can be done accurately with only a few data points.

Here, θ_ρ, θ_f, and θ_δ denote the kernel hyperparameters, which are determined by maximizing the likelihood, as discussed in the previous section. This kernel structure closely mirrors the Auto-Regressive Model (AR1) formulation described in equation (10), assigning each kernel to an individual part of the transformation. Specifically, k_ρ models the scaling term, k_f is responsible for transforming the low-fidelity model, and k_δ handles the additive correction term. The formulation above gives the method its name: Non-linear Auto-Regressive Gaussian Process (NARGP).

Introducing this well-defined prior aids the model in accurately fitting the underlying relationships. A drawback of this formulation is the increase in the number of hyperparameters, which necessitates careful consideration during the optimization process. Despite this challenge, the structured nature of the kernel contributes to a more informed and effective surrogate modeling approach.

We follow Lee *et al* [24] and assume sufficient smoothness of the high-fidelity function f_h. We can then use Taylor expansion to approximate the value of f_h close to x by

$$f_h(x+\Delta x) = f_h(x) + \sum_{i=1}^{d} \frac{\partial}{\partial x_i} f_h(x)\,\Delta x_i + \mathcal{O}\left(\Delta x^2\right). \tag{15}$$

Substituting equation (13) into equation (15), we obtain

$$\begin{aligned} f_h(x+\Delta x) &= g(x, f_l(x)) + \sum_{i=1}^{d} \left(\frac{\partial g}{\partial x_i} g(x, f_l(x)) \right) \Delta x_i + \mathcal{O}\left(\Delta x^2\right) \\ &= g(x, f_l(x)) + \sum_{i=1}^{d} \left(\frac{\partial g}{\partial x_i} + \frac{\partial g}{\partial f_l} \frac{\partial f_l}{\partial x_i} \right) \Delta x_i + \mathcal{O}\left(\Delta x^2\right). \end{aligned}$$

The Taylor expansion converges if

(i) $\left\| \frac{\partial g(\cdots)}{\partial x} \right\| \leqslant c_g$ for $c_g \in \mathbb{R}^+$
(ii) $\left\| \frac{\partial f_l(\cdots)}{\partial x} \right\| \leqslant c_l$ for $c_l \in \mathbb{R}^+$.

The Taylor expansion also motivates us to incorporate the derivative of low-fidelity in equation (13). Lee *et al* [24] suggest writing the non-linear transformation as

$$f_h(x) = g_d\left(x, f_l(x), \frac{\partial f_l}{\partial x}\right). \tag{16}$$

In many legacy simulators, the gradients are unavailable [39]. In cases where gradients are unavailable, one can resort to finite difference techniques to approximate the derivative. Lee *et al* [24] suggest replacing the

derivative in equation (16) with the delay term to accommodate such scenarios. This delay term evaluates the low-fidelity function after a small time step $\tau \in \mathbb{R}$. For a higher-dimensional input variable, we add a delay along each dimension, defining τ_i as a vector of zeros with τ at the i^{th} position. The transformed formulation is now expressed as

$$f_h(x) = g_d\left(x, f_l(x), f_l(x+\tau_1), f_l(x+\tau_2), \ldots, f_l(x+\tau_d)\right). \tag{17}$$

This method is called the Gaussian Process with Delay Fusion (GPDF) proposed in [24]. Additionally, we propose an enhancement to this method by modifying the kernel to mimic the Taylor expansion using a structured composite kernel similar to equation (14)

$$k(x,x') = k_\rho(x,x';\theta_\rho)\, k_f\left(\left(f_l(x), f_l(x+\tau)\right), \left(f_l(x'), f_l(x'+\tau)\right); \theta_f\right) + k_\delta(x,x';\theta_\delta). \tag{18}$$

This improved method is called the Gaussian Process with Delay Fusion and Composite kernel (GPDFC).

The formulations of NARGP, GPDF, and GPDFC come with certain limitations that need to be addressed for broader applicability:

(i) **Nested Evaluation Requirements:**

- NARGP: Requires low-fidelity model evaluations during training and prediction at precisely the same parameters as the high-fidelity model.
- GPDF and GPDFC: Share similar constraints and involve additional low-fidelity function evaluations at delay points.

These limitations were initially disregarded under the assumption of infinitely cheap low-fidelity function evaluations. However, in practice, even low-fidelity function evaluations have nonzero cost, and so these constraints can also become significant. In some cases, we are simply given data collected from the low and high-fidelity models. Gathering any further data might not be possible, and for such cases, ensuring nested training data points may also not be possible. Moreover, the possible absence of low-fidelity data at the prediction parameter point also serves as a hurdle in implementing the methods mentioned in data-driven multi-fidelity problems [26].

(ii) **Limited to Two Fidelity Cases:** Modern NARGP, GPDF, and GPDFC formulations are tailored for scenarios involving only two fidelity models, restricting their application in situations with more than two fidelities.

(iii) **Overfitting:** As shown in [26], NARGP is prone to overfitting, which leads to poor generalization.

To overcome the limitation of application to a two-fidelity case and extend the framework to accommodate multiple fidelity levels, we can explore two scenarios:

- **Nested training data:** In the case of nested training data, where ($X_{\ell+1}^{\text{train}} \subset X_{\ell}^{\text{train}}$), we propose training a Gaussian Process surrogate for the lowest fidelity and multi-fidelity Gaussian Process surrogates for higher levels. We train each multi-fidelity surrogate using the previous level, as shown in figure 3(a) for three fidelity NARGP surrogates. These layers are then cascaded, forming a stacked surrogate for multiple fidelities. During prediction, we draw samples from the surrogate of the previous level to obtain prediction samples for the next level and marginalize it to derive the posterior prediction for the given level. Let us represent y_ℓ as posterior prediction at parameter X. We can evaluate the posterior distribution of $y_{\ell+1}$ as

$$p(y_{\ell+1}|X) = \int p(y_{\ell+1}|y_\ell, X)\, p(y_\ell|X)\, \mathrm{d}y_\ell. \tag{19}$$

One can write the marginalization step after cascading as

$$p(y_{\ell+1}|X) = \int_{y_\ell}\int_{y_{\ell-1}} \cdots \int_{y_1} p(y_{\ell+1}|y_\ell, X)\, p(y_\ell|y_{\ell-1}X) \ldots p(y_2|y_1, X)\, p(y_1|X)\, \mathrm{d}y_\ell \mathrm{d}y_{\ell-1} \ldots \mathrm{d}y_1.$$

We can perform this integration using the Monte Carlo method. The prediction steps for three fidelity stacked NARGP surrogates are shown in figure 3(b).

Every layer in the stacked architecture is a GP. So, the samples drawn from GP will stay close to the posterior mean when the posterior predicted variance is small. If we ensure that there are enough data points in each layer such that the maximum of the posterior variance across the parameter space is smaller than a cut-off limit then we can use the posterior predicted mean of the previous layer as input for the next

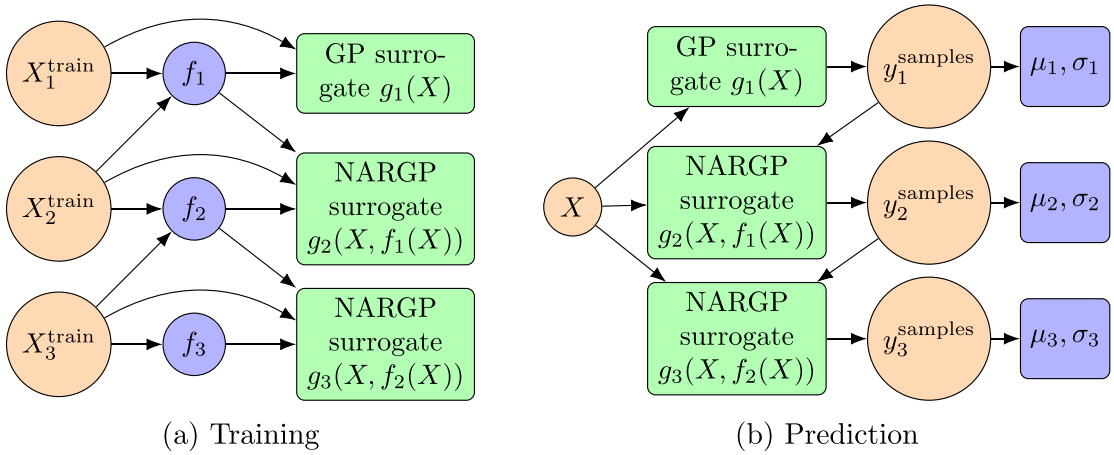


Figure 3. Flowchart showing the training and prediction steps of NARGP surrogate for a three-fidelity scenario. f_1, f_2, and f_3 represent the function at each fidelity. g_1, g_2, and g_3 represent the surrogate of the corresponding fidelity. Training the NARGP model for a particular level involves evaluating the model of that fidelity level and the previous fidelity level. We draw samples to make predictions for a particular level, which involves drawing samples from the surrogate of the previous level and feeding them as input to the surrogate of that level. We can then use the samples to obtain the required statistical moments of the predictions by marginalizing the samples.

layer. In this way, we can bypass the Monte Carlo step. An efficient way to ensure small posterior variance is by using the adaptivity algorithm, which we will discuss later.

- **Non-nested training data:** In situations with non-nested training data, an alternative is to employ the DGP.

3.1.3. Deep Gaussian processes

Drawing inspiration from the stacked architecture of multilayer neural networks, DGPs extend a single GP by composing multiple GP with each other. Each node within DGP represents a Gaussian process. The fundamental concept of deep architectures lies in their ability to model complex functions through combinations of simpler ones, with the output of one layer serving as the input for the next [40]. The primary advantage of DGPs over a single Gaussian Process (GP) lies in the distribution of feature learning across different layers. In a DGP, each layer is dedicated to learning distinct features of the model. This is in contrast to a single GP, where all features are consolidated into one, potentially leading to a complex and intricate model, especially when using traditional kernels.

DGP can effectively capture various aspects or representations of the underlying system by distributing the learning process across multiple layers. DGP emerges as a suitable candidate for modeling multi-fidelity scenarios. With each fidelity level, DGP gradually incorporates new features, enhancing the model's capabilities as it approaches high-fidelity representations. This progressive refinement allows DGP to effectively capture the intricacies of complex systems.

We model the transformation similar to equation (13) with one difference. We replace the actual function evaluation of the previous layer with the previous GP layer

$$f_\ell(X) = g_\ell(X, g_{\ell-1}(\cdots)). \tag{20}$$

The analytic form of the posterior of the GP surrogate g_ℓ is intractable because one of its parameters ($g_{\ell-1}$) is sampled from a posterior distribution. To sample from the posterior of g_ℓ, we use variational inference for each layer of DGP [41]. We use M_ℓ inducing points to approximate each layer representing a fidelity level. Typically, the number of inducing points is significantly smaller than the number of training data points ($M_\ell \ll N_\ell$). This decreases the computational requirements of GP training and prediction steps [42–44]. This approach is also known as the Sparse Variational Gaussian Process (SVGP) [45]. We discuss the modeling approach using DGP for multi-fidelity surrogate modeling in [26, 46].

Let us assume that we have a set of locations of the training parameters as

$$\mathbf{X} = \left\{X_1^{\text{train}}, X_2^{\text{train}}, \ldots, X_L^{\text{train}}\right\}$$

and the corresponding function evaluations

$$\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L\} = \left\{f_1\left(X_1^{\text{train}}\right), f_2\left(X_2^{\text{train}}\right), \ldots, f_L\left(X_L^{\text{train}}\right)\right\}.$$

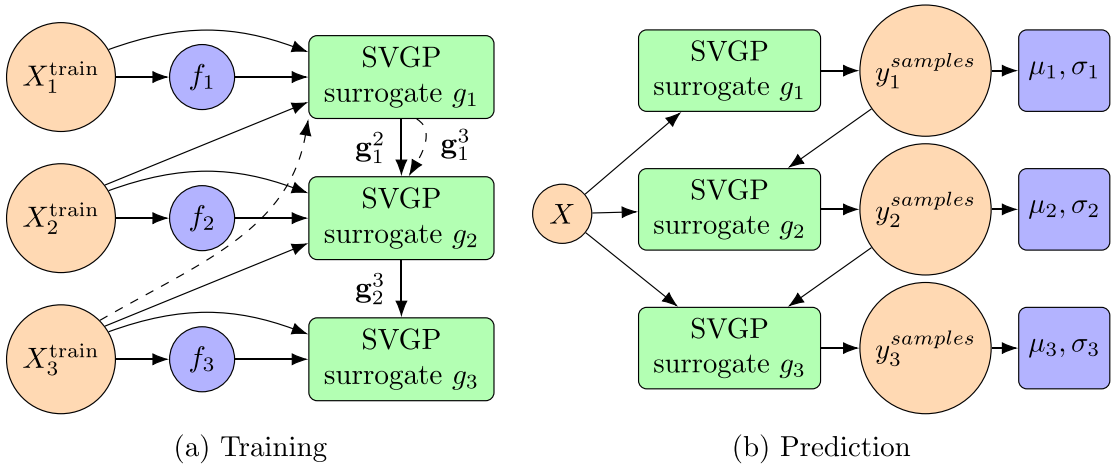


Figure 4. Flowchart showing the training and prediction steps of Deep Gaussian process surrogate for three fidelity scenarios. f_1, f_2, and f_3 represent the function at each fidelity. g_1, g_2, and g_3 represent the surrogate of the corresponding fidelity. Training the DGP model for a particular level involves evaluating the model of that fidelity level and the surrogate of the previous fidelity level, which further involves evaluating the one-level lower surrogate. This creates a cascading effect until we reach the lowest fidelity surrogate. The prediction process is similar to the full GP, which involves drawing and marginalizing samples.

Note that we do not impose the limitation that the location of the training parameters should be nested. Let $\mathbf{g}_\ell^t$ represent the samples drawn from the ℓth layer at the training parameter X_t^{train}. We assume that M_ℓ inducing points are sufficient to represent the distribution at layer ℓ. We represent the location of the inducing point by $\mathbf{Z}_\ell$. Let $\mathbf{u}_\ell$ be the sample drawn from the ℓth layer at the inducing points $\mathbf{Z}_\ell$. We represent the kernel function and the mean function of the prior at layer ℓ using k_ℓ and m_ℓ respectively. The unnormalized posterior distribution at ℓth layer is

$$p\left(\mathbf{y}_\ell, \mathbf{g}_\ell^\ell, \mathbf{u}_\ell | \mathbf{X}_\ell, \mathbf{g}_{\ell-1}^\ell, \mathbf{Z}_\ell\right) = \underbrace{p\left(\mathbf{y}_\ell | \mathbf{g}_\ell^\ell, \mathbf{X}_\ell\right)}_{\text{likelihood}} \underbrace{p\left(\mathbf{g}_\ell^\ell | \mathbf{u}_\ell, \mathbf{g}_{\ell-1}^\ell, \mathbf{X}_\ell, \mathbf{Z}_\ell\right) p\left(\mathbf{u}_\ell | \mathbf{Z}_\ell\right)}_{\text{DGP prior}}. \tag{21}$$

Figure 4(a) shows the training process for a three-fidelity scenario. The calculation of posterior at any layer needs samples from the previous layer at the training points, which in turn requires further evaluation from previous layers. For example, the calculation of the posterior of the third layer needs to draw samples from the second layer at X_3^{train}, which also requires samples from the first layer at X_3^{train}. We represent this hidden calculation with a dashed arrow in figure 4(a).

One can combine all the layers by multiplying the posterior from individual layers from equation (21) to obtain the joint posterior distribution for all layer $p(\mathbf{y}, \{\mathbf{g}_\ell^t, u_\ell\})$. Analytical inference from the posterior is intractable. To approximate the distribution, we assume that the prior $q(\mathbf{u}_\ell)$ of $\mathbf{u}_l$ is a Gaussian distribution with mean $\tilde{\mathbf{m}}_\ell$ and variance $\tilde{\mathbf{S}}_\ell$. Now, the approximated DGP prior is

$$q\left(\mathbf{g}_\ell^\ell, \mathbf{u}_\ell | \mathbf{X}_\ell, \mathbf{g}_{\ell-1}^\ell, \mathbf{Z}_\ell\right) = p\left(\mathbf{g}_\ell^\ell | \mathbf{u}_\ell, \mathbf{g}_{\ell-1}^\ell, \mathbf{X}_\ell, \mathbf{Z}_\ell\right) q\left(\mathbf{u}_\ell\right). \tag{22}$$

If $\mathbf{u}_\ell$ is marginalized out of the equation (22), then the distribution becomes a Gaussian distribution with the following mean and variance:

$$\mu_\ell(x) = m_\ell(x) + \alpha(x)^T\left(\tilde{\mathbf{m}}_\ell - m_\ell(x)\right), \tag{23}$$

$$\Sigma_\ell(x, x') = k_\ell(x, x') - \alpha(x)^T\left(k_\ell\left(\mathbf{Z}_\ell, \mathbf{Z}_\ell\right) - \tilde{\mathbf{S}}_\ell\right)\alpha(x'). \tag{24}$$

We can use this before approximating the posterior at each layer and then extend this to all the layers of DGP to obtain the approximate posterior distribution $q(\mathbf{y}, \{\mathbf{g}_\ell^\ell, \mathbf{u}_\ell\}_{\ell=1}^L)$. We want to approximate the exact posterior distribution in equation (21) by an approximate distribution $q(\mathbf{y}, \{\mathbf{g}_\ell^\ell, \mathbf{u}_\ell\}_{\ell=1}^L)$. We train the DGP by minimizing the Kullback-Leibler (KL) divergence KL$[q||p]$ between the variational posterior distribution q and the true posterior distribution p. An exact evaluation of the KL divergence is not feasible. So, we convert the minimization of the KL divergence to the maximization of Evidence Lower Bound (ELBO) ($\mathcal{L}$) whose evaluation is feasible. ELBO is evaluated as

$$\mathcal{L} = \sum_{\ell=1}^{L} \mathbb{E}_{q(\mathbf{g}_\ell^\ell)}[\log \underbrace{p\left(\mathbf{y}_\ell | \mathbf{g}_\ell^\ell\right)}_{\text{likelihood}}] - \text{KL}\left[q(\mathbf{u}_\ell) || p(\mathbf{u}_\ell | \mathbf{Z}_\ell)\right]. \tag{25}$$

One can refer to [46] for detailed proof. We optimize for the variational parameters ($\tilde{\mathbf{m}}_\ell$, $\tilde{\mathbf{S}}_\ell$, $\mathbf{Z}_\ell$) and the hyperparameters of the kernels (k_ℓ) using the Adam optimization method.

The prediction process in DGPs follows a recursive transfer of results from the previous layer into the next layer. We draw samples at each layer from the approximate distribution defined by equation (22). We fuse the prediction parameter X^* in intermediate layers with the sample drawn from the previous layer. We then use the fused parameter to draw a sample from that layer. We repeat this process until we have S samples for each layer. We use the collected samples to compute the posterior mean and variance through a Monte Carlo approximation. Figure 4(b) visually illustrates these prediction steps for three fidelity models, depicting the recursive transfer and sampling process at each layer.

One can improve the surrogate by choosing a more structured kernel. Cutajar *et al* [26] suggest using the kernel defined in equation (14). We call this method the Non-linear Auto-Regressive Deep Gaussian Process (NARDGP). We suggest using the delay term to further add useful information. We call the method Deep Gaussian Process with Delay Fusion (DGPDF). We further improve upon the suggested method using a structured kernel as mentioned in equation (18). We named the resulting method Deep Gaussian Process with Delay Fusion and Composite kernel (DGPDFC).

Methods discussed in sections 3.1.2 and 3.1.3 have similar ideas but different approaches to approximating the posterior distribution. In future discussions, we will sometimes mention the methods in section 3.1.2 as full GP to differentiate it from DGP because they do not involve sparse variational approximation of the posterior.

3.2. Adaptive model improvement

In many cases, the available training data may not provide sufficient information to accurately model predictions, resulting in poor performance. One approach to enhance the model is adding more data points to the training set. This comes with the added cost of evaluating the corresponding function at the newly added training data locations. Therefore, judiciously choosing where to add points is crucial. The goal is to ensure that we capture the essential features of the target functions using as few training data points as possible.

In this section, we discuss an adaptive algorithm [47]. It is designed to enhance training datasets applicable to any of the methods mentioned earlier, all of which utilize combinations of Gaussian processes. This adaptive algorithm improves the overall surrogate prediction by focusing on one Gaussian process at a time. Suppose we are training a GP using n data points $\mathcal{D}_n = \{X_1, X_2, \ldots, X_n\}$. Let y^{n+1} be the value of a sample from GP at X^{n+1}. We select the next point X_{n+1} where we gain maximum information which is defined as follows

$$\mathcal{I}(X_{n+1}|\mathcal{D}_n) = H(y^*|\mathcal{D}_n) - H(y^*|\mathcal{D}_n \cup X_{n+1}), \tag{26}$$

where $H(y^{n+1}|\mathcal{D}_n)$ is the conditional entropy. We suggest a greedy algorithm to selecting the next point X_{n+1} by solving the optimization problem

$$X_{n+1} = \underset{X_{n+1}}{\operatorname{argmax}} \mathcal{I}(X_{n+1}|\mathcal{D}_n). \tag{27}$$

Upon simplification, it can be shown that for GPs, the gain in information is directly proportional to the logarithm of the posterior predicted standard deviation. Thus, the optimization problem stated in equation (27) can be converted to

$$X_{n+1} = \underset{X^*}{\operatorname{argmax}} \sigma_p(X^*|\mathcal{D}_n), \tag{28}$$

where σ_p is the predicted standard deviation. We can keep adding new points for a particular fidelity level until the maximum posterior standard deviation is below a certain specified threshold level. After that, we can move to the next level until we cover all the fidelity levels.

There are certain drawbacks to this method. Firstly, this method adds one point at a time, limiting its overall speed. Secondly, the suggested points are always towards the boundary in high-dimensional space.

Various other approaches are proposed in the literature depending on alternative metrics to iteratively enhance the Gaussian Process surrogate [48]. One can improve the posterior standard deviation adaptivity metric by adjusting it via some error information as discussed in [49]. However, this requires function evaluation at additional validation points, making it infeasible for computationally expensive high-fidelity functions. Query-by-committee (QBC) [50, 51] is also computationally infeasible due to the need to train multiple models. Gradient-based adaptive approach [52] utilizes gradient information, which may be unavailable in many cases especially involving legacy solvers. Moreover, gradient approximation using the

Table 1. Benchmark problems for the academic examples.

Benchmark name	Low-fidelity function (f_l)	High-fidelity function (f_h)
Linear transformation	$\sin(8\pi x)$	$0.8\sin(8\pi x) + 0.3\sin(2\pi x)$
Non-linear transformation		$\sin^2(8\pi x)$
Phase-shifted oscillation		$\sin^2\left(8\pi x + \frac{\pi}{10}\right) + \cos(4\pi x)$

finite difference method is numerically unstable and computationally expensive for high-fidelity models [53]. There are still many other adaptivity algorithms [48, 54, 55]. We will leave the discussion and the effects of those algorithms on multi-fidelity GP as future works.

4. Numerical examples

In this segment, we evaluate the surrogate modeling techniques mentioned in section 3 using standard academic examples and two real-world problems, each presenting its distinct challenges. Subsequently, we conduct a comparative and critical analysis of the strengths and limitations of each approach in varied scenarios. The method outlined in section 3 was implemented in Python programming language using GPflow library [56]. This implementation is open source and can be accessed on GitHub[5].

4.1. Academic examples

Before applying the problem to real-world scenarios, we test different multi-fidelity surrogate modeling methods on some academic problems shown in table 1 and visualized in figure 5, each posing a different modeling challenge. Researchers addressed the benchmarking of the surrogate models and there exists a variety of them [57]. Still, our low-fidelity function for all the test cases is the same because we want to test the capability of different methods to learn the relationship between low and high-fidelity models. We deal with two fidelity cases and train the multi-fidelity GP surrogate using fifty low-fidelity and eight high-fidelity function evaluations. The high number of low-fidelity train data ensures low predictive variance for the low-fidelity Gaussian process. We use twenty-five and eight induction points in each level for all the cases involving DGPs. Additionally, we also train a GP on high-fidelity data without multi-fidelity features. We then compare the multi-fidelity methods to the single-fidelity GP surrogate.

After this initial training, we run the adaptivity algorithm for ten steps to add ten additional high-fidelity data points. To ensure the robustness of the adaptivity algorithm, we perform multiple runs with distinct seed values. We take the average of the mean square error (MSE) and plot it against the corresponding number of high-fidelity function evaluations shown in figure 6. The resulting plot visualizes the rate at which the respective algorithm convergences to the target high-fidelity curve. We also summarize the average MSE in table 2.

We can observe from table 2 that across all scenarios, the multi-fidelity methods outperform the single-fidelity GP by a considerable margin. This is because the multi-fidelity methods take advantage of the additional information the low-fidelity models provide. This shows the advantage of using multi-fidelity GP over single-fidelity GP. One can also conclude from table 2 that methods involving the DGP have higher MSE as compared to the corresponding non-linear autoregressive methods with full GP, specifically when the number of high-fidelity function evaluations is increased. This is because DGP involves the approximation of the posterior distribution using some finite induction points.

We observe from table 2 that the AR1 surrogate demonstrates the lowest MSE compared to the other methods in a linear transformation problem. The high-fidelity function of the linear scaling problem is written as constant scaling of the low-fidelity function with a non-linear additive term. This aligns with the assumed formulation of AR1 as described in equation (10), giving AR1 an edge over the other methods. Every other non-linear method except DGPDF also has a low MSE compared to the single-fidelity GP. The non-linear transformation methods involve expanding the surrogate dimensionality by introducing additional dimensions for low-fidelity function evaluations and/or delay terms. The assumed non-linear transformation of the low-fidelity term and the additional dimension of the delay term do not contribute meaningful information to the surrogate model in the case of the linear scaling problem. The additional kernel hyperparameters and the increase in dimension might require more evaluation points to represent the function. Therefore, non-linear auto-regressive methods underperform when compared to the AR1 method.

[5] https://github.com/KislayaRavi/MuDaFuGP.

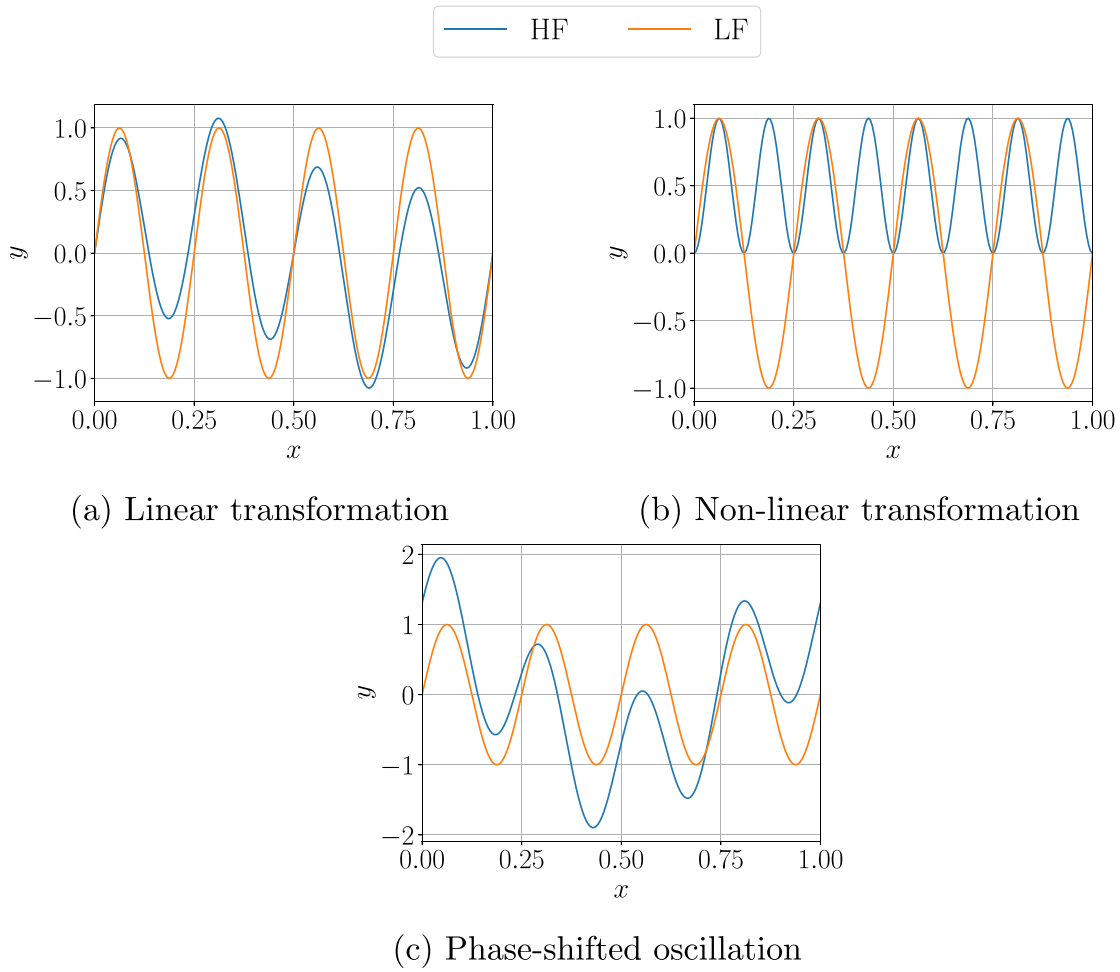


Figure 5. Visualization of different academic problems. The low and high-fidelity functions are visualized in each sub-plot by orange and blue curves, respectively.

Table 2 shows that the AR1 surrogate cannot accurately capture non-linear transformations as expected because of the underlying linear formulation. In contrast, other methods tailored for non-linear transformations exhibit effective predictions with low MSE.

In the next example, we discuss the case where the high-fidelity function (f_h) and the low-fidelity function (f_l) are oscillating functions with phase differences. We observe such cases in our brain where neurons oscillate with certain frequencies where one uses the Hodgekin-Huxley model of them as a surrogate for real data [58].

The high-fidelity function can be expressed as a combination of sine and cosine terms, with the cosine term representing the derivative of the low-fidelity function. Consequently, methods incorporating delay terms are anticipated to outperform others

$$\begin{aligned} f_h(x) &= \sin(8\pi x)\cos\left(\frac{\pi}{10}\right) + \cos(8\pi x)\sin\left(\frac{\pi}{10}\right) + \cos(4\pi x) \\ &= \cos\left(\frac{\pi}{10}\right) f_l(x) + \sin\left(\frac{\pi}{10}\right)\frac{df_l}{dx} + \cos(4\pi x). \end{aligned}$$

NARGP and NARDGP exhibited less accurate fitting to the target high-fidelity curve, as their formulations lack derivatives directly applicable to this example. Conversely, methods incorporating delay terms in surrogate modeling demonstrated accurate predictions by mimicking the derivative of the low-fidelity function. Surprisingly, AR1 also predicts accurately because the phase-shifted oscillation gets simplified into a linear scaling problem which is suitable for AR1.

DGPDF specifically underperforms for linear-scaling problems and phase-shifted oscillations even when other non-linear autoregressive methods had low MSE. Only one kernel is responsible for capturing all the features of the target high-fidelity function. In other non-linear autoregressive methods, the kernel has a

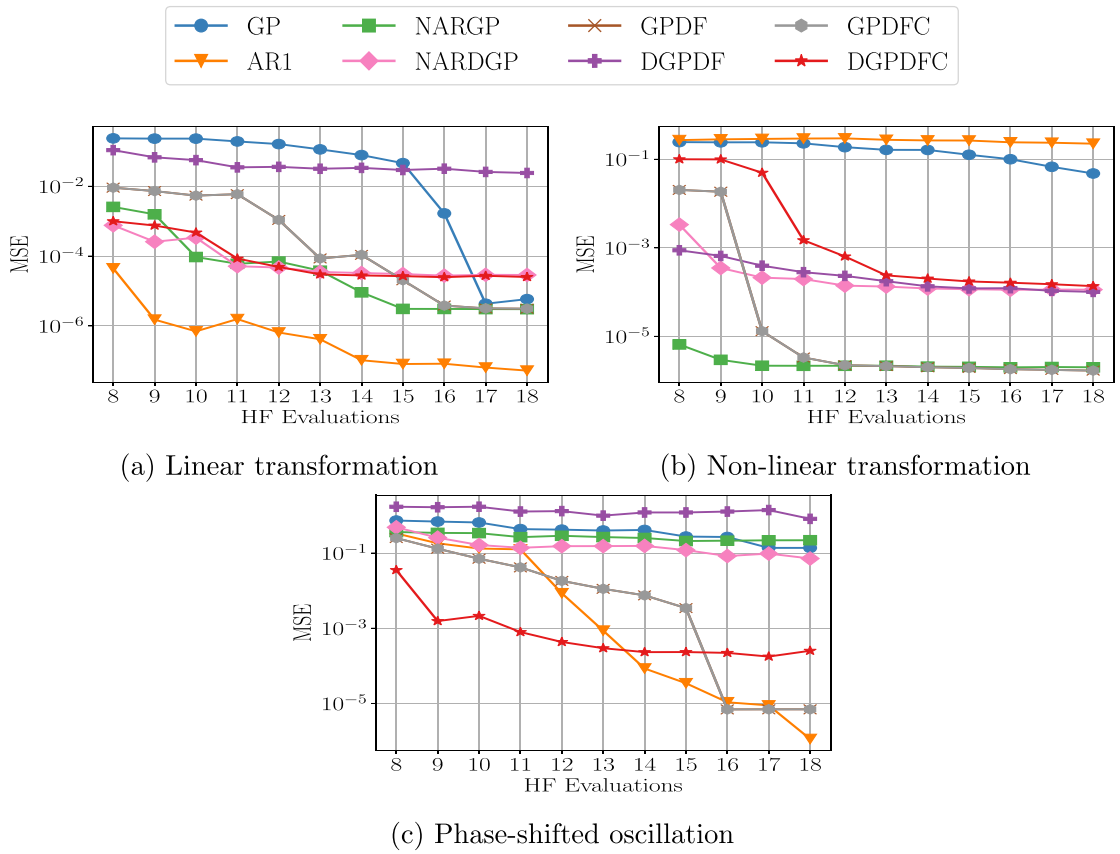


Figure 6. Average MSE evolution with respect to the number of high-fidelity function evaluations for different academic problems discussed in section 4.1.

Table 2. Average value of MSE of each method in three different transformation scenarios, namely linear scaling, non-linear, and phase-shifted transformation as shown in table 1, w.r.t. number of high-fidelity data points used in training. The method with minimum MSE is shown in bold.

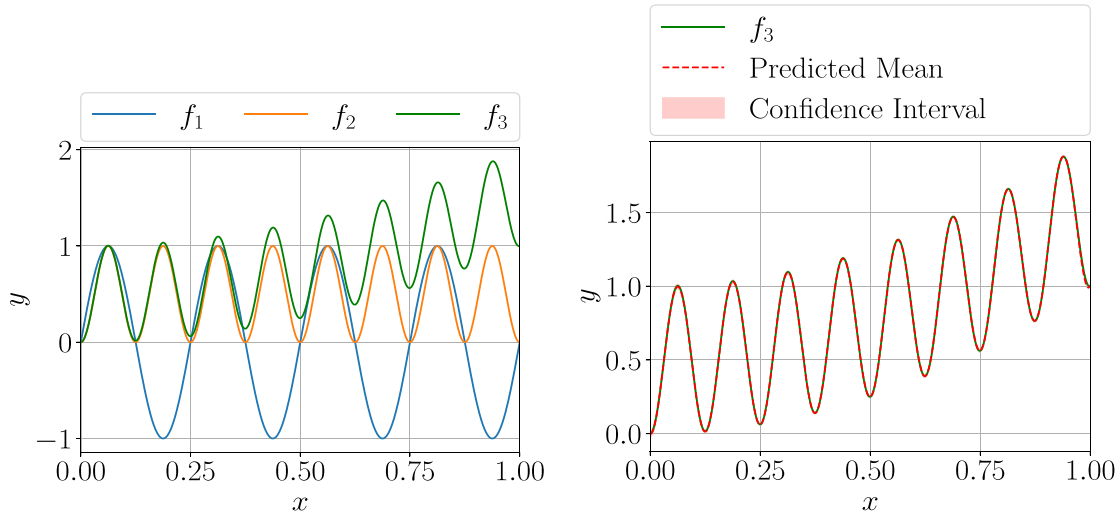
	Type of transformation					
	Linear scaling		Non-linear transformation		Phase-shifted oscillation	
MF method	8 HF points	18 HF points	8 HF points	18 HF points	8 HF points	18 HF points
GP	2.36×10^{-1}	5.85×10^{-6}	2.45×10^{-1}	4.79×10^{-2}	7.32×10^{-1}	1.37×10^{-1}
AR1	$\mathbf{4.46 \times 10^{-5}}$	$\mathbf{5.24 \times 10^{-8}}$	3.73×10^{-1}	2.71×10^{-1}	2.25×10^{-1}	$\mathbf{1.14 \times 10^{-6}}$
NARGP	2.57×10^{-3}	3.02×10^{-6}	$\mathbf{6.57 \times 10^{-6}}$	2.00×10^{-6}	3.62×10^{-1}	2.19×10^{-1}
GPDF	9.15×10^{-3}	3.11×10^{-6}	2.04×10^{-2}	$\mathbf{1.68 \times 10^{-6}}$	2.50×10^{-1}	6.99×10^{-6}
GPDFC	9.15×10^{-3}	3.11×10^{-6}	2.04×10^{-2}	$\mathbf{1.68 \times 10^{-6}}$	2.50×10^{-1}	6.99×10^{-6}
NARDGP	7.63×10^{-4}	2.88×10^{-5}	3.34×10^{-2}	1.14×10^{-4}	4.84×10^{-2}	7.16×10^{-2}
DGPDF	1.09×10^{-1}	2.41×10^{-2}	8.81×10^{-4}	1.01×10^{-4}	17.0×10^{-1}	8.13×10^{-1}
DGPDFC	9.99×10^{-4}	2.52×10^{-5}	1.00×10^{-1}	1.37×10^{-4}	$\mathbf{3.53 \times 10^{-2}}$	2.52×10^{-4}

well-defined structure. This underscores the significance of a well-defined kernel structure specifically for DGP.

The three presented cases with different transformations underscore the variability in the performance of different methods across diverse scenarios. This observation emphasizes the necessity of a judicious selection of methods based on the specific characteristics of the modeled system. Prior knowledge about the system can significantly contribute to choosing the most suitable methodology.

As indicated in section 3, we can extend the surrogate modeling to encompass more than two fidelities. As an example, we consider a three-fidelity case defined as

$$\begin{aligned} f_1(x) &= \sin(8\pi x) \\ f_2(x) &= \sin^2(8\pi x) \\ f_3(x) &= \sin^2(8\pi x) + x^2. \end{aligned} \tag{29}$$



(a) Target functions. (b) Model prediction for f_3 using NARGP.

Figure 7. An example of multi-fidelity gaussian process surrogates for three fidelity case. We use NARGP to create surrogate because the transformation from first to the second fidelity is non-linear.

Figure 7(a) shows the target function for the three fidelity cases as described in equation (29). The transition from the first fidelity to the second fidelity is non-linear, necessitating the use of a non-linear method to construct a surrogate. In this example, we present the results achieved using NARGP. We begin with forty, twelve, and eight randomly sampled points for each fidelity level and then execute the adaptivity algorithm for five steps. We have already observed from the results of two-fidelity cases that the posterior variance for the second fidelity under the same scenario using the NARGP model is very small. Consequently, we can utilize the posterior mean of the surrogate of the second level as the input for the third layer. Figure 7(b) shows that the surrogate aligns closely with the target function. Analogous to the two-fidelity cases, familiarity with the physical model will guide our selection of the appropriate method.

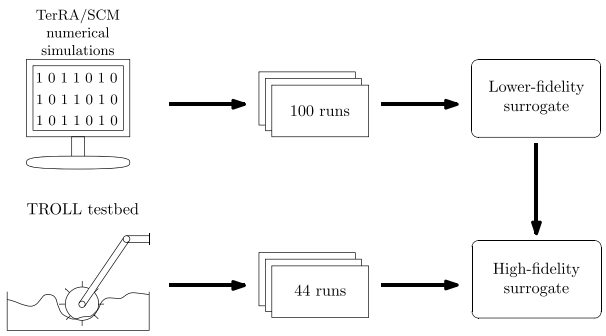
4.2. Terramechanical example

Terramechanics explores the interaction of wheels with the underlying soil [59]. There are no exact physical formulas that describe the process of the wheel-soil interaction because of the non-linearity of the pressure-sinkage relations in the soft soil [60]. Therefore, to simulate wheel movement on soft sands and to estimate forces and torques acting on that wheel, we exploit a wide range of numerical simulations, each one with a different level of discretization, assumptions, accuracy, and computational time. This makes terramechanics a good example of multi-fidelity modeling because different numerical models are dedicated to simulating the same physical process, but with different levels of fidelity.

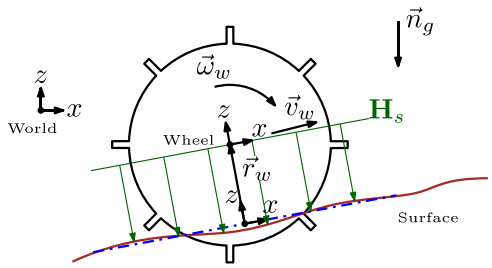
As data sources for lower-fidelity models for this experiment, we used two types of models, both of which were developed at the Terramechanical lab at German Aerospace Center (DLR): TerRA [62] and SCM [60]. As a high-fidelity data source, DLR's testbed TROLL was used [63]. This testbed was initially created to test and validate simulations for the wheel-soil interactions for extraterrestrial rover projects like MMX [64] and Scout [65]. The 44 high-fidelity runs from TROLL were also replicated in the simulation environment to achieve both higher- and lower-fidelity observations on the same set of inputs[6]. These runs included a variety of different steering and titling angles, different rotation and horizontal velocities, and different movement trajectories and accelerations. Movement scenarios for dataset creation were described and justified in the previous paper [61]. The scheme of the overall setup of experiments is depicted in figure 8(a).

Lower fidelity simulations are iterative simulations with each next step dependent on the state of the previous step, and the output of the simulation depends on several technical variables, which do not have a clear physical meaning and were not included as the input features in the dataset. The Gaussian process surrogate takes only a subset of the input variables of the simulation into account. Surrogate GPs were trained to approximate the outputs of TerRA and SCM simulations, and 100 runs were used in both cases. Each run was 20 s long and with a sampling rate of 3 points per second. Runs were conducted with randomly selected surfaces, random sequences of acceleration and deceleration and random sequences of steering

[6] The data is not publicly available but is available upon request.



(a) Setup of the terramechanical experiment. TROLL testbed was used to create 44 high-fidelity runs. Numerical simulations were used to create 100 lower-fidelity runs and train a lower-fidelity surrogate model.



(b) Representation of the input data used in the ML model. $\vec{r}_w$ is a wheel position w.r.t the surface; $\vec{v}_w$ is wheel velocity, $\vec{\omega}_w$ is the angular velocity of the wheel, $\vec{n}_g$ stands for gravity direction, and $\vec{f}_w$ depicts traction force acting on the wheel, our target value we want to predict [61].

Figure 8. Different data generators were simulating the wheel-soil interaction for the same scenarios, thus creating an ideal case for multi-fidelity modeling. Reproduced from permission from [61]. © International Society for Terrain-Vehicle Systems.

Table 3. Description of the variables from the terramechanical dataset.

Variable	Specification	Description
$\vec{r}_w$	3D Input	Wheel position w.r.t the surface
$\vec{v}_w$	3D Input	Translational velocity of the wheel
$\vec{\omega}_w$	3D Input	Angular velocity of the wheel
$\vec{n}_g$	3D Input	Gravity acting on the wheel
f	1D Output	Traction force acting on the wheel

commands. The randomness of a surface is introduced by varying its profile. The profile for each run has 6 breaking points, where after each point the change is introduced. Changes are normally distributed with a mean of 0.1 m and a variance of 0.6 m and are gradually introduced at each breakpoint. Velocities of the wheel simulation are also distributed normally, with a mean of 1 m s^{-1} and variance of 2 m s^{-1}. Steering angle is distributed normally, with 0 mean and 0.5 radians variance. Both velocities and steering angle were changing each second of the simulation.

We split high-fidelity runs into 29 runs used for training purposes and 15 reserved for the test dataset. We varied sampling frequency from the high-fidelity data, to verify how performance changes with changes in the amount of high-fidelity data. Harvesting high-fidelity data is challenging and we want to minimize the damage to the model's performance, therefore we want to analyze the degradation of the prediction accuracy with the decrease in the number of training points. Each next subset is nested into the subset with a bigger number of training points. This logic assures the consistency of training points in all down-sampling examples.

All signals were smoothed with a low-pass filter, as in reality spikes of traction force are considered to be noise because only constant application of a force can change the wheel's movement. The data consists of 12-dimensional input and the objective function is a traction force acting on the wheel. A description of the variables is detailed in table 3. A visualization of the wheel and features describing it is depicted in figure 8(b).

We tested all MF models discussed earlier on the terramechanical data. MF models shown here were trained with both SCM and TerRA surrogates being the lower-fidelity function and compared with each other. Table 4 shows the performance of different MF methods w.r.t. the number of high-fidelity data points used during the modeling and numerical simulation which generated the data for the lower-fidelity level. The performance of the MF models with the TerRA as a lower-fidelity data source shows marginally worse results than with SCM, however still very compatible. This is a positive result, given that the TerRA model is much simpler in implementation and less expensive in exploitation. Changing the number of high-fidelity points can give a hint of the lower acceptable bar when we construct the MF model.

Models comparison with different subsets is illustrated in table 4. As we can see there, the multi-fidelity approach can handle the drastic decrease in the number of used high-fidelity points, while the single-fidelity model's performance decreases, as expected. Several examples of best-performing methods, NARDGP, are shown in figure 9.

Table 4. Normalized MSE of each MF method for wheel-soil locomotion simulation, w.r.t. number of high-fidelity data points used in training. Simulations using SCM were used as a lower-fidelity layer. The normalized MSE of conventional numerical simulation models is presented for comparison. GP trained only on the high-fidelity points shows good results when we have a lot of data points, but its performance deteriorates with a decrease in the training points. While the MF method, especially NARDGP, can show good results with fewer high-fidelity points and has more consistent performance. The method with minimum MSE is shown in bold.

	Number of HF points									
	2085		1042		521		208		104	
	Numerical simulation used as a lower fidelity layer									
Method Name	TerRA	SCM	TerRA	SCM	TerRA	SCM	TerRA	SCM	TerRA	SCM
Single fidelity GP	0.149		0.239		0.331		0.32		0.756	
AR1	0.235	0.235	0.235	0.235	0.235	0.235	0.235	0.235	0.235	0.235
NARGP	0.184	0.211	0.2	0.23	0.218	0.726	0.251	0.256	0.724	0.726
GPDF	0.41	0.207	0.327	0.231	0.3	0.218	0.25	0.233	0.724	0.724
GPDFC	0.41	0.207	0.327	0.231	0.3	0.218	0.25	0.233	0.724	0.724
NARDGP	**0.116**	**0.122**	**0.12**	**0.118**	**0.123**	**0.12**	**0.118**	**0.117**	**0.121**	**0.125**
DGPDF	0.726	0.726	0.726	0.726	0.726	0.726	0.726	0.726	0.726	0.726
DGPDFC	0.217	0.22	0.249	0.233	0.224	0.223	0.226	0.23	0.22	0.235
Conventional numerical simulations										
TerRA	799.92									
SCM	0.209									

Empirically, there is no advantage in building more than two layers of fidelity for this regression task, as SCM simulation covers all the abilities of TerRA simulation and gives much more reliable results [60]. We conducted a comparison of the two-level NARDGP model performance where SCM data serves as a lower-fidelity and TROLL as the high-fidelity data source and three-level NARDGP, with new lower-fidelity level from TerRA data, so that SCM simulations become medium-level. For both cases, we used 43 high-fidelity data points from TROLL experiments, as we are first of all interested in the model's performance with as few high-fidelity points as possible, and 1500 data points from SCM simulation. For the three-level model, we sampled 3000 data points from the TerRA simulation. Two of the examples are depicted in figure 10 and they show a very limited difference between the performance of the two approaches. Given the limited scale of improvements we did not conduct a full analysis as in table 4, but we will leave this for further work.

One of the main advantages of MF for complex systems modeling is that it can decrease computational and time costs without decreasing prediction accuracy. MF ML model outperforms both conventional numerical simulations TerRA and SCM simultaneously by MSE and by the runtime. Figure 11 shows 44 runs for both the MF ML model and two baseline numerical simulations (TerRA and SCM), distributed by their MSE and runtime. As we can see, the performance of the MF ML model is indeed faster and more accurate than the conventional terramechanical models.

Another advantage of the Gaussian process is the implicit handling of uncertainties. Due to its Bayesian nature, we can propagate uncertainties easily using the equation (5). It is an important part of predicting complex simulation systems, both for operational and research purposes. This adds more depth to the understanding and explainability of predictions and makes ML black boxes more transparent.

As we can see in figure 9, the main disadvantage of the NARDGP multi-fidelity prediction is that confidence intervals are too narrow, as discussed in section 2.3. This is a critical flaw when it comes to deploying ML models in practice, especially for highly sensitive operations like extraterrestrial rovers, where the cost of error is very high due to the inability to quickly compensate for the wrong move or repair the rover. To solve this problem, we applied several calibration methods, mentioned in section 2.3. Isotonic regression is an algorithm enabling quantile calibration, inspired by a Platt scaling algorithm from classification [37]. Beta calibration was chosen as an algorithm for achieving distribution calibration [36]. Normal calibration [35] introduces a scaling parameter for the predicted variance, preserving a Gaussian nature of the posterior prediction. A comparison of all calibration techniques on all of the test runs can be seen in table 5. As an example, we took NARDGP trained with 41 high-fidelity data points and calibrated it with the calibration mentioned above techniques. From this table, we can see that isotonic regression performs the best calibration, despite that it is only a global calibration of the marginal distribution. This means that the NARDGP model was constantly making overconfident predictions and global scaling of the variance solves the problem. These findings are consistent with the previous study, researching Gaussian process regression calibration for terramechanical data, but conducted only in the context of single-level medium-fidelity SCM data [68]. An example of a calibration on one test run could be seen in figure 12,

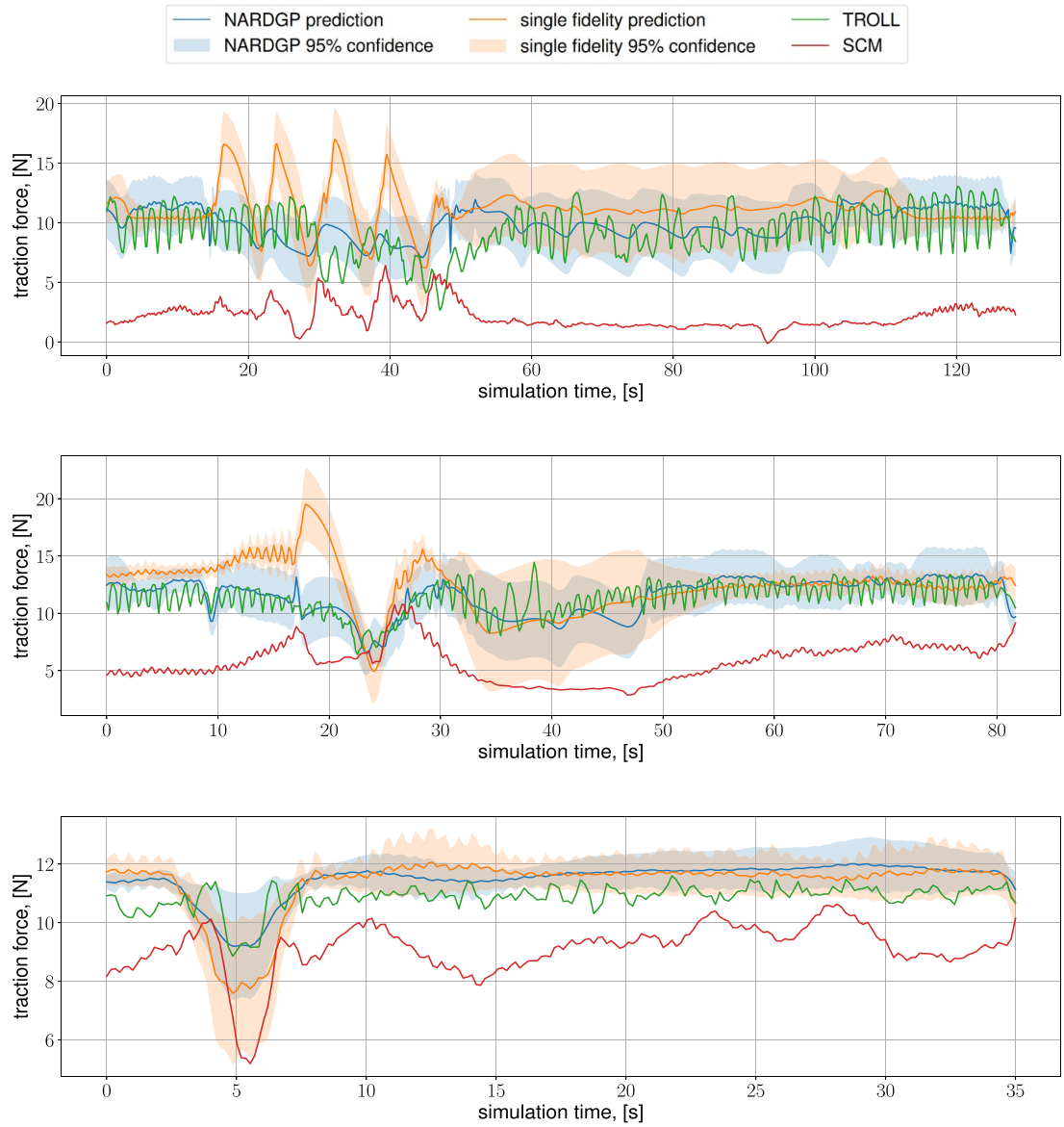


Figure 9. Visual comparison of the best performing MF ML model, NARDGP, on several test runs. It is compared against the ground truth from the TROLL sensors, single-fidelity model and the most popular numerical simulation SCM. The overall performance is reasonably better compared to other methods, but the model is visibly over-confident in its uncertainty estimation.

where prediction became less overconfident and 95% internal captures almost the entire actual observed traction force signal.

4.3. Plasma-physics example

Harnessing energy from plasma fusion promises to offer a clean alternative energy source. To achieve this goal, creating a self-sustained burning plasma is essential. Achieving a sustained burning plasma remains elusive due to physical and technological challenges. One prominent physical hurdle is the occurrence of small-scale fluctuations in confined plasma, which lead to energy loss. These small-scale fluctuations are referred to as microturbulence. Mitigating energy losses caused by microturbulence is a significant challenge within the plasma physics community.

This work simulates plasma fusion using the Automated System for TRansport Analysis (ASTRA) [69, 70] modeling suite. ASTRA solves four 1-dimensional transport equations for electron density (n_e), electron temperature (T_i), ion temperature (T_i) and poloidal flux (Ψ). Interested readers can refer to [71] for details. It uses different turbulent solvers to predict the density and temperature profiles given some initial conditions (initial profiles of T_i, T_e, and n_e) and the last-closed-flux surface. The calculation of the turbulent flux is one of the computationally expensive parts of the code. One can couple ASTRA with different turbulent solvers. In this work, we will use QuaLiKiz (QLK) [72] and QuaLiKiz Neural Network (QLKNN) [73, 74] as the two subroutines. QLK is a quasi-linear turbulence solver for the linearized

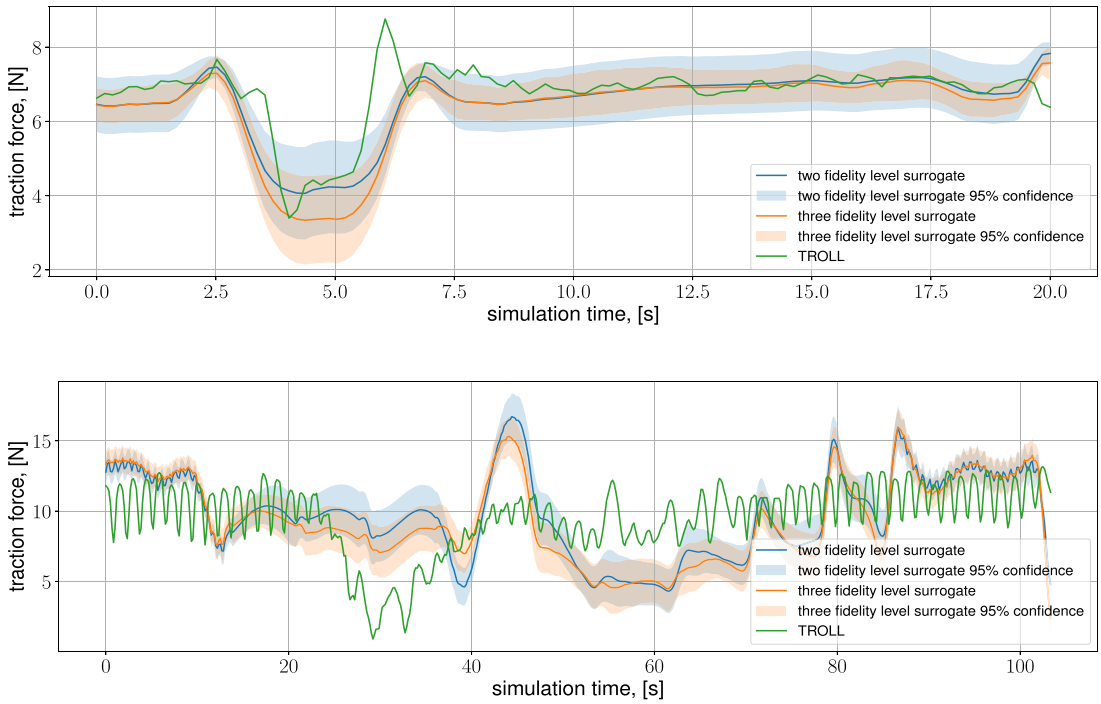


Figure 10. Visual comparison of the two- and three-level NARDGP surrogate performance against the test runs from the high-fidelity TROLL experiments. The results are not identical, but the performance increase is not promising, therefore we put aside proper testing of three-level MF ML surrogates for terramechanical data.

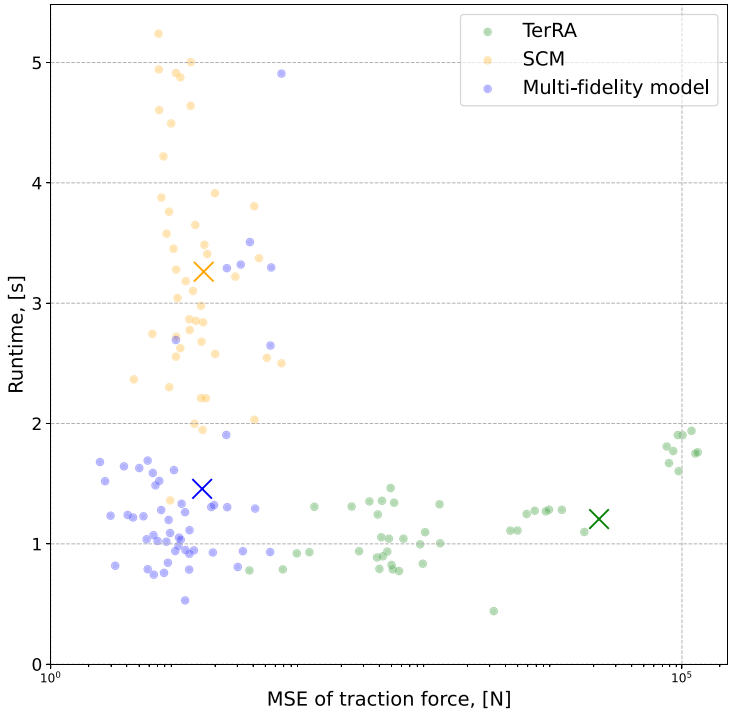


Figure 11. Mean squared error comparison of predictions from the conventional numerical simulation methods and MF ML model. The multi-fidelity approach is computationally efficient while maintaining a high level of precision.

gyrokinetic Vlasov equation. QLKNN is a neural network surrogate trained on a 10-dimensional Latin hypercube for a quick flux evaluation [74]. ASTRA simulation with QLK is the high-fidelity model, whereas ASTRA with QLKNN is the low-fidelity model.

We simulate a plasma discharge until it reaches a steady state. At the steady state, the heat and particle fluxes match the integrated sources at all the radial locations. In this work we simulate shot #34954 until it

Table 5. Mean pinball loss measures the errors of over- and under-confidence for each quantile individually [66]. Negative loglikelihood (NLL) measures how likely observed data was generated by the model [36]. Expected normalized calibration error (ENCE) orders and splits instances by variance into bins and calculates the scaled difference between the predicted error and variance in each bin [38]. Mean predictive interval width (MPIW) measures the sharpness of the prediction and is defined for each confidence interval α as a width of this interval [67]. Normal and isotonic calibration have similar results almost by every metric, but normal calibration has a significantly worse sharpness metric, which leaves us only with Isotonic regression. The method with minimum error is shown in bold.

	Calibration metrics			
Calibration method	Pinball	NLL	ENCE	MPIW
Uncalibrated	0.696	6.648	2.232	**0.093**
Isotonic regression	**0.424**	**1.932**	0.237	0.422
Normal calibration	**0.424**	2.229	**0.21**	175.675
Beta calibration	0.484	2.74	0.62	0.187

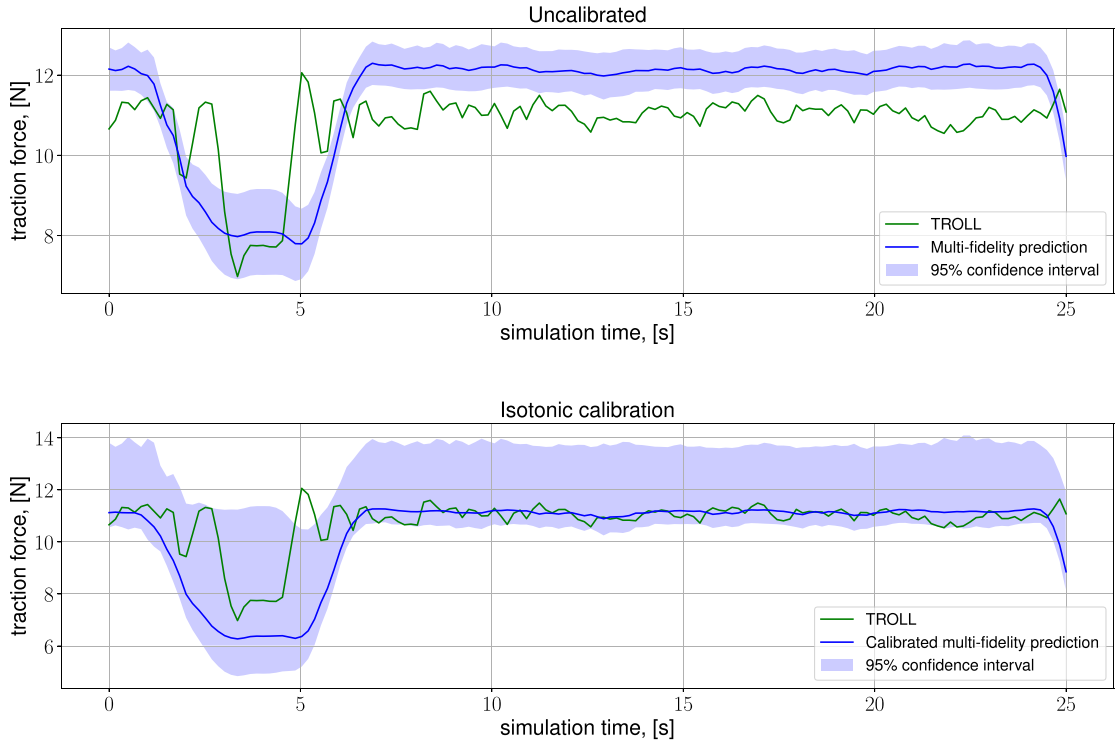


Figure 12. Original uncalibrated prediction is visibly overconfident because the ground truth traction force from TROLL quite often finds itself outside even the 95% confidence interval. By applying Isotonic regression, we achieve quantile calibrated prediction and here predicted confidence intervals capturing ground truth much better.

Table 6. List of parameters for plasma microturbulence surrogate modeling.

Parameter Name	Range
Radial distance (ρ_r)	$[0.1, 0.8]$
Initial T_i scaling	$[0.9, 1.1]$
Initial T_e scaling	$[0.9, 1.1]$
Initial n_e scaling	$[0.9, 1.1]$
Toroidal velocity scaling	$[0.9, 1.1]$
Safety factor scaling	$[0.9, 1.1]$

reaches a steady state. We train the surrogate on a six-dimensional space mentioned in table 6. We create six surrogates for six different quantities, namely, steady-state T_i, T_e, and n_e, and the corresponding negative derivative of the logarithm of each quantity ($-\nabla T_i/T_i$, $-\nabla T_e/T_e$, and $-\nabla n_e/n_e$). We normalize each quantity before training the surrogate. We use 40 high-fidelity and 400 low-fidelity training points.

Figure 13 shows the simulation results for QLK (high-fidelity model) and QLKNN (low-fidelity model) when parameters other than radial distance (ρ_r) are set as 1. The figure also shows the predicted results for multi-fidelity surrogates using NARGP. Table 7 shows the spatial average of mean square error for each surrogate.

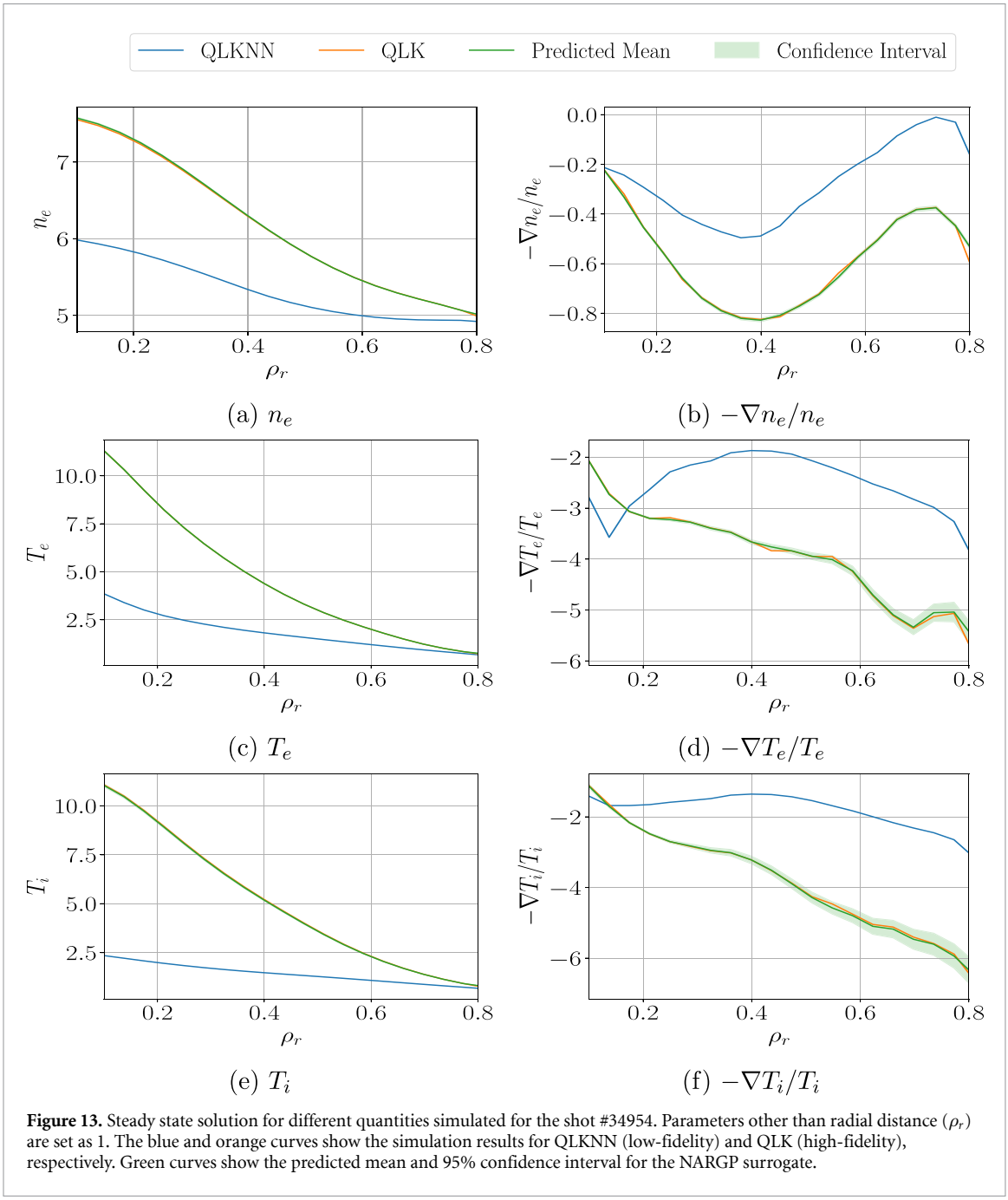


Figure 13. Steady state solution for different quantities simulated for the shot #34954. Parameters other than radial distance (ρ_r) are set as 1. The blue and orange curves show the simulation results for QLKNN (low-fidelity) and QLK (high-fidelity), respectively. Green curves show the predicted mean and 95% confidence interval for the NARGP surrogate.

Table 7. Spatial average value of MSE for different surrogates across different quantities for the plasma microturbulence surrogate modelling. The method with minimum MSE is shown in bold.

Method Name	Quantity of Interest					
	T_i	T_e	n_e	$-\nabla T_i/T_i$	$-\nabla T_e/T_e$	$-\nabla n_e/n_e$
Single-fidelity GP	8.57×10^{-3}	8.68×10^{-4}	1.07×10^{-2}	2.26×10^{-1}	1.04×10^{-1}	7.83×10^{-2}
AR1	1.73×10^{-2}	2.26×10^{-3}	3.31×10^{-2}	4.89×10^{-1}	8.21×10^{-2}	1.33
NARGP	4.46×10^{-3}	$\mathbf{8.13 \times 10^{-4}}$	7.12×10^{-3}	1.39×10^{-1}	$\mathbf{6.9 \times 10^{-2}}$	$\mathbf{9.57 \times 10^{-3}}$
GPDF	3.07×10^{-3}	$\mathbf{8.13 \times 10^{-4}}$	6.3×10^{-3}	1.94×10^{-1}	$\mathbf{6.9 \times 10^{-2}}$	1.44×10^{-2}
GPDFC	3.07×10^{-3}	$\mathbf{8.13 \times 10^{-4}}$	6.3×10^{-3}	1.94×10^{-1}	$\mathbf{6.9 \times 10^{-2}}$	1.44×10^{-2}
NARDGP	2.89×10^{-3}	8.56×10^{-4}	6.15×10^{-3}	4.41×10^{-1}	1.65×10^{-1}	1.72×10^{-1}
DGPDF	4.52×10^{-3}	8.58×10^{-4}	$\mathbf{5.42 \times 10^{-3}}$	1.83×10^{-1}	8.06×10^{-2}	3.57
DGPDFC	$\mathbf{2.88 \times 10^{-3}}$	8.58×10^{-4}	$\mathbf{5.42 \times 10^{-3}}$	$\mathbf{1.38 \times 10^{-1}}$	1.82×10^{-1}	4.37×10^{-2}

We observe that non-linear auto-regressive multi-fidelity surrogates perform better than the single-fidelity GP. However, the difference between single-fidelity GP and multi-fidelity surrogates is not too

significant. This may be due to the underlying simple structure that single-fidelity GP was able to learn relatively easily. AR1 struggles to model the quantity $-\nabla n_e/n_e$, which suggests that the relation between low-fidelity and high-fidelity could be non-linear for the case of $-\nabla n_e/n_e$. Moreover, DGPDF also has a high MSE for $-\nabla n_e/n_e$, whereas DGPDFC fits the quantity accurately. This re-iterates the significance of structured kernels, especially for DGPs.

5. Conclusions

This paper describes different multi-fidelity Gaussian process surrogate modeling methods. We extend non-linear autoregressive models for full GP to accommodate cases that involve more than two fidelities. Using a structured kernel with delay terms, we also suggest a new family of multi-fidelity GP models (GPDFC and DGPDFC). Finally, we test all the modeling methods on different academic and real-world problems.

We observe that not all the multi-fidelity methods performed well under all the scenarios. The quality of prediction depends upon the underlying relationship between the low and high-fidelity models. Prior knowledge about that would help us choose the correct modeling method. In many scenarios, that relationship is not known. Therefore, after constructing the surrogate, one must carefully validate its predictions. We observe that the structured kernel significantly improves models involving DGPs.

We also conclude that the multi-fidelity simulations can preserve the performance of higher-fidelity simulations but reach the speed and cost of low-fidelity ones. This can aid different research fields in analyzing the underlying system or improving the algorithm using outer loop methods [7].

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgments

Research by Vladyslav Fediukov, Michael Bergmann and Kislaya Ravi is funded by the Helmholtz Association under the ‘Munich School for Data Science—MuDS’(HIDSS-006). Felix Dietrich acknowledges funding by the DFG Project No. 468830823, and also association to DFG-SPP-229.

Appendix

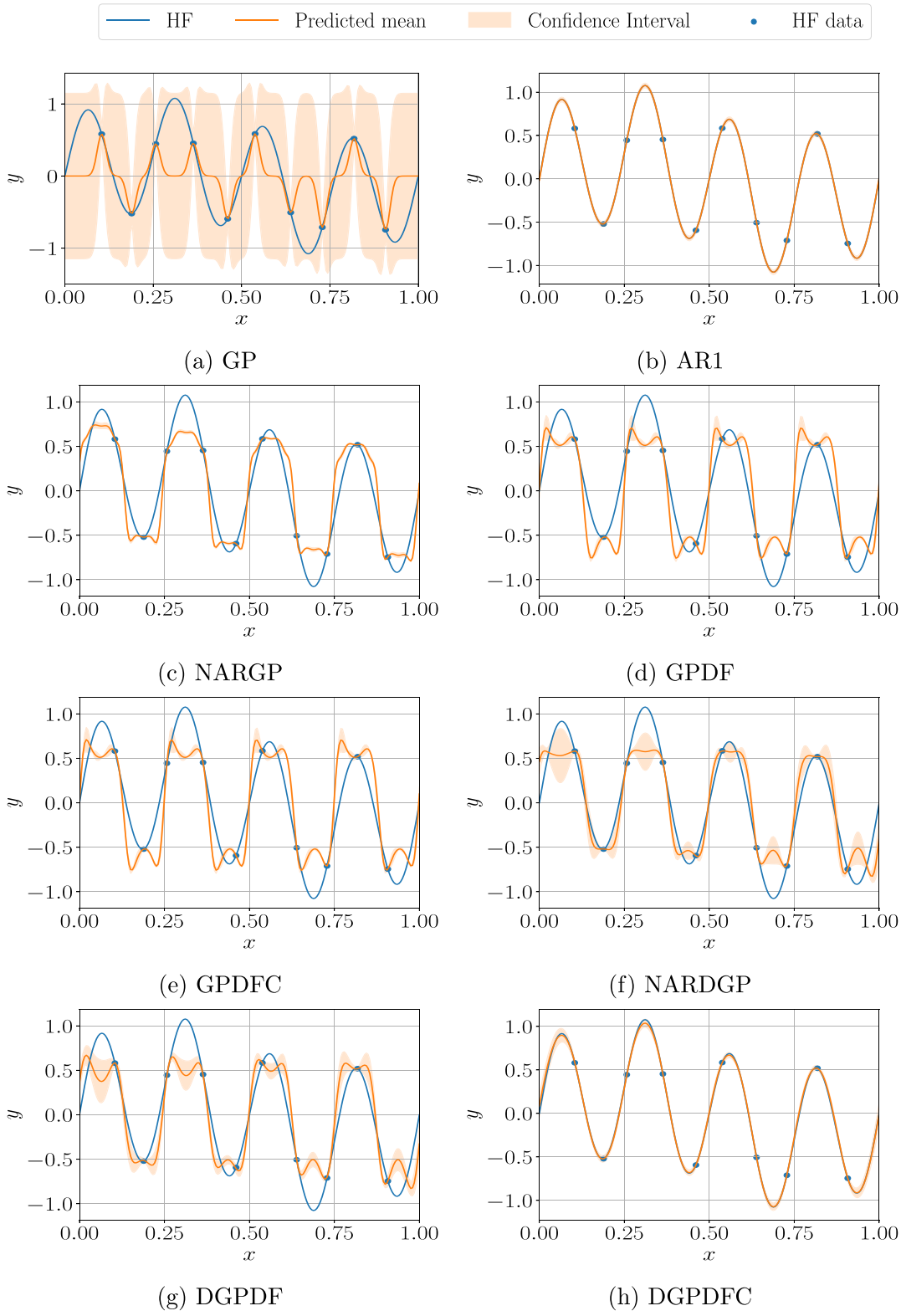


Figure A1. Comparison of different multi-fidelity Gaussian Process surrogate modeling methods on a linear transformation problem stated in table 1. The plots show the high-fidelity function in the blue curve, the high-fidelity training data in blue dots, the predicted mean in the orange curve, and 95% confidence interval using the orange-shaded region.

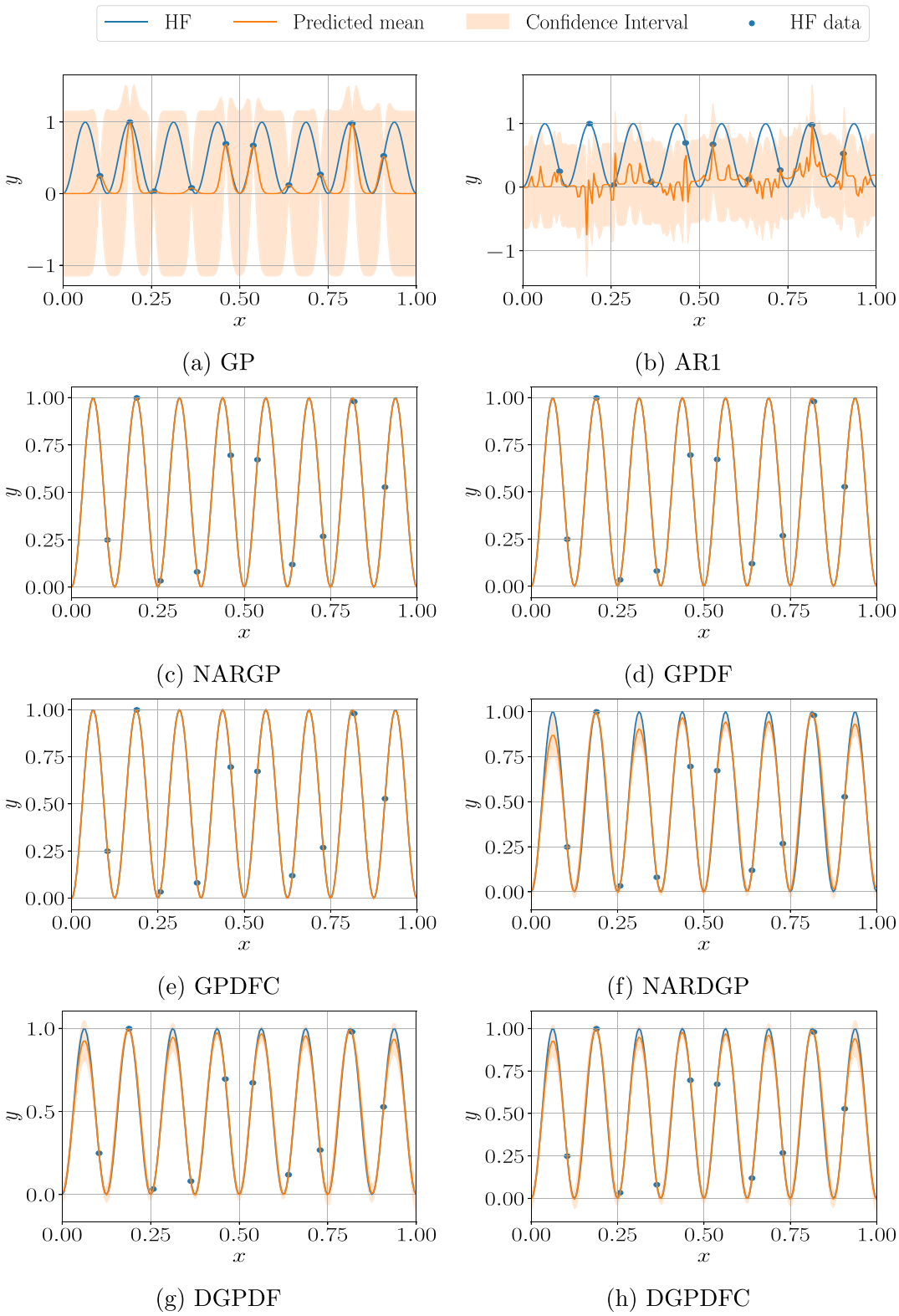


Figure A2. Comparison of different multi-fidelity Gaussian Process surrogate modeling methods on a non-linear transformation problem stated in table 1. The plots show the high-fidelity function in the blue curve, the high-fidelity training data in blue dots, the predicted mean in the orange curve, and 95% confidence interval using the orange-shaded region.

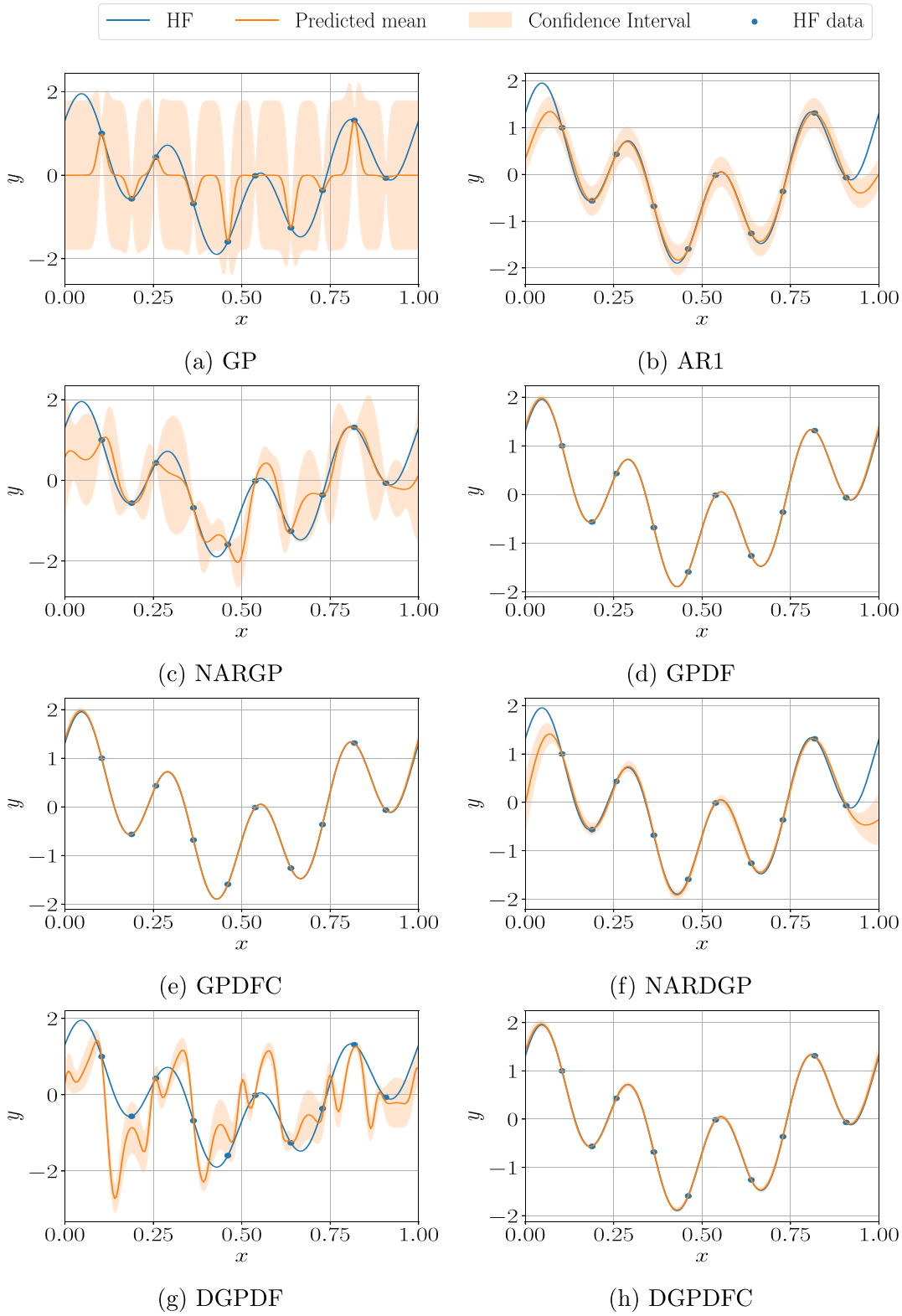


Figure A3. Comparison of different Multi-fidelity Gaussian Process surrogate modeling methods on a phase-shifted oscillation stated in table 1. The plots show the high-fidelity function in the blue curve, the high-fidelity training data in blue dots, the predicted mean in the orange curve, and 95% confidence interval using the orange-shaded region.

ORCID iDs

Kislaya Ravi https://orcid.org/0000-0003-1927-7651
Vladyslav Fediukov https://orcid.org/0009-0009-9257-3909
Felix Dietrich https://orcid.org/0000-0002-2906-1769
Tobias Neckel https://orcid.org/0000-0002-3442-7171
Fabian Buse https://orcid.org/0000-0002-2279-5735
Michael Bergmann https://orcid.org/0009-0005-6052-6420
Hans-Joachim Bungartz https://orcid.org/0000-0002-0171-0712

References

[1] Jiang P, Zhou Q, Shao X, Jiang P, Zhou Q and Shao X 2020 *Surrogate-Model-Based Design and Optimization* (Springer)
[2] Mack Y, Goel T, Shyy W and Haftka R 2007 Surrogate model-based optimization framework: a case study in aerospace design *Evolutionary Computation in Dynamic and Uncertain Environments* (Springer) pp 323–42
[3] Wang C, Qiang X, Xu M and Wu T 2022 Recent advances in surrogate modeling methods for uncertainty quantification and propagation *Symmetry* **14** 1219
[4] Farcaş I-G, Uekermann B, Neckel T and Bungartz H-J 2018 Nonintrusive uncertainty analysis of fluid-structure interaction with spatially adaptive sparse grids and polynomial chaos expansion *SIAM J. Sci. Comput.* **40** B457–82
[5] Yondo R, Andrés E and Valero E 2018 A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses *Prog. Aerosp. Sci.* **96** 23–61
[6] Balu A, Sarkar S, Ganapathysubramanian B and Krishnamurthy A 2022 Physics-aware machine learning surrogates for real-time manufacturing digital twin *Manuf. Lett.* **34** 71–74
[7] Peherstorfer B, Willcox K and Gunzburger M 2018 Survey of multifidelity methods in uncertainty propagation, inference and optimization *SIAM Rev.* **60** 550–91
[8] Giles M B 2015 Multilevel monte carlo methods *Acta Numer.* **24** 259–328
[9] Ng L W T and Eldred M 2012 Multifidelity uncertainty quantification using non-intrusive polynomial chaos and stochastic collocation *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conf.* p 1852
[10] Farcaş I-G, Peherstorfer B, Neckel T, Jenko F and Bungartz H-J 2023 Context-aware learning of hierarchies of low-fidelity models for multi-fidelity uncertainty quantification *Comput. Methods Appl. Mech. Eng.* **406** 115908
[11] Vinod V, Kleinekathöfer U and Zaspel P 2023 Optimized multifidelity machine learning for quantum chemistry *Mach. Learn.: Sci. Technol.* **5** 015054
[12] Ravi K, Neckel T and Bungartz H-J 2023 Multi-fidelity no-u-turn sampling *Monte Carlo and Quasi-Monte Carlo Methods. MCQMC 2022* (*Linz, Austria, 17–22 July 2022*) ed A Hinrichs, P Kritzer and F Pillichshammer pp 543–60
[13] Bue Lykkegaard M, Dodwell T J, Fox C, Mingas G and Scheichl R 2023 Multilevel delayed acceptance mcmc *SIAM/ASA J. Uncertain. Quantification* **11** 1–30
[14] Prescott T P and Baker R E 2020 Multifidelity approximate Bayesian computation *SIAM/ASA J. Uncertain. Quantification* **8** 114–38
[15] Agrawal A, Ravi K, Koutsourelakis P-S and Bungartz H-J 2023 Multi-fidelity constrained optimization for stochastic black box simulators (arXiv:2311.15137)
[16] Irshad F, Karsch S and Döpp A 2024 Leveraging trust for joint multi-objective and multi-fidelity optimization *Mach. Learn.: Sci. Technol.* **5** 015056
[17] Lazin M F, Shelton C R, Sandhofer S N and Wong B M 2023 High-dimensional multi-fidelity Bayesian optimization for quantum control *Mach. Learn.: Sci. Technol.* **4** 045014
[18] Görtz S *et al* (eds) 2020 Overview of collaborative multi-fidelity multidisciplinary design optimization activities in the dlr project victoria *AIAA Aviation 2020 Forum* (https://doi.org/10.2514/6.2020-3167)
[19] Abu-Zurayk M *et al* 2020 Sensitivity-based multifidelity multidisciplinary optimization of a powered aircraft subject to a comprehensive set of loads *AIAA Aviation 2020 Forum* (https://doi.org/10.2514/6.2020-3168)
[20] Sebastian Zakrzewski A, Lange F and Walter Hollmann R (eds) 2022 Multi-fidelity aerodynamic design process for moveables at dlr virtual product house *AIAA Aviation 2022 Forum* (American Institute of Aeronautics and Astronautics, Inc) Video Presentation: (https://doi.org/10.2514/6.2022-3938.vid)
[21] Williams C K I and Edward Rasmussen C 2006 *Gaussian Processes for Machine Learning* vol 2 (MIT Press)
[22] Kennedy M C and O'Hagan A 2000 Predicting the output from a complex computer code when fast approximations are available *Biometrika* **87** 1–13
[23] Le Gratiet L 2013 Multi-fidelity Gaussian process regression for computer experiments *PhD Thesis* Université Paris-Diderot-Paris VII
[24] Lee S, Dietrich F, Karniadakis G E and Kevrekidis I G 2019 Linking gaussian process regression with data-driven manifold embeddings for nonlinear data fusion *Interface Focus* **9** 20180083
[25] Perdikaris P, Raissi M, Damianou A, Lawrence N D and Em Karniadakis G 2017 Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling *Proc. R. Soc.* A **473** 20160751
[26] Cutajar K, Pullin M, Damianou A, Lawrence N and González J 2019 Deep gaussian processes for multi-fidelity modeling (arXiv:1903.07320)
[27] Krige D G 1951 A statistical approach to some basic mine valuation problems on the witwatersrand *J. South. Afr. Inst. Min. Metall.* **52** 119–39
[28] Owhadi H and Ryan Yoo G 2019 Kernel flows: from learning kernels from data into the abyss *J. Comput. Phys.* **389** 22–47
[29] David Hüwel J, Berns F and Beecks C 2021 Automated kernel search for gaussian processes on data streams *2021 IEEE Int. Conf. on Big Data (Big Data)* (IEEE) pp 3584–8
[30] Duvenaud D, Lloyd J, Grosse R, Tenenbaum J and Zoubin G 2013 Structure discovery in nonparametric regression through compositional kernel search *Int. Conf. on Machine Learning* (PMLR) pp 1166–74
[31] Horn D, Stork J, Schüßler N-J and Zaefferer M 2019 Surrogates for hierarchical search spaces: the wedge-kernel and an automated analysis *Proc. Genetic and Evolutionary Computation Conf.* pp 916–24
[32] Liu D C and Nocedal J 1989 On the limited memory bfgs method for large scale optimization *Math. Program.* **45** 503–28

[33] Capone A, Lederer A and Hirche S 2022 Gaussian process uniform error bounds with unknown hyperparameters for safety-critical applications *Int. Conf. on Machine Learning* (PMLR) pp 2609–24
[34] Capone A, Pleiss G and Hirche S 2023 Sharp calibrated gaussian processes (arXiv:2302.11961)
[35] Küppers F, Schneider J and Haselhoff A 2022 Parametric and multivariate uncertainty calibration for regression and object detection *European Conf. on Computer Vision* (Springer) pp 426–42
[36] Song H, Diethe T, Kull M and Flach P 2019 Distribution calibration for regression *Int. Conf. on Machine Learning* (PMLR) pp 5897–906
[37] Kuleshov V, Fenner N and Ermon S 2018 Accurate uncertainties for deep learning using calibrated regression *Int. Conf. on Machine Learning* (PMLR) pp 2796–804
[38] Levi D, Gispan L, Giladi N and Fetaya E 2022 Evaluating and calibrating uncertainty prediction in regression tasks *Sensors* **22** 5540
[39] Cranmer K, Brehmer J and Louppe G 2020 The frontier of simulation-based inference *Proc. Natl Acad. Sci.* **117** 30055–62
[40] Damianou A and Lawrence N D 2013 Deep gaussian processes *Artificial Intelligence and Statistics* (PMLR) pp 207–15
[41] Titsias M and Lawrence N D 2010 Bayesian gaussian process latent variable model *Proc. 13th Int. Conf. on Artificial Intelligence and Statistics* (*JMLR Workshop and Conference Proceedings*) pp 844–51
[42] Bach F R and Jordan M I 2005 Predictive low-rank decomposition for kernel methods *Proc. 22nd Int. Conf. on Machine Learning* pp 33–40
[43] Bauer M, van der Wilk M and Edward Rasmussen C 2016 Understanding probabilistic sparse gaussian process approximations *Advances in Neural Information Processing Systems* p 29
[44] Quinonero-Candela J and Edward Rasmussen C 2005 A unifying view of sparse approximate gaussian process regression *J. Mach. Learn. Res.* **6** 1939–59
[45] Burt D, Edward Rasmussen C and Van Der Wilk M 2019 Rates of convergence for sparse variational gaussian process regression *Int. Conf. on Machine Learning* (PMLR) pp 862–71
[46] Salimbeni H and Deisenroth M 2017 Doubly stochastic variational inference for deep gaussian processes *Advances in Neural Information Processing Systems* p 30
[47] Villemonteix J, Vazquez E and Walter E 2009 An informational approach to the global optimization of expensive-to-evaluate functions *J. Glob. Optim.* **44** 509–34
[48] Liu H, Ong Y-S and Cai J 2018 A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design *Struct. Multidiscip. Optim.* **57** 393–416
[49] Lin Y 2004 *An Efficient Robust Concept Exploration Method and Sequential Exploratory Experimental Design* (Georgia Institute of Technology)
[50] Sebastian Seung H, Opper M and Sompolinsky H 1992 Query by committee *Proc. 15th Annual Workshop on Computational Learning Theory* pp 287–94
[51] Freund Y, Sebastian Seung H, Shamir E and Tishby N 1992 Information, prediction and query by committee *Advances in Neural Information Processing Systems* p 5
[52] Rumpfkeil M, Yamazaki W and Dimitri M 2011 A dynamic sampling method for kriging and cokriging surrogate models *49th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition* p 883
[53] Gray J S, Hearn T A, Moore K T, Hwang J, Martins J R R A and Ning A 2014 Automatic evaluation of multidisciplinary derivatives using a graph-based problem formulation in openmdao *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conf.* p 2042
[54] Busby D 2009 Hierarchical adaptive experimental design for Gaussian process emulators *Reliab. Eng. Syst. Saf.* **94** 1183–93
[55] Xu Z and Liao Q 2020 Gaussian process based expected information gain computation for Bayesian optimal design *Entropy* **22** 258
[56] Matthews A G G, Van Der Wilk M Nickson T, Fujii K, Boukouvalas A, León-Villagrá P, Ghahramani Z and Hensman J 2017 GPflow: a Gaussian process library using TensorFlow *J. Mach. Learn. Res.* **18** 1–6
[57] Bliek L, Guijt A, Karlsson R, Verwer S and de Weerdt M 2021 Expobench: benchmarking surrogate-based optimisation algorithms on expensive black-box functions (arXiv:2106.04618)
[58] Hodgkin A L and Huxley A F 1952 A quantitative description of membrane current and its application to conduction and excitation in nerve *J. Physiol.* **117** 500
[59] Gregory Bekker M 1969 Introduction to terrain-vehicle systems. Part I: the terrain. Part II: the vehicle
[60] Buse F 2022 Development and validation of a deformable soft soil contact model for dynamic rover simulations *PhD Thesis* Tohoku University
[61] Fediukov V, Dietrich F and Buse F 2022 Multi-fidelity machine learning modeling for wheel locomotion *11th Asia-Pacific Regional Conf. Int. Society for Terrain-Vehicle Systems, ISTVS 2022. ISTVS* (https://doi.org/10.56884/WGPV6693)
[62] Barthelmes S 2018 Terra: terramechanics for real-time application *5th Joint Int. Conf. on Multibody System Dynamics*
[63] Buse F, Bellmann T, Lichtenheldt R and Krenn R 2018 The DLR terramechanics robotics locomotion lab *Int. Symp. on Artificial Intelligence, Robotics and Automation in Space*
[64] Buse F, Pignède A, Bertrand J, Goulet S'ebastien and Lagabarre S 2022 Mmx rover simulation-robotic simulations for phobos operations *2022 IEEE Aerospace Conf. (AERO)* (IEEE) pp 1–14
[65] Pignède A, Schindler W, Lichtenheldt R, Thiele B, Schütt M and Franke D 2022 Toolchain for a mobile robot applied on the DLR scout rover *2022 IEEE Aerospace Conf. (AERO)* (IEEE) pp 1–15
[66] Fasiolo M, Wood S N, Zaffran M, Nedellec R and Goude Y 2021 Fast calibrated additive quantile regression *J. Am. Stat. Assoc.* **116** 1402–12
[67] Yao J, Pan W, Ghosh S and Doshi-Velez F 2019 Quality of uncertainty quantification for bayesian neural network inference (arXiv:1906.09686)
[68] Huhne J 2023 Uncertainty quantification for gaussian processes *Bachelor thesis* Technical University of Munich
[69] Pereverzev G V and Yushmanov P N 2002 IPP 5/98 Garching: Max-Planck-Institut für Plasmaphysik
[70] Fable E *et al* 2013 Novel free-boundary equilibrium and transport solver with theory-based models and its validation against ASDEX upgrade current ramp scenarios *Plasma Phys. Control. Fusion* **55** 124028
[71] Hinton F L and Hazeltine R D 1976 Theory of plasma transport in toroidal confinement systems *Rev. Mod. Phys.* **48** 239
[72] Bourdelle C 2015 Turbulent transport in tokamak plasmas: bridging theory and experiment *PhD Thesis* Aix Marseille Université
[73] Ho A, Citrin J, Bourdelle C, Camenen Y, Casson F J, van de Plassche K L and Weisen H (JET Contributors) 2021 Neural network surrogate of qualikiz using jet experimental data to populate training space *Phys. Plasmas* **28** 032305
[74] van de Plassche K L, Citrin J, Bourdelle C, Camenen Y, Casson F J, Dagnelie V I, Felici F, Ho A and Van Mulders S (JET Contributors) 2020 Fast modeling of turbulent transport in fusion plasmas using neural networks *Phys. Plasmas* **27** 022310