Available online at www.sciencedirect.com
ScienceDirect

Comput. Methods Appl. Mech. Engrg. 387 (2021) 114128

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Latent map Gaussian processes for mixed variable metamodeling

Nicholas Oune, Ramin Bostanabad*

Mechanical and Aerospace Engineering, University of California, Irvine, Irvine, CA, USA

Received 7 February 2021; received in revised form 21 July 2021; accepted 15 August 2021
Available online xxxx

Abstract

Gaussian processes (GPs) are ubiquitously used in sciences and engineering as metamodels. Standard GPs, however, can only handle numerical or quantitative variables. In this paper, we introduce latent map Gaussian processes (LMGPs) that inherit the attractive properties of GPs and are also applicable to mixed data which have both quantitative and qualitative inputs. The core idea behind LMGPs is to learn a continuous, low-dimensional latent space or manifold which encodes all qualitative inputs. To learn this manifold, we first assign a unique prior vector representation to each combination of qualitative inputs. We then use a low-rank linear map to project these priors on a manifold that characterizes the posterior representations. As the posteriors are quantitative, they can be directly used in any standard correlation function such as the Gaussian or Matern. Hence, the optimal map and the corresponding manifold, along with other hyperparameters of the correlation function, can be systematically learned via maximum likelihood estimation. Through a wide range of analytic and real-world examples, we demonstrate the advantages of LMGPs over state-of-the-art methods in terms of accuracy and versatility. In particular, we show that LMGPs can handle variable-length inputs, have an explainable neural network interpretation, and provide insights into how qualitative inputs affect the response or interact with each other. We also employ LMGPs in Bayesian optimization and illustrate that they can discover optimal compound compositions more efficiently than conventional methods that convert compositions to qualitative variables via manual featurization.

Keywords: Gaussian processes; Emulation; Metamodeling; Mixed-variable optimization; Computer experiments; Manifold learning

1. Introduction

Metamodeling (a.k.a. emulation, surrogate modeling, or supervised learning) of physical experiments or expensive simulations is critical for the development of research in many fields of science and engineering. As an example, consider the design of the airfoil shape for an aircraft wing. Many possible airfoil designs exist and testing each design, physically or via finite element (FE) simulations, could take minutes to possibly days. In this scenario, metamodels accelerate the design process by mimicking the input–output behavior of the system in a computationally inexpensive manner. Many metamodels have been developed over the past few decades and some of the most popular ones are based on Gaussian processes (GPs, aka Kriging) [1–11], neural networks (NNs) [12–19], and trees [20–22]. In this paper, we focus on GPs which are easy to train, quantify prediction uncertainty, and perform extremely well with sparse datasets [2,3,23–29].

* Corresponding author.
E-mail address: raminb@uci.edu (R. Bostanabad).

https://doi.org/10.1016/j.cma.2021.114128

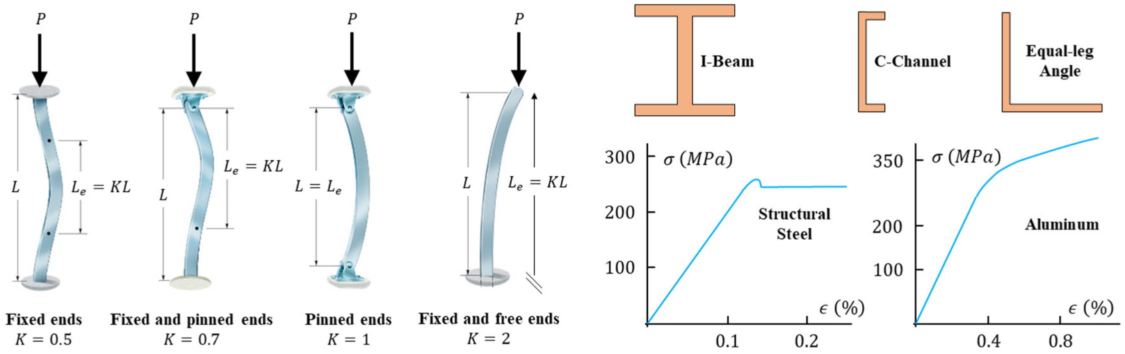
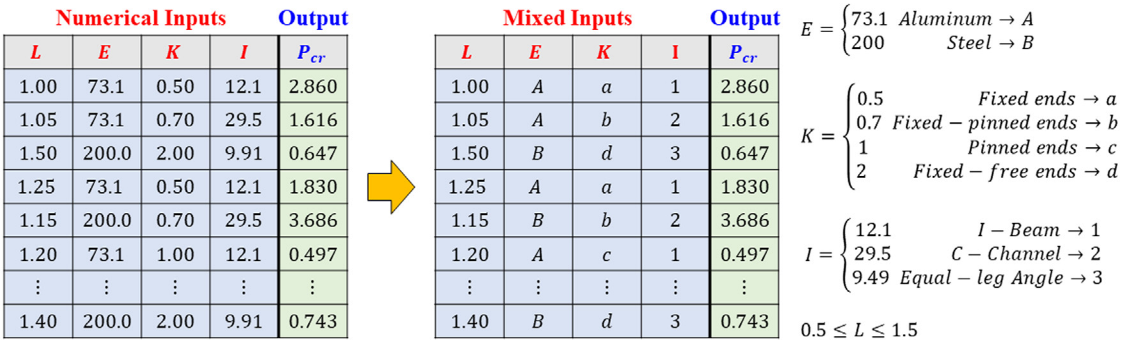


Fig. 1. Buckling of a column: For slender long members stability analysis based on buckling is necessary. The critical load, P_{cr}, that marks the onset of buckling depends on the length, material, cross-section, and end-support type of the column.

As detailed in Section 2, a GP metamodel relies on a covariance function that measures the weighted distance between any two input variables. In many real-world applications some inputs may be qualitative/categorical for which defining a distance measure is not straightforward. In such scenarios, traditional GPs break down as their covariance function cannot readily quantify the distances between qualitative inputs. To demonstrate such an application, consider the stability analysis of an ideal column whose critical load that marks the onset of buckling depends on four factors: material type, cross-section type, column length, and the end-support types, see Fig. 1 (we assume that the column buckles before it yields which is a valid assumption for long and slender members under compression). Three of these factors are qualitative (only length is quantitative) and hence a traditional GP cannot be used to link the critical load, P_{cr}, to all four factors simultaneously. As another example, consider the material design problem of identifying the optimal composition of the lacunar spinal family $XY^aY_3^bZ_8$ with trivalent main group X, transition metal Y and chalcogenide Z ions [30]. The lacunar spinel family contains properties desirable for microelectronics, and the goal is to find the composition that maximizes phase stability and band gap tunability. In this design example, the inputs are all categorical and include elements for each site, e.g., either of $\{Al, Ga, In\}$ for the X site. Since the differences such as $Al - Ga$ are not defined, GPs are also not directly applicable to this problem. Other engineering systems with qualitative factors include (1) fiber composites whose tensile strength depends on the fiber arrangement (unidirectional, bi-directional, random, woven, braided) [23,31], (3) cast metal alloys whose fracture toughness depends on the machine identification number and degassing status, (3) a stamping operation where system response (maximum strain over a stamped panel) depends on the lubricant type [32], and (4) thermal management of a data center where the thermal dynamics depends on diffuser location, power unit type, or rack heat load nonuniformity [33].

There are three broad approaches for handling qualitative inputs with a GP. In the first approach distinct GPs are trained for each combination of qualitative inputs. This approach is rarely adopted since it not only ignores possible correlations across qualitative variables, but also does not scale well to problems with even a moderate number of variables. In the second approach domain knowledge is used to manually convert qualitative inputs to quantitative ones which can subsequently be used in a traditional GP. This approach is ad-hoc and quite expensive but can prove useful in problems where either the training data is extremely scarce or strong prior information is available on the underlying numerical features that give rise to the nature of qualitative factors (see Section 5.4 for an example). The third approach requires re-structuring the covariance function. As detailed in Section 3, most of these methods accommodate qualitative inputs via covariance functions that resemble those of multi-response GPs [34] where each combination of the qualitative inputs corresponds to a single response.

Recently, Zhang et al. [32] developed a novel method that projects each qualitative input to a distinct continuous latent space which allows to directly use the corresponding latent variables in the GP's covariance function. This method builds latent variable Gaussian processes (LVGPs) and has been shown to consistently outperform prior methods. From the standpoint of converting qualitative inputs to quantitative ones, LVGPs are similar to the second approach discussed above. However, instead of relying on domain knowledge and manual conversion, LVGPs employ the training data and maximum likelihood estimation (MLE) and hence are much more versatile, efficient, and generally more accurate.



Numerical Inputs / Output

L	E	K	I	P_{cr}
1.00	73.1	0.50	12.1	2.860
1.05	73.1	0.70	29.5	1.616
1.50	200.0	2.00	9.91	0.647
1.25	73.1	0.50	12.1	1.830
1.15	200.0	0.70	29.5	3.686
1.20	73.1	1.00	12.1	0.497
⋮	⋮	⋮	⋮	⋮
1.40	200.0	2.00	9.91	0.743

Mixed Inputs / Output

L	E	K	I	P_{cr}
1.00	A	a	1	2.860
1.05	A	b	2	1.616
1.50	B	d	3	0.647
1.25	A	a	1	1.830
1.15	B	b	2	3.686
1.20	A	c	1	0.497
⋮	⋮	⋮	⋮	⋮
1.40	B	d	3	0.743

$$E = \begin{cases} 73.1 & Aluminum \rightarrow A \\ 200 & Steel \rightarrow B \end{cases}$$

$$K = \begin{cases} 0.5 & Fixed\ ends \rightarrow a \\ 0.7 & Fixed - pinned\ ends \rightarrow b \\ 1 & Pinned\ ends \rightarrow c \\ 2 & Fixed - free\ ends \rightarrow d \end{cases}$$

$$I = \begin{cases} 12.1 & I - Beam \rightarrow 1 \\ 29.5 & C - Channel \rightarrow 2 \\ 9.49 & Equal - leg\ Angle \rightarrow 3 \end{cases}$$

$0.5 \leq L \leq 1.5$

Fig. 2. Mixed Input data for the buckling example: The critical load, P_{cr}, is obtained via Euler's formula for any length and any combination of E, I, and K. To build a metamodel that predicts P_{cr} given the inputs, we train an LMGP using the dataset with the mixed variables where the underlying numerical values are masked with arbitrarily chosen labels. We show in Section 4.3 that the latent space of the trained LMGP clearly demonstrates that P_{cr} only depends on $\frac{EI}{K^2}$ rather than the individual labels (and hence the unseen individual numerical values) assigned to these inputs.

Our method for handling qualitative variables is in spirit similar to LVGP in that we also work with systematically learnt latent variables that encode each combination of qualitative inputs. However, as opposed to LVGP, our method relies on a parametric map that projects all combinations of qualitative inputs to a single latent space. Correspondingly, we call our method *latent map Gaussian processes* (LMGPs) and demonstrate in Sections 4 and 5 that they have some important advantages over LVGPs such as accommodating variable-length inputs or providing insights into the underlying physics of the problem. To articulate on the latter advantage, we return to the buckling example. We know that P_{cr} for an ideal column can be obtained by Euler's formula:

$$P_{cr} = \frac{\pi EI}{(KL)^2} = \frac{\pi EI}{(L_e)^2}, \tag{1}$$

where E is the Young's modulus, I is the (smallest) moment of inertial with respect to the neutral axes of the cross-section, K is the effective-length factor that depends on the end-support types, L is the length of the column, and L_e is the equivalent length. Suppose we do not know Eq. (1) and only have access to a *mixed dataset* where P_{cr} is recorded at various column lengths for different combinations of materials (aluminum or structural steel), beam cross-section (I-beam, C-channel, or equal-leg angle), and end-supports (fixed, pinned, pinned and fixed, or fixed and free), see Fig. 2. In Section 4.1 we argue and demonstrate that LMGPs are capable of discovering the single latent variable $\frac{EI}{K^2}$ which is neither directly observed nor a linear function of the qualitative inputs. An LVGP model, however, learns three distinct latent spaces in this example (one for each qualitative input) and hence fails to indicate that there is a single underlying latent variable that completely encodes the effect of three qualitative factors on P_{cr}. LVGPs are also incapable of handling variable length inputs.

The rest of the paper is organized as follows. Section 2 reviews standard methods for GP modeling. Section 3 summarizes existing techniques developed for handling qualitative inputs via GPs. Section 4 discusses our proposed strategy for training GPs on datasets that include qualitative inputs. Connections between the learnt latent space and the underlying physics, neural network (NN) interpretation of LMGPS, and potential modifications to our approach are also discussed in this section. Section 5 reports the results from evaluating our method against state-of-the-art on a set of analytic functions, real-world datasets, and a Bayesian optimization problem. Section 6 concludes the paper with some final remarks.

2. Gaussian process modeling

In this section, we review how to fit GP models to a purely numerical training dataset whose inputs and outputs are denoted by $\boldsymbol{x} = [x_1, x_2, \ldots, x_{d_x}]^T$ and y, respectively. Assume the training data come from a realization of a Gaussian random process, $\eta(\boldsymbol{x})$, is defined as the following:

$$\eta(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})\boldsymbol{\beta} + \xi(\boldsymbol{x}),$$

where $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), \ldots, f_h(\boldsymbol{x})]$ are a set of pre-determined parametric basis functions (e.g., $x_1^2 x_2$, $x_2^2 sin(x_1)$, $log(x_1 x_2), \ldots$), $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_h]^T$ are the unknown coefficients of the basis functions, and $\xi(\boldsymbol{x})$ is a zero-mean GP. Since $\xi(\boldsymbol{x})$ is zero-mean, it is completely characterized by its parameterized covariance function:

$$cov\left(\xi(\boldsymbol{x}), \xi\left(\boldsymbol{x}'\right)\right) = c\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \sigma^2 r\left(\boldsymbol{x}, \boldsymbol{x}'\right), \tag{2}$$

where σ^2 is the process variance and $r(\cdot, \cdot)$ is a user-defined parametric correlation function. There are many types of correlation functions [1,35–37], but the most common one is the Gaussian correlation function defined as:

$$r\left(\boldsymbol{x}, \boldsymbol{x}'\right) = exp\left\{-\sum_{i=1}^{d_x} 10^{\omega_i}\left(x_i - x_i'\right)^2\right\} = \exp\left(\left(\boldsymbol{x} - \boldsymbol{x}'\right)^T 10^{\boldsymbol{\Omega}}\left(\boldsymbol{x} - \boldsymbol{x}'\right)\right), \tag{3}$$

where $\boldsymbol{\omega} = \left[\omega_1, \ldots, \omega_{d_x}\right]^T$, $-\infty < \omega_i < \infty$ are the roughness or scale parameters (in practice the ranges are limited to $-10 < \omega_i < 6$ to ensure numerical stability) and $\boldsymbol{\Omega} = diag(\boldsymbol{\omega})$. σ^2 and $\boldsymbol{\omega}$ are collectively referred to as the hyperparameters of the covariance function.

For GP emulation, point estimates of $\boldsymbol{\beta}, \boldsymbol{\omega}$, and σ^2 must be determined based on the data. These estimates can be found via either cross-validation (CV) or MLE. Alternatively, Baye's rule can be applied to find posterior distributions of the hyperparameters if prior knowledge is available. In this paper, MLE is employed because it provides a high generalization power while minimizing the computational costs [1,38]. MLE works by estimating $\boldsymbol{\beta}$, $\boldsymbol{\omega}$, and σ^2 such that they maximize the likelihood of the n training data being generated by $\eta(\boldsymbol{x})$, that is:

$$\left[\hat{\boldsymbol{\beta}}, \hat{\sigma}, \hat{\boldsymbol{\omega}}\right] = \underset{\boldsymbol{\beta}, \sigma^2, \boldsymbol{\omega}}{\operatorname{argmax}} \left|2\pi\sigma^2 \boldsymbol{R}\right|^{-\frac{1}{2}} \times exp\left\{\frac{-1}{2}(\boldsymbol{y} - \boldsymbol{F\beta})^T \left(\sigma^2 \boldsymbol{R}\right)^{-1}(\boldsymbol{y} - \boldsymbol{F\beta})\right\},$$

Or equivalently,

$$\left[\hat{\boldsymbol{\beta}}, \hat{\sigma}, \hat{\boldsymbol{\omega}}\right] = \underset{\boldsymbol{\beta}, \sigma^2, \boldsymbol{\omega}}{\operatorname{argmin}} \frac{n}{2} log\left(\sigma^2\right) + \frac{1}{2} log(|\boldsymbol{R}|) + \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{F\beta})^T \boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F\beta}), \tag{4}$$

where $log(\cdot)$ is the natural logarithm, $|\cdot|$ denotes the determinant operator, $\boldsymbol{y} = \left[y_{(1)}, \ldots, y_{(n)}\right]^T$ is an $n \times 1$ vector of outputs in the training data, $\boldsymbol{R}$ is the $n \times n$ correlation matrix with the (i, j)th element $\boldsymbol{R}_{ij} = r\left(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)}\right)$ for $i, j = 1, \ldots, n$, and $\boldsymbol{F}$ is an $n \times h$ matrix with the (k, l)th element $\boldsymbol{F}_{kl} = f_l\left(\boldsymbol{x}_{(k)}\right)$ for $k = 1, \ldots, n$ and $l = 1, \ldots, h$. By setting the partial derivatives with respect to $\boldsymbol{\beta}$ and σ^2 to zero, their estimates can be solved in terms of $\boldsymbol{\omega}$ as follows:

$$\hat{\boldsymbol{\beta}} = \left[\boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{F}\right]^{-1}\left[\boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{y}\right], \tag{5}$$

$$\hat{\sigma}^2 = \frac{1}{n}\left(\boldsymbol{y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}\right)^T \boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}\right), \tag{6}$$

Plugging these estimates into Eq. (4) and removing the constants yields:

$$\hat{\boldsymbol{\omega}} = \underset{\boldsymbol{\omega}}{\operatorname{argmin}}\, n log\left(\hat{\sigma}^2\right) + log(|\boldsymbol{R}|) = \underset{\boldsymbol{\omega}}{\operatorname{argmin}} L. \tag{7}$$

By minimizing L (i.e., solving Eq. (7)), one can solve for $\hat{\boldsymbol{\omega}}$ and subsequently, obtain $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}^2$ using Eq. (5) and (6). While many heuristic global optimization methods exist such as genetic algorithms [39] and particle swarm optimization [40], gradient-based optimization techniques based on, e.g., the L-BFGS algorithm [41], are generally preferred due to their ease of implementation and superior computational efficiency [3,35]. With gradient-based approaches, it is essential to start the optimization via numerous initial guesses to improve the chances of achieving global optimality.

After obtaining the hyperparameters via MLE, the following closed-form formula is used to predict the response at any $\boldsymbol{x}^*$:

$$\mathbb{E}[y^*] = \boldsymbol{f}(\boldsymbol{x}^*)\hat{\boldsymbol{\beta}} + \boldsymbol{g}^T(\boldsymbol{x}^*)\boldsymbol{V}^{-1}\left(\boldsymbol{y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}\right), \tag{8}$$

where $\mathbb{E}$ denotes expectation, $\boldsymbol{f}(\boldsymbol{x}^*) = [f_1(\boldsymbol{x}^*), \ldots, f_h(\boldsymbol{x}^*)]$, $\boldsymbol{g}(\boldsymbol{x}^*)$ is an $n \times 1$ vector with the ith element $c\left(\boldsymbol{x}_{(i)}, \boldsymbol{x}^*\right) = \hat{\sigma}^2 r\left(\boldsymbol{x}_{(i)}, \boldsymbol{x}^*\right)$, and $\boldsymbol{V}$ is the covariance matrix with the (i, j)th element $\hat{\sigma}^2 r\left(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)}\right)$. The posterior covariance between the responses at the two inputs $\boldsymbol{x}^*$ and $\boldsymbol{x}'$ is:

$$cov\left(y^*, y'\right) = c\left(\boldsymbol{x}^*, \boldsymbol{x}'\right) - \boldsymbol{g}^T(\boldsymbol{x}^*)\boldsymbol{V}^{-1}\boldsymbol{g}\left(\boldsymbol{x}'\right) + \boldsymbol{h}(\boldsymbol{x}^*)^T\left(\boldsymbol{F}^T\boldsymbol{V}^{-1}\boldsymbol{F}\right)^{-1}\boldsymbol{h}\left(\boldsymbol{x}'\right),$$

where $\boldsymbol{h}(\boldsymbol{x}^*) = \left(\boldsymbol{f}(\boldsymbol{x}^*) - \boldsymbol{F}^T\boldsymbol{V}^{-1}\boldsymbol{g}(\boldsymbol{x}^*)\right)$.

The above formulations can be easily extended to cases where the dataset is noisy. GPs can address noise and smooth data by using a nugget or jitter parameter, δ [42]. As a result, $\boldsymbol{R}$ becomes $\boldsymbol{R}_\delta = \boldsymbol{R} + \delta \boldsymbol{I}_{n\times n}$ where $\boldsymbol{I}_{n\times n}$ is the identity matrix of size $n \times n$. If the nugget parameter is used, the estimated (stationary) noise variance in the data is $\delta\hat{\sigma}^2$. GPs are also applicable to multi-response datasets by using, e.g., a separable covariance function [34,43,44] which replaces σ^2 with the matrix $\boldsymbol{\Sigma}$ whose off-diagonal elements represent the covariance between the corresponding responses at any fixed $\boldsymbol{x}$.

As the above formulations indicate, GP modeling relies on the correlation function, $r(\cdot,\cdot)$ in Eq. (3). $r(\cdot,\cdot)$ measures the correlation between the outputs at any two input locations as a function of the relative distance between those two inputs. Since the distance between categorical variables (such as gender, zip code, country, material coating type, etc.) cannot be directly defined, standard GP modeling techniques are not applicable to datasets that contain categorical variables. This issue is well established in the literature [45] and in the next section, we review the most common existing strategies that address it by reformulating the covariance function such that it can handle categorical variables.

3. Existing approaches for handling categorical variables

Let us denote the categorical inputs by $\boldsymbol{t} = \left[t_1, \ldots, t_{d_t}\right]^T$ where the total number of distinct levels for qualitative variable t_i is m_i. For instance, $t_1 = \{92697, 92093\}$ and $t_2 = \{math, physics, chemistry\}$ are two categorical inputs that encode zip code ($m_1 = 2$ levels) and course subject ($m_2 = 3$ levels), respectively. Inputs for mixed (numerical and categorical) training data are collectively denoted by $\boldsymbol{w} = [\boldsymbol{x}; \boldsymbol{t}]$ which is a column vector of size $(d_x + d_t) \times 1$.

3.1. Unrestrictive covariance (UC)

One popular strategy for GP modeling with categorical variables, introduced by Qian et al. [46], assumes a correlation function with the following form:

$$r\left(\boldsymbol{w}, \boldsymbol{w}'\right) = \prod_{i=1}^{d_t} \tau^i_{l,l'} \times exp\left\{-\left(\boldsymbol{x} - \boldsymbol{x}'\right)^T 10^{\boldsymbol{\Omega}} \left(\boldsymbol{x} - \boldsymbol{x}'\right)\right\}, \tag{9}$$

where $\tau^i_{l,l'}$ is a parameter that correlates levels l and l' of the variable t_i. That is, Eq. (9) assigns the correlation matrix $\boldsymbol{\tau}^i$ to the categorical variable t_i where $\tau^i_{l,l'}$ serves as a distance metric between levels l and l' of t_i. Since $\boldsymbol{\tau}^i$ is a correlation matrix, it must be symmetric positive definite with unit diagonal elements. Hence, there are a total of $\sum_{i=1}^{d_t} m_i \left(m_i - 1\right)/2$ parameters that need to be estimated (in addition to $\boldsymbol{\Omega}$) in Eq. (9). Since the number of hyperparameters in the UC function increases quadratically, it is not applicable to problems where there are many levels or categorical variables. Even in simple problems, the constraints on $\boldsymbol{\tau}^i$ render the optimization of the log-likelihood function quite difficult. Additionally, Eq. (9) has limited generalization power. For example, as Deng et al. [47] point out, if $\tau^i_{l,l'}$ is estimated as 0 for any categorical variable, the entire correlation between two sample points, $r\left(\boldsymbol{w}, \boldsymbol{w}'\right)$, reduces to 0.

3.2. Multiplicative covariance (MC)

Multiplicative covariance function is a simplified version of the UC function [46] which assumes that for all $\boldsymbol{t} \neq \boldsymbol{t}'$:

$$\tau^i_{l,l'} = exp\left\{-\left(\theta^i_l + \theta^i_{l'}\right)\right\}, \tag{10}$$

where $\theta^i_l > 0$ is a parameter associated with the lth level of categorical variable t_i. That is, Eq. (10) assigns a number to each level of each categorical variable and hence requires estimating a total of $\sum_{i=1}^{d_t} m_i$ parameters (in addition to $\boldsymbol{\Omega}$) during the MLE process. While the MC function is simpler to optimize than the UC function, it is quite inflexible [48]. To demonstrate this, consider a scenario where there is one categorical variable with four levels. Also, suppose that the response surfaces corresponding to (i) levels 1 and 2 are highly correlated, (ii) levels 3 and 4 are highly correlated, and (iii) levels 1 and 2 are uncorrelated with those of levels 3 and 4. According to Eq. (10), we need to have $\theta_1 \approx \theta_2 \approx 0$ for the response surfaces for levels 1 and 2 to be highly correlated. With a similar reasoning, we have $\theta_3 \approx \theta_4 \approx 0$. However, for the response surfaces of levels 2 and 3 to be uncorrelated, $\theta_2 + \theta_3$ must be large, which cannot be true if both are close to zero.

3.3. Additive Gaussian process (AGP)

The UC and MC strategies both assume a multiplicative covariance structure. Deng et al. [47] proposed a new additive covariance structure as follows:

$$c\left(\boldsymbol{w}\left(\boldsymbol{x},\boldsymbol{t}\right),\boldsymbol{w}'\left(\boldsymbol{x}',\boldsymbol{t}'\right)\right)=\sum_{i=1}^{d_t}\sigma_i^2\tau_{l,l'}^i r\left(\boldsymbol{x},\boldsymbol{x}'|\boldsymbol{\omega}_i\right)$$

where $r\left(\boldsymbol{x},\boldsymbol{x}'|\boldsymbol{\omega}_i\right)$ is the Gaussian correlation function as defined in Eq. (3) with correlation parameter vector $\boldsymbol{\omega}_i$ associated with qualitative factor t_i, σ_i^2 is the prior variance term for categorical variable t_i, and $\tau_{l,l'}^i$ has the same definition as in Eq. (10). According to Deng et al. [47], the AGP is more flexible than the UC function when there are multiple categorical variables. This is because the UC model assumes a fixed covariance structure over the numerical features, $\boldsymbol{x}$, for all categorical variables while the additive structure does not. However, the AGP also has a few major limitations. For example, the optimization based on MLE involves estimating a total number of $(1+d_x)\times d_t+\sum_{i=1}^{d_t}m_i\left(m_i-1\right)/2$ parameters which rapidly increases as the dimensionality of the problem grows. Visualization of how the underlying response surface changes within and across the categorical variables is also not straightforward with AGP.

3.4. Latent Variable Gaussian Process (LVGP)

Latent Variable Gaussian Process [32,49] is a recent work that handles categorical variables by learning a latent space of dimensionality d_z for each categorical variable t_i for $i=1,\ldots,d_t$. In other words, the m_i levels of t_i are represented as m_i points in the ith latent space. With this latent representation, the distance between any two points is defined. Hence, the latent points can be directly used in any standard correlation function such as the Gaussian:

$$r\left(\boldsymbol{w},\boldsymbol{w}'\right)=exp\left\{-\sum_{i=1}^{d_t}\left\|\boldsymbol{z}^i\left(t_i\right)-\boldsymbol{z}^i\left(t_i'\right)\right\|_2^2-\left(\boldsymbol{x}-\boldsymbol{x}'\right)^T10^{\boldsymbol{\Omega}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)\right\}, \tag{11}$$

where $\boldsymbol{z}^i\left(l\right)=\left[z_1^i\left(l\right),\ldots,z_{d_z}^i\left(l\right)\right]^T$ is the latent space point corresponding to level l (for $l=1,\ldots,m_i$) of the qualitative factor t_i and $\|\cdot\|_2$ denotes the Euclidean 2-norm. With this formulation, all the latent points (along with $\boldsymbol{\omega}$) can be found via MLE as described in Section 2 where Eq. (3) must be replaced via Eq. (11). Zhang et al. [32] recommend using a $2D$ latent space for each categorical variable where three constraints are imposed to ensure translation and rotation invariances in each $2D$ space. Thus, fitting an LVGP model involves estimating $d_x+\sum_{i=1}^{d_t}\left(2m_i-3\right)$ parameters.

Zhang et al. [32] show that LVGP consistently outperforms previously mentioned strategies in a wide range of problems. This superior performance is primarily because (i) in many real-world scenarios, categorical variables represent underlying numerical features whose collective effects can be captured in the learned $2D$ latent space, and (ii) the correlation function in Eq. (11) provides a much more versatile reformulation and does not impose any a priori relation between the categorical variables.

4. Latent map Gaussian process (LMGP)

Our proposed approach, similar to LVGP, involves mapping categorical variables to some points in a latent space. However, there are two key differences between LMGP and LVGP. Firstly, instead of directly estimating the latent positions, LMGP learns a linear transformation that maps a prior representation of the categorical variables to the latent space. Secondly, LMGP uses a single latent space while LVGP uses a unique latent space for each categorical variable. As argued below and shown in Section 5, these differences make LMGP a more versatile and scalable metamodel than LVGP.

In Sections 4.1–4.3 we provide the motivation for LMGP and the technical details, respectively. In Section 4.4, we draw some connections between LMGP and some other concepts (NNs, sufficient dimension reduction, and active subspaces) and also introduce an extension that enables LMGP to handle variable-length (or conditional) categorical inputs.

4.1. Motivations for latent space representation

Mapping categorical variables into a latent space has a strong justification because in all physical systems with such inputs, there exist some underlying numerical features that characterize the levels of each categorical variable. This is clearly the case for the buckling example introduced in Section 1 where the critical load depends on the column's material type, cross-section type, and end-support type which are *sufficiently* quantified by Young's modulus (E), moment of inertia of the cross-section (I), and effective-length factor (K), respectively. In other words, E, I, and K are the latent variables that encode the effects of the corresponding qualitative factors on P_{cr} for an ideal column. For instance, E encapsulates the effective strength of a large number of material bonds at different length-scales which resist lateral deflections of the column by absorbing energy.

In the simple buckling example, we *a priori* knew the mapping between the qualitative factors and the corresponding sufficient latent variables via the Euler equation. In more complex problems, these relations are generally either unknown or too high-dimensional to directly encode. To demonstrate such a complex case, recall the design problem regarding the lacunar spinal family discussed in the introduction. The differences between the qualitative factors (i.e., the elements in the periodic table) can be captured through some numerical features such as atomic number, atomic mass, or number of valence electrons. It is evident that there are many underlying numerical features that characterize the level-wise differences in a qualitative factor (e.g., l_1 vs l_2 of t_1) or across different qualitative factors (e.g., l_1 of $t_1$vs l_1 of t_2). This intrinsic high-dimensionality and the fact that we do not know (or cannot observe/measure) all the underlying numerical features challenge the identification of the few latent variables that sufficiently encode the qualitative factors.

To address the above challenges we note that in most physical problems the underlying numerical features are generally highly correlated and some of them have little effect on the response of interest [12,50]. Hence, with an appropriate learning algorithm and a representative training dataset, a low dimensional latent space can be learned that sufficiently quantifies the underlying numerical features. Returning to the buckling example, we can show via either LMGP or Euler's formula in Eq. (1) that instead of assigning a latent variable to each qualitative factor (as LVGP does), a single latent variable can sufficiently characterize the effect of all qualitative factors on P_{cr}. These and similar arguments underlie the widespread use of latent variables in deep NNs which, unlike LMGP, have a large number of parameters and are generally constructed using big data.

4.2. Emulation via LMGP

LMGP begins with an initial latent space representation of the categorical inputs. This prior representation is then projected to a lower dimensional space via a linear map which is learned via MLE, see Fig. 3. In particular, we first assign a unique vector (i.e., a prior representation) to each combination of the categorical variables. Then, we use matrix multiplication to map each of these unique vectors to a point in a latent space of dimensionality d_z:

$$\boldsymbol{z}\left(\boldsymbol{t}\right)=\boldsymbol{\zeta}\left(\boldsymbol{t}\right)\boldsymbol{A},$$

where $\boldsymbol{t}$ is a particular combination of the categorical variables, $\boldsymbol{z}\left(\boldsymbol{t}\right)$ is the $1\times d_z$ posterior latent representation of $\boldsymbol{t}$, $\boldsymbol{\zeta}\left(\boldsymbol{t}\right)$ is the $1\times\sum_{i=1}^{d_t}m_i$ unique prior vector representation of $\boldsymbol{t}$, and $\boldsymbol{A}$ is a $\sum_{i=1}^{d_t}m_i\times d_z$ mapping matrix that maps $\boldsymbol{\zeta}\left(\boldsymbol{t}\right)$ to $\boldsymbol{z}\left(\boldsymbol{t}\right)$. These points can now be directly inserted into any standard correlation function such as the Gaussian:

$$r\left(\boldsymbol{w},\boldsymbol{w}'\right)=exp\left\{-\left\|\boldsymbol{z}\left(\boldsymbol{t}\right)-\boldsymbol{z}\left(\boldsymbol{t}'\right)\right\|_2^2-\left(\boldsymbol{x}-\boldsymbol{x}'\right)^T10^{\boldsymbol{\Omega}}\left(\boldsymbol{x}-\boldsymbol{x}'\right)\right\}. \tag{12}$$

Finally, we optimize $\boldsymbol{A}$ simultaneously with $\boldsymbol{\Omega}=diag\left(\boldsymbol{\omega}\right)$ via MLE:

$$\left[\hat{\boldsymbol{\omega}},\hat{\boldsymbol{A}}\right]=\underset{\boldsymbol{\omega},\boldsymbol{A}}{\operatorname{argmin}}\ nlog\left(\hat{\sigma}^2\right)+log\left(|\boldsymbol{R}|\right),$$

where $\boldsymbol{R}$ and $\hat{\sigma}^2$ are now functions of both $\boldsymbol{\omega}$ and $\boldsymbol{A}$. When a $2D$ latent space is used ($d_z=2$), which we do in this paper, three constraints can be applied to the posterior latent positions to ensure rotation and translation invariance of the learned representation. Denoting the horizontal and vertical axes of this posterior space by z_1 and z_2, respectively, these constraints are: (i) The first latent position is located at the origin ($z_1=z_2=0$), (ii) the second latent position has $z_1\geq 0$ and $z_2=0$, and (iii) the third latent position has $z_2\geq 0$.

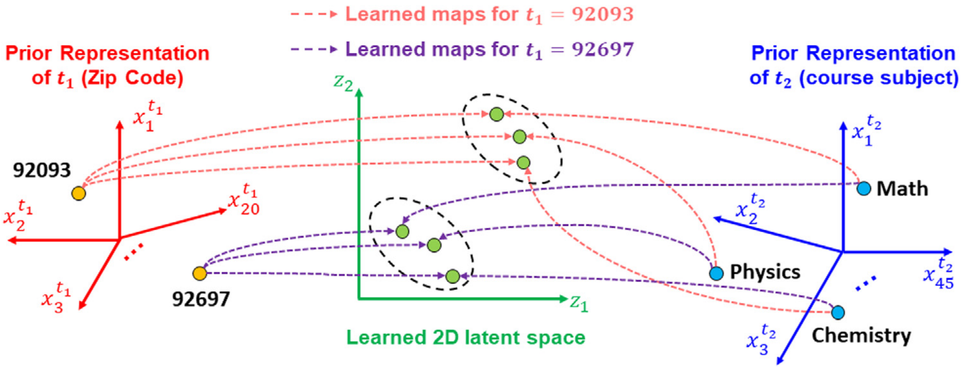


Fig. 3. Learning latent space via LMGP: The high-dimensional prior representations of categorical variables are mapped into a $2D$ latent space where the mapping is learnt via MLE. The mapped are colored based on the levels of t_1. In this illustrative example, changing the level of t_1 affects the latent positions more and hence the response is more sensitive to t_1 than t_2.

While $\boldsymbol{A}$ is learned via MLE based on some training data, the prior representations, $\boldsymbol{\zeta}(\boldsymbol{t})$, are user defined and can affect the performance of LMGP. We propose two strategies for defining $\boldsymbol{\zeta}(\boldsymbol{t})$. One method, which we call the random initialization, is to define $\boldsymbol{\zeta}(\boldsymbol{t})$ as a $1 \times \sum_{i=1}^{dt} m_i$ vector of random values ranging from, e.g., 0 to 1 (other ranges can be used which will result in larger/smaller estimates for the elements of $\boldsymbol{A}$). For instance, consider the example in Section 3 where $t_1 = \{92697, 92093\}$ and $t_2 = \{math, physics, chemistry\}$. With random initialization, $\boldsymbol{\zeta}(\boldsymbol{t})$ for each combination of levels of the categorical variables is defined as follows:

$$\begin{bmatrix} \{92697, math\} \\ \{92697, physics\} \\ \{92697, chemistry\} \\ \{92093, math\} \\ \{92093, physics\} \\ \{92093, chemistry\} \end{bmatrix} \rightarrow \begin{bmatrix} \boldsymbol{\zeta}(1,1) \\ \boldsymbol{\zeta}(1,2) \\ \boldsymbol{\zeta}(1,3) \\ \boldsymbol{\zeta}(2,1) \\ \boldsymbol{\zeta}(2,2) \\ \boldsymbol{\zeta}(2,3) \end{bmatrix} = \boldsymbol{\chi}_{6\times 5}, \qquad \chi_{ij} \sim Uni(0,1)$$

where $\boldsymbol{\zeta}(a, b)$ is the unique vector representation when the first and second categorical variables are at levels a and b, respectively, and $\boldsymbol{\chi}$ is a matrix whose elements are independent and identically distributed (IID) random numbers that follow a standard uniform distribution. While $\boldsymbol{\chi}$ is random and completely changes[1] each time we fit an LMGP to a particular dataset, our studies indicate that the resulting latent positions (and hence the accuracy of LMGP) are not affected. This consistency in posterior representation is provided by $\boldsymbol{A}$ whose elements are estimated via MLE. Furthermore, to reduce computational costs (esp. in very high dimensional problems), one can reduce the number of columns of $\boldsymbol{\chi}$ which will reduce the number of rows of $\boldsymbol{A}$. This strategy is very appealing when the number of categorical variables and/or their levels are high. However, we found through testing that this would be at the expense of potential reduction in prediction performance.

The second initialization strategy is to use a grouped one-hot encoded vector for $\boldsymbol{\zeta}(\boldsymbol{t})$ that consists of 1s and 0s. In a $1-0$ vector representation, the 1s correspond to the levels used for each categorical variable while the 0s correspond to the rest of the levels. By applying this approach to the zip code-course subject example, $\boldsymbol{\zeta}(\boldsymbol{t})$ is obtained as follows:

$$\begin{bmatrix} \{92697, math\} \\ \{92697, physics\} \\ \{92697, chemistry\} \\ \{92093, math\} \\ \{92093, physics\} \\ \{92093, chemistry\} \end{bmatrix} \rightarrow \begin{bmatrix} \boldsymbol{\zeta}(1,1) \\ \boldsymbol{\zeta}(1,2) \\ \boldsymbol{\zeta}(1,3) \\ \boldsymbol{\zeta}(2,1) \\ \boldsymbol{\zeta}(2,2) \\ \boldsymbol{\zeta}(2,3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}_{6\times 5}$$

In our studies, the $1-0$ representation consistently outperformed the random representation based on the performance of LMGP on test data. It also resulted in better structured latent positions that more clearly demonstrate

[1] Assuming the random number generator seed is not fixed.

the relations between the categorical variables and their relative effect on the response. These favorable properties are because the $1-0$ representation acts as an informative prior that helps LMGP in distinguishing the interactions between categorical variables and their levels. However, the random vector representation provides an uninformative prior where $\boldsymbol{\zeta}(\boldsymbol{t})$ is not generated based on any knowledge of the categorical variable levels.

To verify that LMGP is actually utilizing knowledge of the levels used for each categorical variable, we compared its prediction performance in two scenarios: standard LMGP (as described above with $1-0$ representation) and LMGP that lumps all categorical variables into a single new one where each level corresponds to a set of levels for the original categorical variable. Consider the zip code-course subject example again. By combining the two categorical variables into a single new categorical variable, $\boldsymbol{\zeta}(\boldsymbol{t})$ becomes a diagonal matrix as shown below:

$$\begin{bmatrix} \{92697, math\} \\ \{92697, physics\} \\ \{92697, chemistry\} \\ \{92093, math\} \\ \{92093, physics\} \\ \{92093, chemistry\} \end{bmatrix} \rightarrow \begin{bmatrix} \boldsymbol{\zeta}(1) \\ \boldsymbol{\zeta}(2) \\ \boldsymbol{\zeta}(3) \\ \boldsymbol{\zeta}(4) \\ \boldsymbol{\zeta}(5) \\ \boldsymbol{\zeta}(6) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{6\times 6}$$

Because the categorical variables are combined into a single new categorical variable, the levels used for each categorical variable is unknown to LMGP. Thus, poorer prediction performance is expected. Through testing, we found that standard LMGP consistently outperformed LMGP with the categorical variables combined. This implies that the prior knowledge of the levels for each categorical variable is assisting LMGP with discovering a more representative latent position structure.

We argue that LMGP is a more suitable approach to latent space learning than LVGP because of the following four key reasons. Firstly, LMGP provides a systematic mechanism to embed prior knowledge (the $1-0$ vector representation in our case) into the training process while LVGP directly estimates the latent positions. Our mechanism greatly improves the optimization process which, in turn, results in models with higher predictive power. Secondly, mapping all possible $\boldsymbol{t}$ vectors to a single latent space (as opposed to having a latent space for each categorical variable) allows the user to analyze and visualize the interactions across the categorical variables. Thirdly, while LMGP requires estimating more hyperparameters than LVGP, it achieves a more aggressive dimensionality reduction (the total number of hyperparameters in LMGP and LVGP are $d_x + 2\times\sum_{i=1}^{d_t} m_i$ and $d_x + \sum_{i=1}^{d_t}(2m_i - 3)$, respectively, for $d_z = 2$). This is because all the latent positions in LMGP are enforced to lie on a single latent space (aka manifold [51]) while LVGP uses a manifold for each categorical variable. Lastly, LMGP avoids non-identifiability issues that LVGP encounters: As we show in Section 5.1, when LVGP is trained on noisy data where one or more of the categorical variables have negligible effect on the response, the latent positions cannot be robustly estimated because their effect on the correlation function is akin to that of the nugget parameter. We demonstrate some of these remarks below and the rest in Section 5.

4.3. Interpreting the latent space

We introduced the buckling example in the introduction and elaborated on its latent space representation in Section 4.1. In this section we demonstrate that an LMGP can discover the single sufficient latent variable $\frac{EI}{K^2}$ that encodes the effect of the qualitative factors (material type, cross-section type, and end-support type) on P_{cr}.

Following the procedures outlined in Section 4.2, we fit an LMGP with $d_z = 2$ to a mixed data of size $n = 100$ where the numerical values are masked with randomly chosen labels, see Fig. 2 for the relations between labels and the corresponding numerical values (these relations are not used when training the LMGP). The latent positions learned via LMGP are illustrated in Fig. 4 where the range of z_2 is much smaller than that of z_1. This observation indicates that a single latent variable is sufficient to encode the effect of the three qualitative inputs on P_{cr}.

To demonstrate that the horizontal axis indeed encodes $\frac{EI}{K^2}$ we include the numerical value of $\frac{EI}{K^2}$ that corresponds to each latent position in Fig. 4. As it can be observed, these values monotonically increase from left to right (this direction is a result of forcing $Aa1$ and $Ba1$ to be at, respectively, the origin and the positive side of z_1). Note that the relation between $\frac{EI}{K^2}$ and z_1 is nonlinear since, while P_{cr} linearly depends on $\frac{EI}{K^2}$, $\mathbb{E}[y^*]$ in Eq. (8) nonlinearly depends on z_1 through the correlation function in Eq. (12).

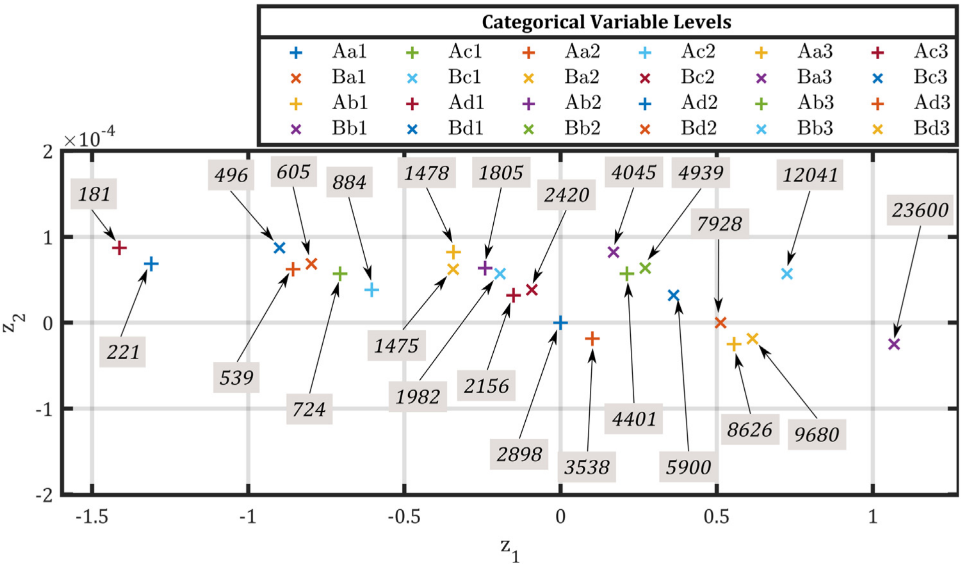


Fig. 4. Latent positions learned via LMGP for the column buckling example: Each latent position corresponds to a set of levels for each categorical variable in the borehole function. The first, second, and third categorical variables correspond to E, K, and I, respectively. Since the scale of the vertical axis is much smaller than that of the horizontal axis, we conclude that all the points lie almost on a horizontal line and hence this problem possess a single sufficient latent variable. The numbers in the gray boxes indicate the numerical values of $\frac{EI}{K^2}$ which monotonically increase from left to right. These numbers are not observed by LMGP which uses the mixed data in Fig. 2.

We now use a more complex and high-dimensional function to further demonstrate some of the nice properties of LMGPs. The borehole function [52] is ubiquitously used to assess the performance of surrogates. It is defined as:

$$y = \frac{2\pi T_u (H_u - H_l)}{\ln\left(\frac{r}{r_\omega}\right)\left(1+\frac{2LT_u}{\ln\left(\frac{r}{r_\omega}\right)r_\omega^2 K_\omega}+\frac{T_u}{T_l}\right)}. \tag{13}$$

The inputs of the borehole function are all quantitative. To have mixed variables, we convert T_l, L, and K_w into categorical variables with 5, 3, and 3 levels, respectively. Each level corresponds to a distinct numerical value *unknown* to LMGP (see Table 11 in the Appendix for details). The latent space positions estimated via LMGP are demonstrated in Fig. 5 where the legend shows the combination of levels (the triplets belong to T_l, L, and K_w, respectively) that corresponds to a point in the $2D$ latent space. Notice that the range of the axes is quite different and that the estimated latent positions are structured on a grid with eight vertical and five horizontal lines. On the vertical lines only the level of the first categorical variable changes (to see this, locate the markers with the same shape in the legend) while on the horizontal lines either the level of L or K_w changes. This figure suggests that the underlying function, while having 3 categorical variables, only depends on two hidden features. Furthermore, the range of the axes indicates that one of these hidden features affects the response, i.e., y in Eq. (13), more than the other. This relative importance of the two features is deduced from the term $-\left\|z\left(t\right)-z\left(t'\right)\right\|_2^2$ in Eq. (10): in Fig. 5, the hidden feature that is encoded by the horizontal axis has more variations and thus contributes more to this term. The higher contribution indicates that this feature affects the response more than the feature encoded by the vertical axis.

To relate the above insights with the underlying function, we rewrite the borehole function as:

$$y = \frac{C_1}{C_2\left(1+C_3\frac{L}{K_w}+\frac{C_4}{T_l}\right)} = f\left(\frac{L}{K_w}, T_l, C_1, C_2, C_3, C_4\right) \tag{14}$$

where the numerical variables are all lumped to C_1, C_2, C_3 and C_4. Eq. (14) clearly shows that the three original categorical variables can be compressed to two variables, namely, L/K_w and T_l. In fact, these two variables are the hidden features that LMGP learns purely based on the data. That is, in Fig. 5, one axis encodes T_l while the other

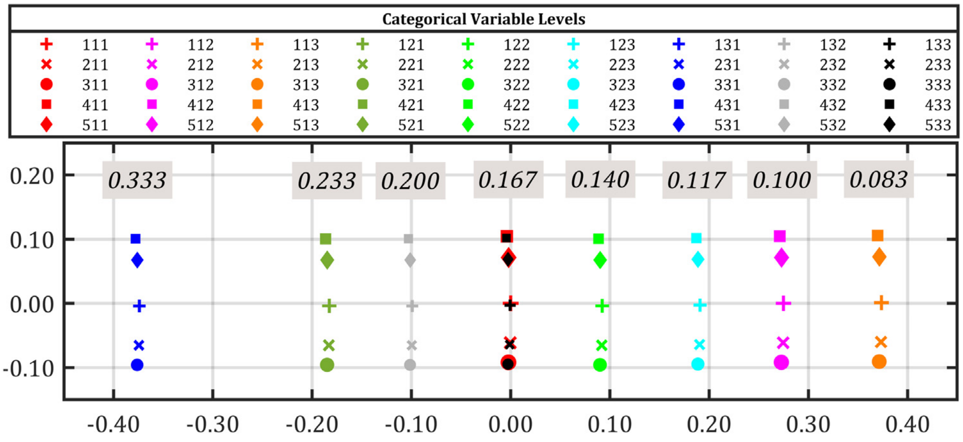


Fig. 5. Latent positions learned via LMGP for the borehole function: Each latent position corresponds to a set of levels for each categorical variable in the borehole function. The first, second, and third categorical variables correspond to T_l, L, and K_w, respectively. The points with the same level of T_l are structured approximately on a horizontal line. The points with the same levels of L and K_w are structured approximately on a vertical line. The underlying numerical value of L/K_w is indicated in the gray box on top of each vertical line.

Table 1
Underlying numerical values of L/K$_\mathbf{w}$: Values are reported for each combination of levels for L and K_w.

Level of L	1	1	1	2	2	2	3	3	3
Level of K_w	1	2	3	1	2	3	1	2	3
L/K_w	0.167	0.100	0.083	0.233	0.140	0.117	0.333	0.200	0.167

axis encodes L/K_w (note that the latter is a nonlinear function of the original variables L and K_w). Next, we find the correspondence between the axes of the latent space with L/K_w and T_l.

Table 1 lists the underlying numerical value of L/K_w for each combination of levels for L and K_w. By matching these numbers with the latent points in Fig. 5 it can be seen that each vertical line is associated with a unique number and that these numbers monotonically decrease from left to right. That is, the left- and right-most latent positions have, respectively, the largest ($L/K_w = 0.333$) and the smallest ($L/K_w = 0.083$) values. Notice that the ratio L/K_w is 0.167 when L and K_w both have a level of either 1 or 3 (see the first and last columns of Table 1). This situation is also accurately reflected in the latent space where the corresponding latent positions overlap (see the black and red markers in Fig. 5). The preceding discussions highlight that LMGP accurately discovers the latent feature that captures the collective effects of both L and K_w. This latent feature is encoded by the horizontal axis in Fig. 5 and thus the vertical axis encodes T_l.

While the latent representation along the horizontal axis in Fig. 5 is consistent with the ratio L/K_w, the same is not true for T_l. The underlying numerical values of T_l are organized in ascending order (they are [10, 30, 100, 200, 500], see Table 11). So, a monotonically ascending/descending order is expected for the levels of T_l in the latent space (The latent axis can have either an opposite or similar ordering as the underlying numerical values. In case of the horizontal axis, the positive direction is aligned with a reduction in L/K_w). That is, the levels of T_l on the horizontal lines from top to bottom in Fig. 5 should be either [1, 2, 3, 4, 5] or [5, 4, 3, 2, 1]. Instead, we see that the corresponding levels of T_l are ordered as 3, 2, 1, 5, and 4 (from bottom to top on each horizontal line). To better understand why LMGP seems to discover a sub-optimal latent representation for T_l, we employ Sobol sensitivity analysis [53].

Sobol sensitivity analysis is a method used for analyzing each input's total contribution to the output variance given the range of the inputs. The input's total contribution to the output variance can be decomposed into two parts: variance from each individual input and variance from interactions among inputs. Table 2 lists each input's total contribution to the output variance, referred to as the "total-effect index", for the borehole function in Eq. (13). The

Table 2
Total-effect index: The total-effect index is a metric that defines each input's total (individually and through interaction) contribution to the output variance. Unlike L and K_w, T_l almost has no effect on the variability of the response which makes it difficult to encode it in the latent space.

	T_u	H_u	H_l	r	r_w	T_l	L	K_w
Total-effect index	0.0000	0.0463	0.0465	0.0000	0.7445	0.0001	0.1290	0.1177

ranges used for the numerical features are described in Table 11, and the ranges used for the categorical variables are the maximum and minimum underlying numerical values which are also listed in Table 11.

Table 2 quantitatively indicates that T_l has a much smaller total-effect index than L or K_w. In fact, the extremely small sensitivity index of T_l signals that it almost has no effect on the response. This behavior is the reason that LMGP cannot find the correct latent representation for the varying levels of T_l. The sensitivity indices in Table 2 also explain why the range of the axes in Fig. 5 are quite different: since T_l negligibly affects the response, its contribution to the correlation function (through the term $-\left\|z(t)-z\left(t'\right)\right\|_2^2$ in Eq. (12) should be small which, in turn, requires small vertical distance between the latent points in Fig. 5.

4.4. Discussions and extensions

In this section, we first elaborate on the connections between NNs and LMGPs which can lead to further developments of LMGPs. Then, we compare LMGPs with sufficient dimension reduction and active subspaces. Finally, we discuss how to handle variable-length (or conditional) inputs with LMGPs.

4.4.1. Neural network interpretation of LMGP

LMGP can be perceived as an NN that encodes the categorical variables to a latent space as shown in Fig. 6. In this particular NN architecture, the latent space mapping in LMGP corresponds to a single hidden layer with linear activation functions and no bias terms (due to MLE's translation invariance). For this hidden layer, $\boldsymbol{\zeta}(\boldsymbol{t})$ is the input, $\boldsymbol{z}(\boldsymbol{t})$ is the output, and $\boldsymbol{A}$ represents the neural network weights.

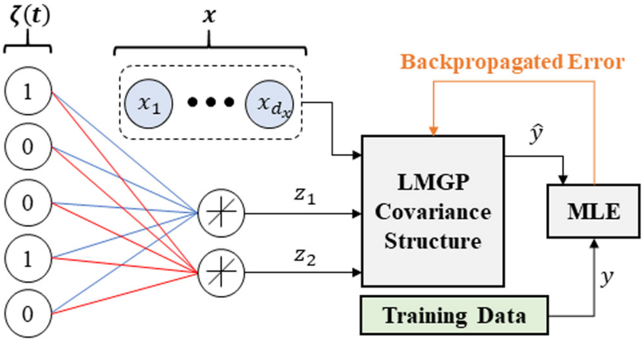


Fig. 6. Neural network interpretation of LMGP with a $2D$ latent space: Categorical data, $\boldsymbol{t}$, are converted to prior vector representations, $\boldsymbol{\zeta}(\boldsymbol{t})$, and fed into the network's hidden layer that maps $\boldsymbol{\zeta}(\boldsymbol{t})$ to $\boldsymbol{z}$ using linear activation functions and no bias. The LMGP covariance structure then uses $\boldsymbol{x}$ and $\boldsymbol{z}$ as inputs to approximate y.

With this interpretation, we can extend LMGP in a few ways. First, we can increase the number of hidden layers and their neurons to improve the learning capacity at the expense of increasing the number of hyperparameters that must be estimated. Second, we can use a nonlinear activation function (e.g., sigmoid, swish, or tangent hyperbolic) instead of a linear one. In our studies, we have observed marginal changes when testing the second idea but the integration of the two ideas may be more effective.

4.4.2. Active subspaces and sufficient dimension reduction

Active subspace methods use the gradient information for dimensionality reduction [54]. Assuming the gradient of the function is available at training points, one starts by rotating the input space (i.e., a linear transformation via singular value decomposition, SVD) to separate the directions based on their variability. The input space is then projected to the directions where the most variability is observed. Finally, surrogate modeling (with GPs or any other method) is done at this lower dimensional space which is called the active subspace of the underlying function.

LMGP is similar to active subspace methods in that they both rely on linear transformations. However, there are some fundamental differences between the two. First, unlike LMGP, active subspaces are not applicable to categorical data as the derivates are not defined. Second, while both methods reduce dimensions through some linear transformations, the underlying mechanism behind LMGP is different because it is supervised, relies on MLE (rather than SVD), and does not require gradient information (it is noted that some active subspace methods also do not require gradients, for example if a GP is used for metamodeling [55]).

Similar to active subspace methods, the core idea behind sufficient dimension reduction is to build a surrogate using a lower dimensional input space that is constructed via a linear transformation (SVD) of the original input space [56–58]. Sufficient dimension reduction does unsupervised dimension reduction (the responses may be used to slice the covariance matrices though) and cannot handle categorical inputs.

4.4.3. Variable-length categorical inputs

Both standard LMGP and LVGP cannot accept variable-length inputs. However, LMGP can be easily modified to handle variable-length inputs. We demonstrate this using the zip code-course subject example where $t_1 = \{92697, 92093\}$ and $t_2 = \{math, physics, chemistry\}$. Assume that when $t_1 = 92093$, t_2 is no longer an input, i.e., the system's response is independent of t_2 if $t_1 = 92093$. This conditional situation is illustrated in Table 3 where NaN indicates that the categorical variable is not an input (or if it is an input, it does not affect the system's response).

Table 3
Combinations of levels for the variable-length example: All combinations of levels for the first and second categorical variable are listed. When the level is NaN, that categorical variable is not an input given the level of the other categorical variable.

First categorical variable	Second categorical variable
1	1
1	2
1	3
2	NaN

To make LMGP compatible with variable-length categorical inputs, only the prior vector representations need to be adjusted. Applying the $1-0$ representation described in Section 4.1 to the current example results in:

$$\begin{bmatrix} \boldsymbol{\zeta}(a=1,b=1) \\ \boldsymbol{\zeta}(a=1,b=2) \\ \boldsymbol{\zeta}(a=1,b=3) \\ \boldsymbol{\zeta}(a=2,NaN) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & NaN & NaN & NaN \end{bmatrix}$$

where $\boldsymbol{\zeta}(a,b)$ is the unique vector representation when the first and second categorical variables are at levels a and b, respectively Because NaN is not a valid value it must be replaced with a number. For this, we propose two potential approaches. One strategy is to replace NaN values with IID random numbers so $\boldsymbol{\zeta}(\boldsymbol{t})$ for each combination of levels of the categorical variables becomes:

$$\begin{bmatrix} \boldsymbol{\zeta}(a=1,b=1) \\ \boldsymbol{\zeta}(a=1,b=2) \\ \boldsymbol{\zeta}(a=1,b=3) \\ \boldsymbol{\zeta}(a=2,NaN) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & \chi & \chi & \chi \end{bmatrix}, \ \chi \sim Uni(0,1)$$

Another strategy is to simply replace all NaN values with 0. In Section 5.3, we see through testing that both strategies yield very similar performance.

Table 4
List of analytical functions: The functions possess a wide range of dimensionality and complexity. When emulating each function, we use training datasets of sizes 100, 200, 300,and 400. We also add IID normal noise to both training and test data. Three noise variances are considered for each function. The smallest variance is 0 in all cases while the other two depend on the range of the function.

ID-Name [Ref]	Function
1− OLT Circuit [59]	$y = \frac{(V_{b1}+0.74)\,\beta\,(R_{c2}+9)+11.35R_f}{\beta\,(R_{c2}+9)+R_f} + \frac{0.74R_f\beta\,(R_{c2}+9)}{\left(\beta\,(R_{c2}+9)+R_f\right)R_{c1}}$ $V_{b1} = \frac{12R_{b2}}{R_{b1}+R_{b2}}$
2− Piston Simulator [59]	$y = 2\pi\sqrt{\frac{M}{k+S^2\frac{P_0V_0T}{T_0V^2}}}$ $V = \frac{S}{2k}\sqrt{A^2+4k\frac{P_0}{T_0}T}, \qquad A = P_0S+19.62M-\frac{kV_0}{S}$
3− Borehole [52]	$y = \frac{2\pi T_u(H_u-H_l)}{\ln\left(\frac{r}{r\omega}\right)\left(1+\frac{2LT_u}{\ln\left(\frac{r}{r\omega}\right)r_\omega^2K_\omega}+\frac{T_u}{T_l}\right)}$
4− Effective Potential [60]	$y = 100{*}\frac{9}{2}x_9\varepsilon_m^2 + \frac{x_8x_{10}}{1+x_7}\left[\frac{\varepsilon_{eq}}{x_{10}}\right]^{1+x_7}$ $\boldsymbol{\varepsilon} = \begin{pmatrix} x_1 & x_6 & x_5 \\ x_6 & x_2 & x_4 \\ x_5 & x_4 & x_3 \end{pmatrix}, \qquad \varepsilon_m = \frac{1}{3}Tr\,(\boldsymbol{\varepsilon})\,, \varepsilon_d = \boldsymbol{\varepsilon} - \varepsilon_m 1, \varepsilon_{eq} = \sqrt{\frac{2}{3}\,(\varepsilon_d : \varepsilon_d)}$
5− Wing Weight [61]	$y = 0.036S_\omega^{0.758}W_{f\omega}^{0.0035}\left(\frac{A}{cos^2(\Lambda)}\right)^{0.6}q^{0.006}\lambda^{0.04}\left(\frac{100t_c}{\cos(\Lambda)}\right)^{-0.3}\left(N_zW_{dg}\right)^{0.49}+S_\omega W_p$
6− Custom Function [62]	$y = 4\left(x_1-2+8x_2-8x_2^2\right)^2+(3-4x_2)^2+16\sqrt{x_3+1}\,(2x_3-1)^2+\sum_{i=4}^{8} iln\left(1+\sum_{j=3}^{i}x_j\right)$

5. Results

In this section, we compare the performance of LMGP against LVGP and also apply LMGP to two variable-length problems. We do not compare LMGP with the other methods reviewed in Section 3 as LVGP is shown to consistently outperform them [32]. Both algorithms are coded in Matlab and leverage continuation [35] to estimate the optimum nugget variance. More algorithmic details on both LMGP and LVGP are provided in the Appendix, see Appendix A.1.

In Section 5.1, we compare LMGP to LVGP using six analytical functions with various sample sizes, noise levels, and number of categorical variables. In Section 5.2, we apply both methods to two real-world datasets and in Section 5.3 we analyze the performance of LMGP on handling variable-length categorical inputs. Finally, in Section 5.4 we use LMGP in Bayesian optimization for identifying the compound composition that maximizes the bulk modulus.

5.1. Analytical functions

Table 4 summarizes the analytical functions used for comparing LMGP to LVGP. Since these functions only have numerical variables, we modify them by converting a few numerical features into categorical features. That is, each level of the categorical variables corresponds to an underlying numerical value unknown to LMGP and LVGP. The underlying numerical values for the categorical variable levels for each analytical function are listed in the Appendix, see Appendix A.2. These analytical functions are chosen as they have a wide range of dimensionality and degree of nonlinearity. Additionally, the conversion of numerical variables to categorical ones allows to have multiple categorical variables with many levels, see Table 5.

For testing, we compare the performance across various training dataset sizes (ranging between 100 to 400 samples) and noise levels, which varies based on the range of the analytical function. After fitting LMGP and LVGP, we then evaluate the mean squared error (MSE) across 10,000 test samples. Training and validation input samples are (for both quantitative and categorical variables) generated via Sobol sequence [63–65]. To account for

Table 5
Input descriptions: The numerical (in red) and categorical (in blue) inputs, their ranges, and the number of level combinations, b_t, are listed for each function.

ID	Variables (Numerical, Categorical)	Min, Max	b_t
1	$R_{b1}, R_{b2}, R_f, R_{c1}, R_{c2}, \beta$	$[1, 50, 1, 1.2, 0.01, 1]$, $[3, 70, 3, 2.5, 5, 3]$	27
2	$M, S, V_0, k, P_0, T, T_0$	$[1, 1, 1, 2000, 2E5, 10, 10]$, $[3, 3, 3, 3000, 1.5E6, 500, 760]$	27
3	$T_u, H_u, H_l, r, r_w, T_l, L, K_w$	$[100, 990, 700, 100, 0.05, 1, 1, 1]$, $[1000, 1110, 820, 1E4, 0.15, 5, 3, 3]$	45
4	$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}$	$[0, 0, 0, 0, 0, 0, 1, 1, 1, 1]$, $[1, 1, 1, 1, 1, 1, 5, 5, 5, 5]$	625
5	$S_w, W_{fw}, A, \Lambda, q, \lambda, t_c, N_z, W_{dg}, W_p$	$[1, 1, 6, -10, 16, 0.5, 1, 2.5, 1, 0.025]$, $[3, 3, 10, 10, 45, 1, 3, 6, 3, 0.08]$	81
6	$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$	$[0, 0, 1, 1, 0, 0, 0, 1]$, $[1, 1, 6, 4, 1, 1, 1, 3]$	72

randomness and measure consistency, the training and validation processes are repeated 10 times with a new dataset each time.

Fig. 7 summarizes the MSE results and indicates that LMGP consistently outperforms LVGP across all noise levels for large training datasets. In particular, LMGP achieves a test MSE on noisy data that is very close to the noise variance which indicates that it is able to extract as much information from the data as possible. With small datasets, LVGP often outperforms LMGP and is more robust. We believe this is due to the fact that LMGP has more hyperparameters than LVGP and hence needs more data. Additionally, we note that in two cases (borehole and wing weight functions), increasing the training dataset size decreases LVGP's performance which is unintuitive because the performance is generally expected to improve once more data are used in training. This unintuitive behavior of LVGP is due to its failure in distinguishing noise from variables that negligibly affect the response. For instance, in the borehole function, one of the categorical variables, T_l, insignificantly affects the response, as evidenced by its Sobol index in Table 2. In this case, LVGP is mistakenly interpreting the variations in y that are rooted in T_l to be due to noise. This non-identifiability issue is not seen in LMGP as it is able to distinguish noise from categorical variables that marginally affect the response.

Fig. 8 shows the estimated noise variance via LVGP and LMGP across all the functions when 400 training samples are used. From the boxplots, we see that LMGP more accurately estimates the noise variances in all cases except for the OLT circuit model (although the difference is very small in this case). When LVGP is less accurate, noise estimates are off quite noticeably. We believe that the combination of embedding prior information on the categorical variable levels and using a single latent space for all categorical variables allows LMGP to discover a more general solution, making noise estimation via MLE easier.

In terms of computational costs, LVGP is more efficient than LMGP with small datasets. However, as the size of the training dataset or the variance of the added noise increase, the computational performance of LMGP gains advantage over LVGP. These trends are illustrated in Fig. 9 for the borehole function (other functions exhibit very similar trends, see Appendix A.3 and we explain them as follows. The total number of hyperparameters that must be optimized using MLE for LMGP and LVGP are $d_x + 2 \times \sum_{i=1}^{d_t} m_i$ and $d_x + \sum_{i=1}^{d_t} (2m_i - 3)$, respectively (assuming both approaches use $d_z = 2$). With small data, this difference in the number of optimization parameters dominates the computational costs. However, as either the noise variance or the number of samples increase, LMGP is better

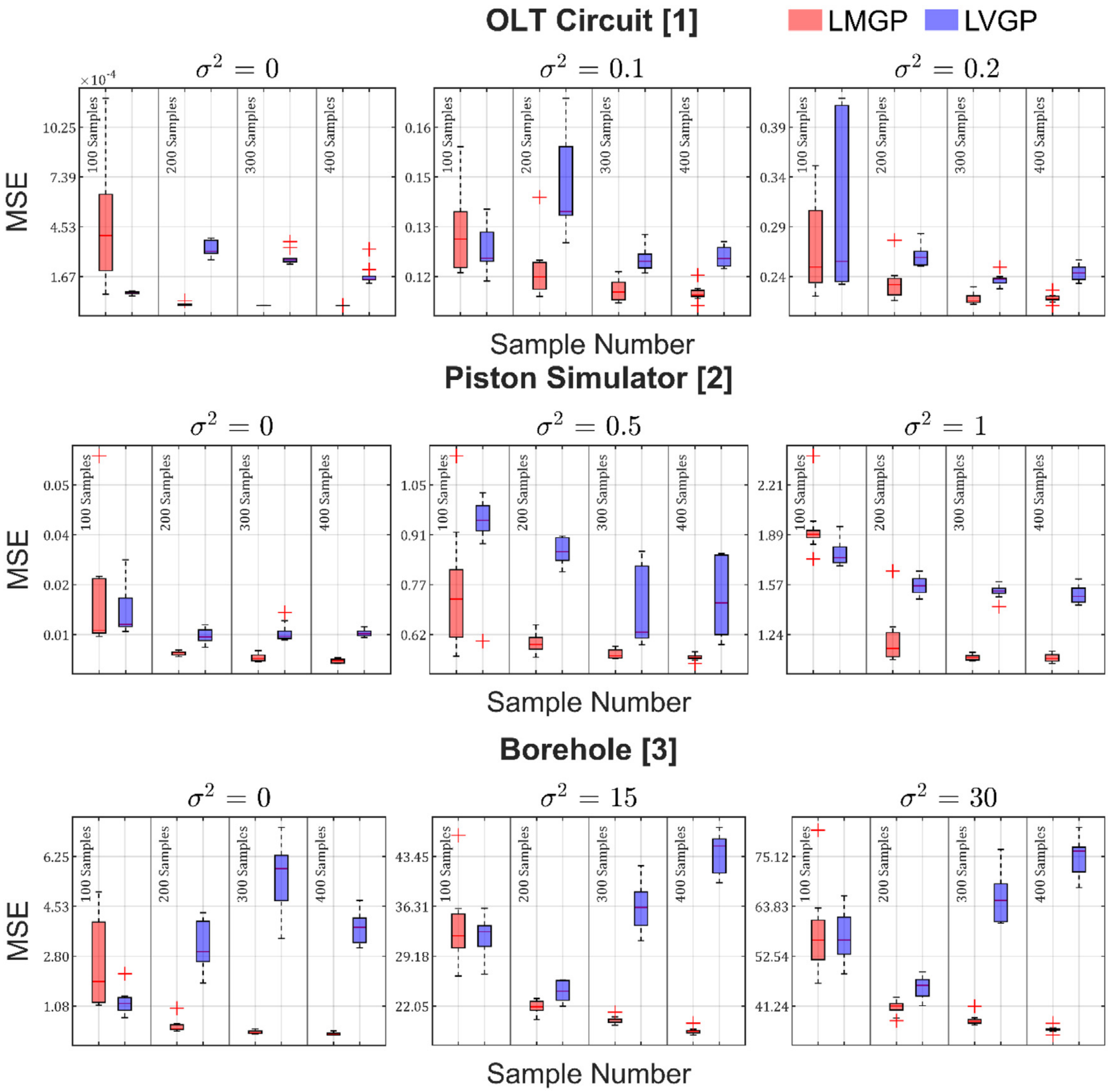


Fig. 7. Results on analytical functions: We compare LMGP to LVGP across six different analytical functions. For each case, LMGP and LVGP are fitted to datasets of sizes 100, 200, 300, and 400 with three different noise levels (one noise level being 0 and the other two depending on the range of the analytical function). 10,000 noisy test data points are used to obtain MSE. The training and validation process are repeated 10 times to account for randomness.

able to distinguish between the required nugget parameter and the latent positions and hence gains some speedups over LVGP.

Compared to other metamodels such as NNs and random forests, the overall training costs of both LMGP and LVGP are much lower in applications with small to medium size and dimensionality. For instance, as opposed to LVGP and LMGP, an NN requires tuning the architecture and optimization settings (e.g., learning rate) which can be time-consuming.

5.2. Real-world datasets

In this section, we compare the performance of LMGP, LVGP, and standard GP across two datasets: the Boston housing dataset [66] and auto-MPG dataset [67]. In the former dataset, the goal is to predict the median housing

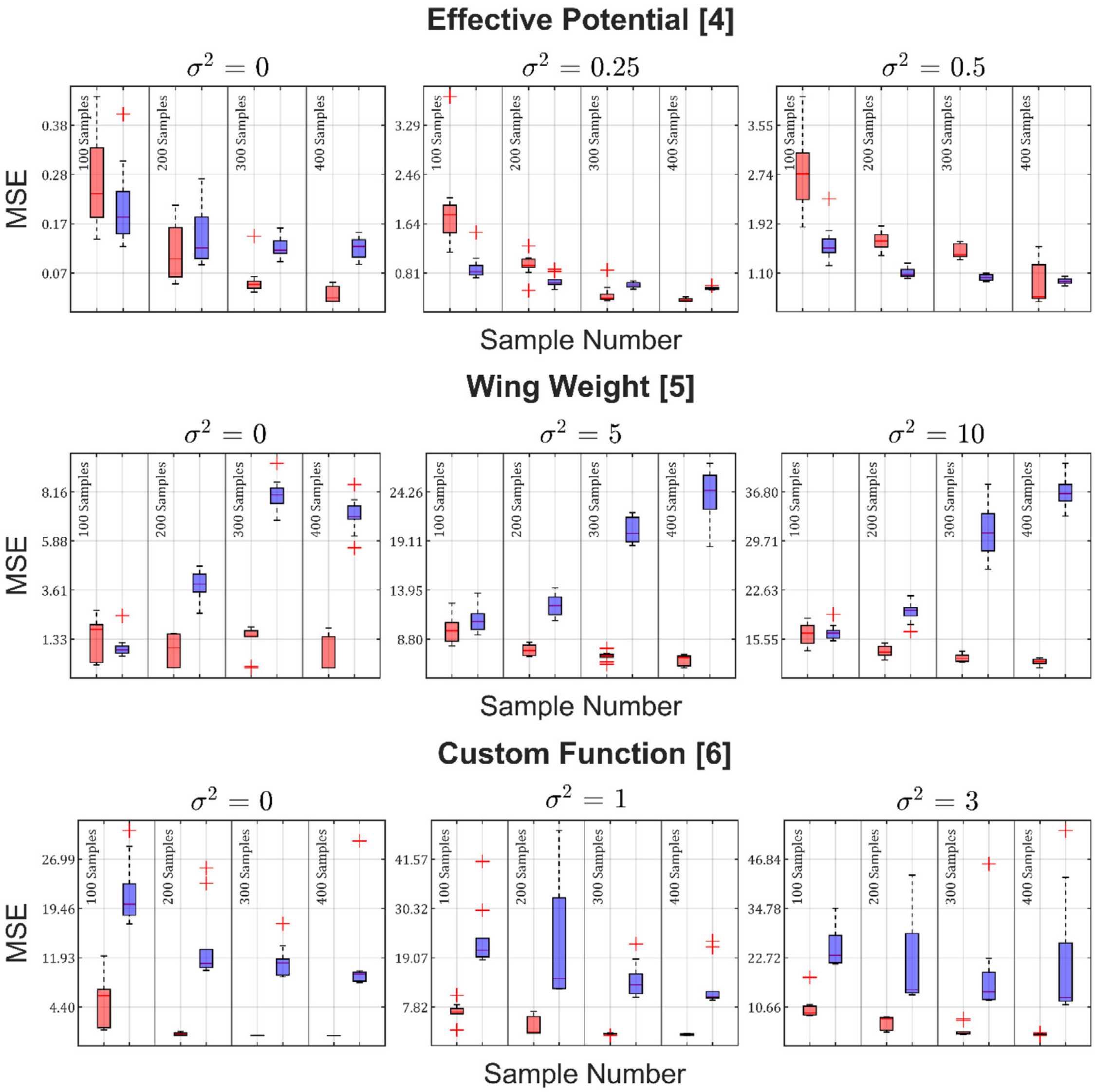


Fig. 7. (*continued*).

prices in the suburbs of Boston in 1978. The dataset has 506 samples, 13 inputs, and one output. Since the output is capped at 50, the output of some samples is not trustworthy. Hence, we remove samples whose output is 50. As a result, the number of samples is reduced to 503. For LMGP and LVGP, CHAS (Charles River dummy variable) and RAD (index of accessibility to radial highways) are treated as categorical inputs. For GP, all inputs, all inputs are treated as numerical. The dataset is randomly split into 70% training and 30% validation. To account for randomness, the comparison test is performed 10 times.

In the auto-MPG dataset, the goal is to predict the MPG of various cars based on 8 inputs. After removing samples with missing values, 392 samples are available. For LMGP and LVGP, the number of cylinders and origin (i.e., the country the car was built in) are treated as categorical inputs while all inputs are treated as numerical for GP. The dataset is randomly divided into 50% training and 50% validation, and the comparison test is performed 10 times.

Table 6 summarizes the results which show that LMGP slightly outperforms LVGP. Both models perform better GP, especially with the auto-MPG dataset. LMGP and LVGP perform similarly in these two datasets for the following reasons: (*i*) Both datasets only have two categorical variables where none of them has many levels,

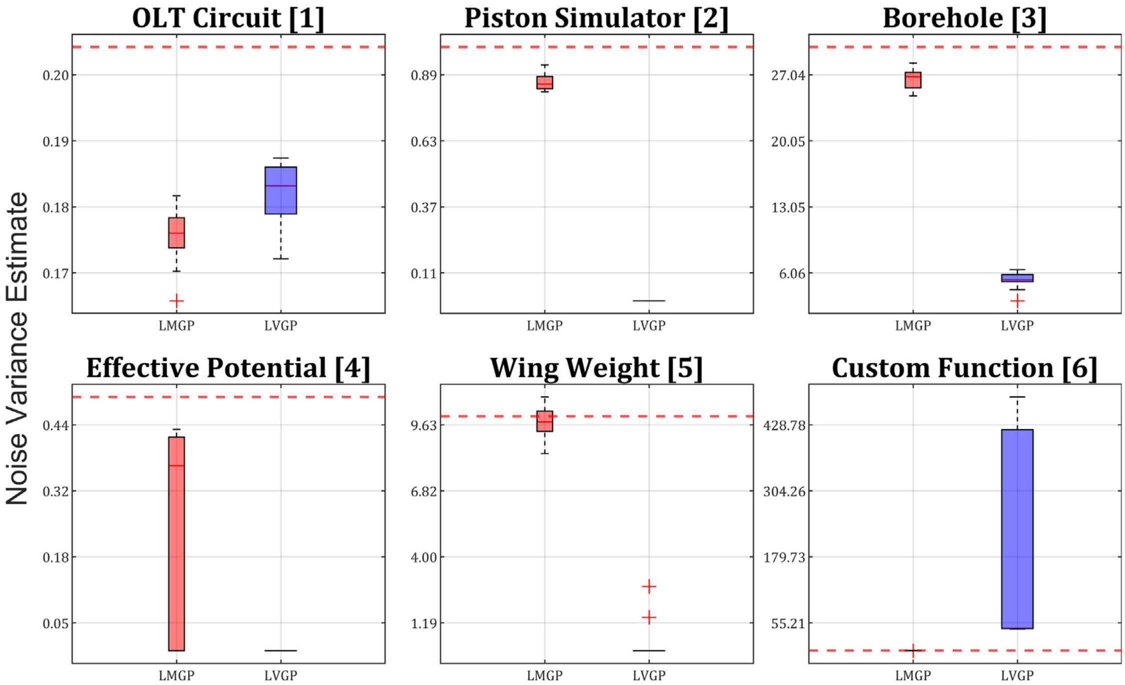


Fig. 8. Estimated noise variance: Each time LVGP and LMGP are fitted, the noise variance is estimated using the training data. For each analytical function, 400 training samples are used with the injected noise levels indicated by the dashed red line in each subplot. It is clear that on average LMGP estimates the noise variance more accurately (narrower box plots located close to the horizontal red dashed line indicate better performance).

Table 6
Results on real world datasets: The performance of LMGP, LVGP, and GP are analyzed across two real-world datasets. For each dataset, 10 different permutations of training and validation subsets are used, and the mean (μ_{MSE}) and standard deviation (σ_{MSE}) of mean squared error (MSE) is reported.

	Boston Housing		**Auto-MPG**	
	μ_{MSE}	σ_{MSE}	μ_{MSE}	σ_{MSE}
LMGP	**7.166**	**1.060**	**7.934**	1.520
LVGP	7.371	1.408	8.241	**1.427**
GP	8.900	1.847	19.416	22.845

(ii) the categorical variables have little interaction, and (iii) at least one of the categorical variables has little effect on the response. GP's poor performance on the auto-MPG dataset is likely because the input, origin, should not be treated as a numerical input while the other categorical variables have some justification for being treated as numerical features. We note that the performance of LMGP and LVGP is either better or comparable to that of state-of-the-art NNs fitted to these datasets [68–71]. Unlike NNs, however, neither LMGP nor LVGP require iterative adjustment of, e.g., architecture, learning rate, or epoch number. In other words, training LMGP and LVGP is much simpler on such a small to medium size dataset.

5.3. LMGP for variable-length categorical inputs

In this section, we analyze the performance of LMGP for handling variable-length categorical inputs. To this end, we reuse the borehole and OLT circuit functions defined in Table 4 with some modifications. In particular, we remove certain categorical variables based on the level of other categorical variables. As shown in Table 7, we

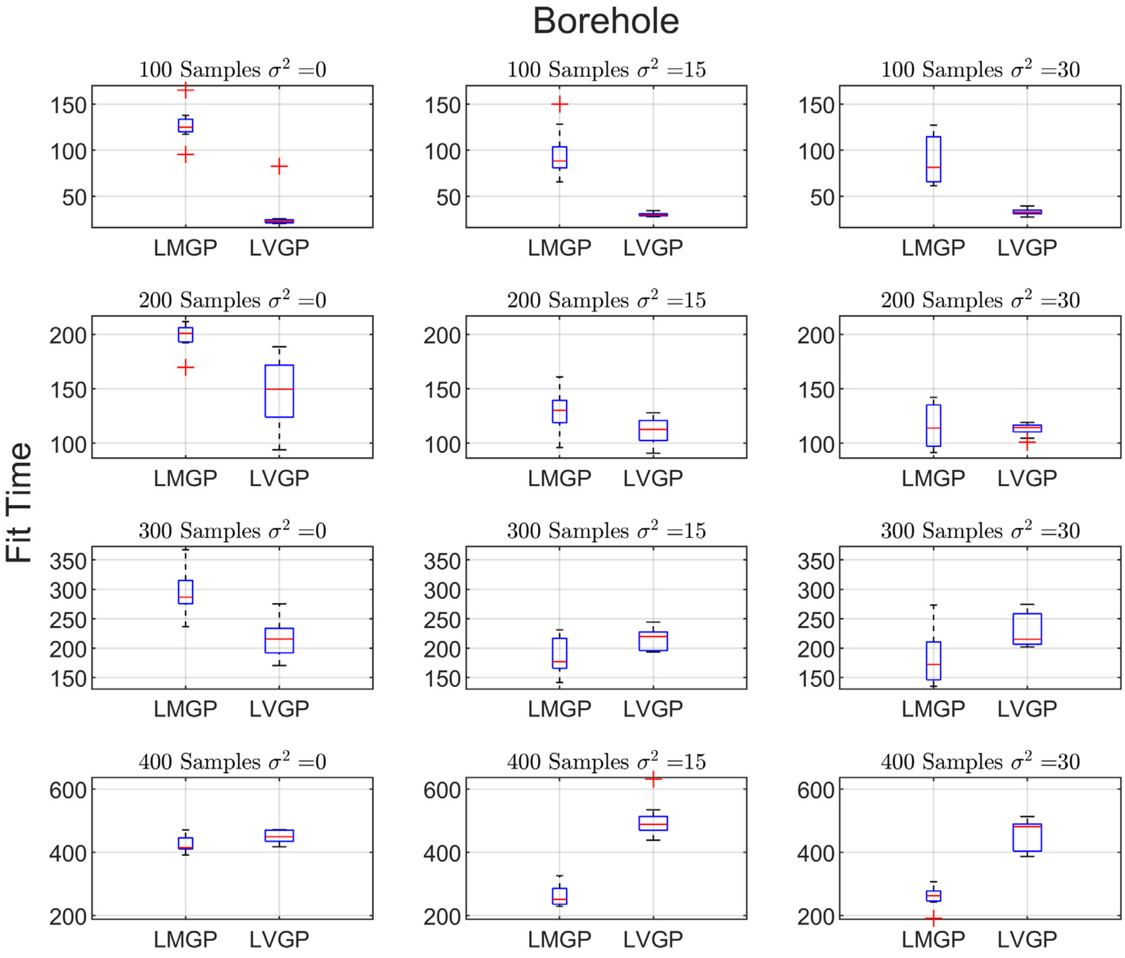


Fig. 9. Training costs for the borehole function: The rows and columns are organized based on training dataset size and noise variance, respectively.

Table 7
Combinations of levels: Listed are all cases, for the borehole and OLT circuit model, when one of the categorical variables is not an input (i.e., is NaN) given the level of the other categorical variables.

Levels of Cat. Variable 1 T_l (borehole) and R_{b1} (OLT)	Levels of Cat. Variable 2 L (borehole) and R_f (OLT)	Levels of Cat. Variable 3 K_w (borehole) and β (OLT)
1	1	NaN
2	NaN	2
NaN	3	3

remove (i) the third categorical variable when the first two categorical variables are at their respective level one, (ii) the second categorical variable when the first and third categorical variables are at their respective level two, and (iii) the first categorical variable when the last two categorical variables are at their respective level three. Regarding the underlying function, when the categorical variable is NaN, we set the variable to a numerical value as if it were another level (see Table 8). Note that this value is unknown to LMGP and not used in any way during the training.

Table 8
Underlying numerical values: When a categorical variable's level is set to NaN (i.e., the cases listed in Table 7), the underlying analytical function simply sets the categorical variable to an underlying numerical value while LMGP treats the variable as if it is no longer an input.

	Borehole model			OLT circuit model		
	T_l	L	K_w	R_{b1}	R_f	β
Underlying value	350	1100	8000	35	1	2

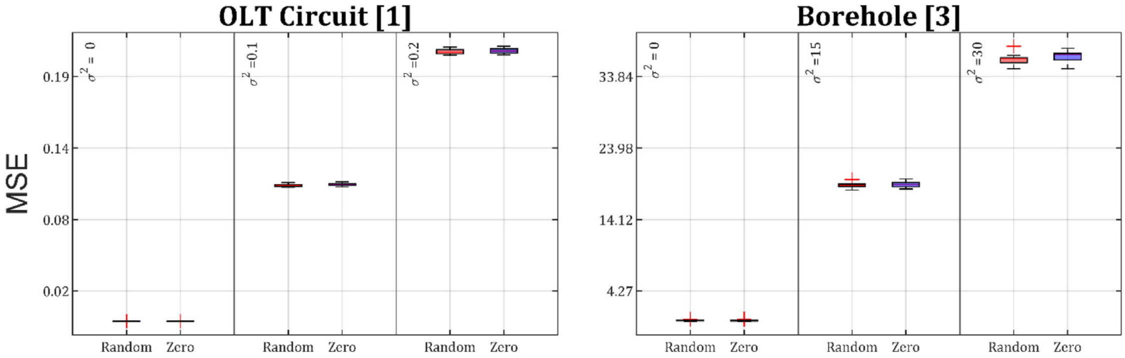


Fig. 10. Comparing LMGP methods for handling variable-length inputs: Using the borehole and OLT circuit model, we compare predictive performance between two methods of handling variable-length inputs: The random and zero approach. The strategy of setting NaN to a random value, χ, and 0 will be denoted as the zero and random approach, respectively. Both methods perform similarly with an MSE close to the injected noise variances.

For each function, we randomly generate 400 training samples and 10,000 test samples following the data generation strategy described in Section 5.1. We add IID normal noise with different variances (see Fig. 10) to both training and test data. We then fit LMGP to the training data and evaluate it on the test data. To account for randomness, the procedures are repeated 10 times. Fig. 10 summarizes the results and indicates that both strategies described in Section 4.4.3 have similar performance. In particular, they both achieve MSE on test data relatively close to the applied noise, implying that the "Zero" and "Random" approaches are not causing significant losses in prediction performance.

5.4. Material design with Bayesian optimization

In this section, we apply LMGP to a material design problem previously studied in [72] where the goal is to use as few data points as possible (from a dataset of size 240 [73]) to find the elements in the family of M_2AX compounds that maximize bulk modulus. These compounds are nanolaminate ternary alloys that exhibit many of the beneficial properties of both ceramic and metallic materials and hence are appealing for many technological applications [73–75]. The family of M_2AX compounds have three building blocks that can take on different elements: an early transition metal $M = \{Sc, Ti, V, Cr, Zr, Nb, Mo, Hf, Ta, W\}$, a main group element $A = \{Al, Si, P, S, Ga, Ge, As, Cd, In, Sn, Tl, Pb\}$, and either carbon or nitrogen $X = \{C, N\}$.

A direct strategy for finding the optimal compound is to compute the bulk modulus for all the $10 \times 12 \times 2 = 240$ candidates via density functional theory (DFT). However, this approach is suboptimal because DFT is computationally expensive. An alternative strategy is to use Bayesian optimization (BO) to discover the optimum compound by only obtaining the modulus of some of the candidates via DFT. A generic BO framework starts by fitting a probabilistic predictive model to an initial training data. Then, this model is used in an acquisition function that balances exploration and exploitation to identify the next candidate that must be evaluated and added to the training data. This three-step iterative process (that consists of model training, evaluation of acquisition function, and updating the training data) continues until the convergence criterion is met (e.g., resources are exhausted).

In BO, the user chooses the acquisition function and model type [76–79]. In this paper, we use expected improvement (EI) for the acquisition function defined as:

$$EI(\boldsymbol{x}) = E\left[max\left(y_{max} - \hat{y}(\boldsymbol{x}), 0\right)\right], \tag{15}$$

where $E[\cdot]$ denotes the expectation operator, y_{max} is the best candidate in the current training dataset, and $\hat{y}(\boldsymbol{x})$ is the prediction for candidate $\boldsymbol{x}$. If the model's prediction has a normal distribution with mean $\mu(\boldsymbol{x})$ and variance $\sigma(\boldsymbol{x})$, Eq. (15) takes on the following closed form formula:

$$EI(\boldsymbol{x}) = (y_{max} - \mu(\boldsymbol{x}))\,\Phi\left(\frac{y_{max}-\mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})}\right) + \sigma(\boldsymbol{x})\,\phi\left(\frac{y_{max}-\mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})}\right), \tag{16}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ denote, respectively, the cumulative distribution and probability density functions of the standard normal distribution, $\mathcal{N}(\mu = 0, \sigma = 1)$.

Choosing the type of the predictive model is a challenge for this design optimization problem because all inputs are categorical where each categorical variable corresponds to a site (M, A, or X) and the levels represent potential elements for each respective site (e.g., C or N for site X). The strategy adopted in [72] is to use domain knowledge to convert categorical variables into quantitative inputs which can then be used in a standard GP. In particular, in this strategy each element (e.g., Sc, Al, or C) is characterized with its orbital radii (s-, p-, and d-orbital radii for elements at site M while s- and p-orbital radii for A and X sites[2]) which, in turn, converts a candidate compound into a $7D$ quantitative variable.

Unlike the strategy of [72], LMGP can be directly applied to the original dataset which eliminates the time-consuming and problem dependent feature engineering step. This is advantageous because the replaced numerical variables may not sufficiently represent the effects of changing an element in one of the three sites. To demonstrate this benefit, we compare standard GP (with the categorical variables replaced with the $7D$ numerical variables) to LMGP when they are used as the predictive model in BO. In particular, we exclude the compound with the largest bulk modulus from the original dataset and then start the BO with randomly selected 20 compounds. We continue taking samples from the original dataset, one by one as guided by the acquisition function in Eq. (16), until the best compound is found. To compare GP vs. LMGP, we record the number of additional samples that BO evaluates until convergence. To account for the randomness, we repeat this process 30 times where each time a unique set of initial compounds is used.

Fig. 11 is a histogram of the number of additional samples needed before finding the optimal compound which indicates that, on average, LMGP and GP require sampling 16.38 and 18.13 additional compounds, respectively. Thus, LMGP is more likely to find the optimal compound earlier than standard GP. We believe this is because the numerical features chosen for standard GP do not sufficiently capture the effects of switching elements for each site. LMGP does not assume the underlying numerical variables are solely defined by the orbital radii and thus, it is more flexible. Furthermore, we emphasize that LMGP did not require domain knowledge to identify the underlying numerical variables. This eliminates the need for feature engineering and makes our strategy very desirable for materials design and analysis where the underlying numerical features are not even known by domain experts. Lastly, we note that similar to LMGP, LVGP can also be directly applied to this material design problem. As illustrated in Fig. 11, the performance of both methods is very close and both outperform GP.

We close this section by noting that, unlike LVGP, LMGP can benefit from domain knowledge in a straightforward setting. For instance, in this materials design example, the grouped one-hot encoded prior vectors, that is $\boldsymbol{\zeta}(\boldsymbol{t})$, can be replaced with quantitative vectors whose elements are selected based on s-, p-, and d-orbital radii (or any other features deemed to characterize the differences between different elements). Note that this approach is different than a Bayesian approach where prior distributions are placed on the parameters (both LVGP and LMGP will benefit from a Bayesian implementation where in LMGP priors will be placed on the $\boldsymbol{A}$ matrix while in LVGP the priors will be placed on the latent positions). We will investigate these approaches in our future works.

6. Conclusion

In this paper we introduced LMGPs which are extensions of GPs that can build surrogates with quantitative and qualitative inputs. As we showed, the main idea behind LMGPs is to learn a linear map that converts each combination of qualitative inputs to a point in a low-dimensional latent space. Since these latent points are endowed

[2] It is unclear why d-orbital radius is not used for A and X sites.

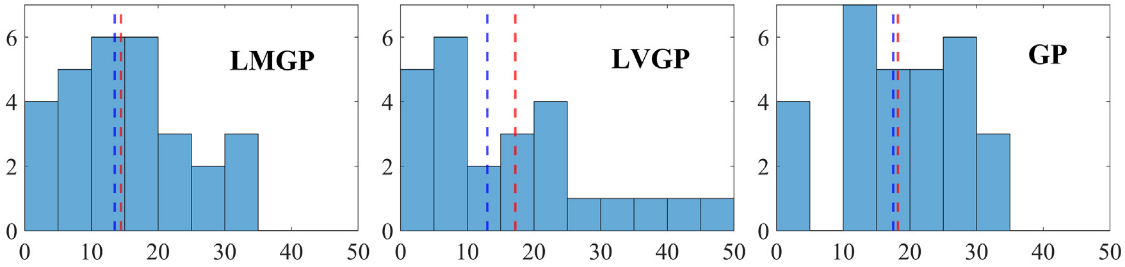


Fig. 11. Results of Bayesian optimization: We compare the performance of LMGP, LVGP, and standard GP across 30 tests, each with a unique initial dataset of 20 compounds. The histograms indicate the number of additional compounds required to sample before finding the compound with the highest bulk modulus (smaller is better). The average and median number in each case is shown with, respectively, the blue and red dashed vertical lines.

with an automatically learned distance measure, they can be directly used in any standard correlation function such as the Gaussian or Matérn.

We estimated the optimal linear map simultaneously with other hyperparameters by maximizing the Gaussian likelihood function. Alternatively, a Bayesian approach can be used to find the posterior distribution of LMGP's linear map. We have not pursued this in our studies yet.

By interpreting the linear map as an operator that projects all prior latent representations to their corresponding posteriors, we studied the effect of priors on LMGP. We showed that an informative prior consisting of grouped one-hot encoded inputs helps LMGP in building well-structured latent spaces and maximizes the performance on test data. Other types of priors may be more useful in applications where the fitted LMGP has to satisfy some physical constraints or where there is some prior knowledge on how qualitative inputs are related.

LMGPs can be interpreted as neural networks with certain architecture and activation functions. This interpretation opens up interesting possibilities that we will study in our future works. For instance, (i) the current linear map can be converted to a highly nonlinear one by adding hidden layers with nonlinear activation functions (e.g., sigmoid or swish activations), (ii) physics-informed LMGP can be built by infusing governing dynamics to the loss function, or (iii) very high-dimensional inputs–outputs can be handled by using convolutional layers.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank the original authors of LVGP for providing their Matlab code to compare the performance of LMGP to LVGP. We also thank the anonymous reviewers as well as Professor Daniel Apley for his detailed feedback on LMGP. This work was supported by the Advanced Research Projects Agency-Energy, USA (ARPA-E), U.S. Department of Energy, under award number DE-AR0001209.

Appendix

A.1. Selection of LVGP and LMGP parameters

For LVGP, the hyperparameter ranges are limited to: $\omega_i \in [-8, 3]$, $z_j^i(t_i) \in [-5, 5]$, and $\delta \in [0.1, 1E-10]$. For LMGP, the hyperparameter ranges are limited to: $\omega_i \in [-8, 3]$, $A_{i,j} \in [-1, 1]$, and $\delta \in [0.1, 1E-10]$. Both LMGP and LVGP estimate all the hyperparameters via a gradient-based optimization approach that starts the search via 12 initial, randomly selected points.

A.2. Underlying numerical values

In this section, we list the underlying numerical value associated with the different levels of each categorical variable for the analytical functions introduced in Section 5.1 (see Tables 9–14).

Table 9
Underlying numerical values: OLT circuit model [1].

Categorical variable level	R_{b1}	R_f	R_f
1	25	0.5	1
2	32.5	2	4
3	40	3	5

Table 10
Underlying numerical values: Piston Simulator Model [2].

Categorical variable level	M	S	V_0
1	30	0.005	0.002
2	40	1	0.4
3	50	2	1

Table 11
Underlying numerical values: Borehole Model [3].

Categorical variable level	T_l	L	K_w
1	10	1000	6000
2	30	1400	10000
3	100	2000	12000
4	200	N/A	N/A
5	500	N/A	N/A

Table 12
Underlying numerical values: Effective Potential Model [4].

Categorical variable level	x_7	x_8	x_9	x_{10}
1	0.1	1	5	0.01
2	0.25	2	10	0.02
3	0.7	4	12.5	0.1
4	0.8	9	25	0.3
5	1	10	30	0.5

Table 13
Underlying numerical values: Wing Weight Model [5].

Categorical variable level	S_w	W_{fw}	t_c	W_{dg}
1	150	220	0.08	1700
2	180	250	0.12	2000
3	200	300	0.18	2500

Table 14
Underlying numerical values: Custom Function [6].

Categorical variable level	x_3	x_4	x_8
1	0	0	0
2	0.1	0.2	0.4
3	0.3	0.7	1
4	0.6	1	N/A
5	0.7	N/A	N/A
6	1	N/A	N/A

A.3. Training costs

See Figs. 12–14.

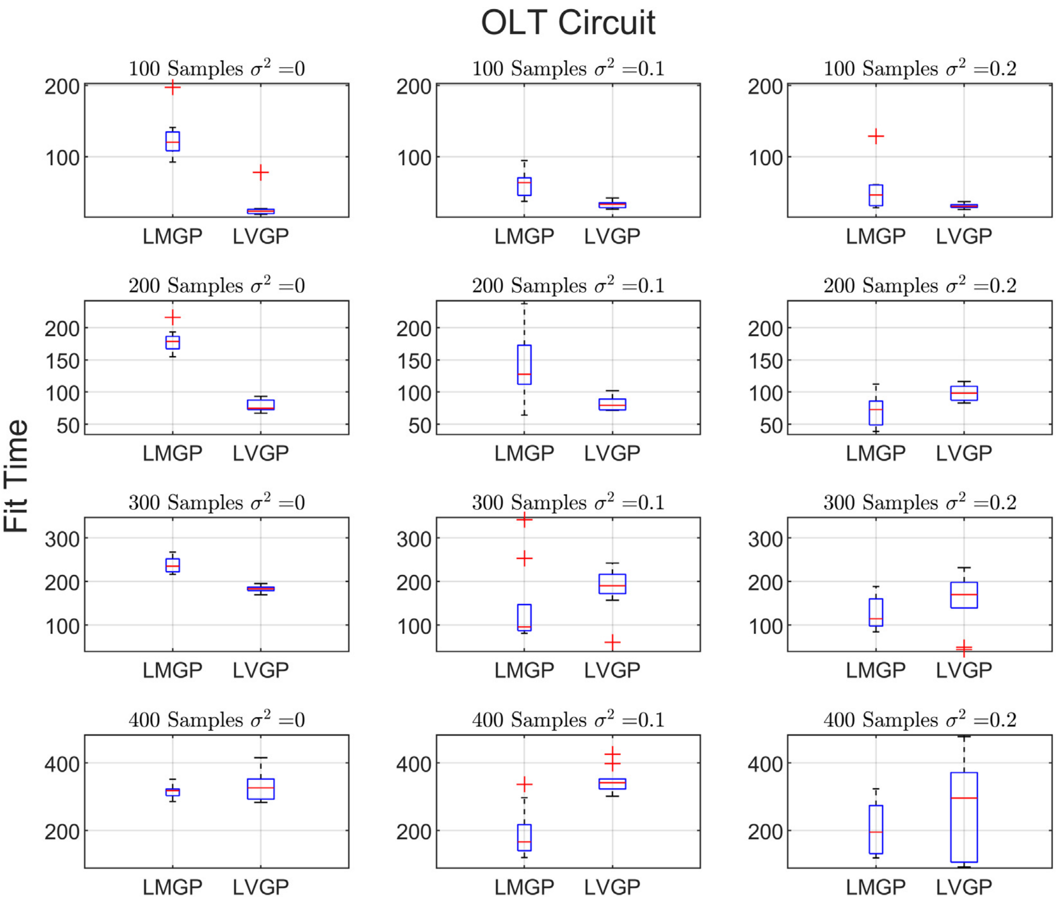


Fig. 12. Training costs for the OLT circuit function: The rows and columns are organized based on training dataset size and noise variance, respectively.

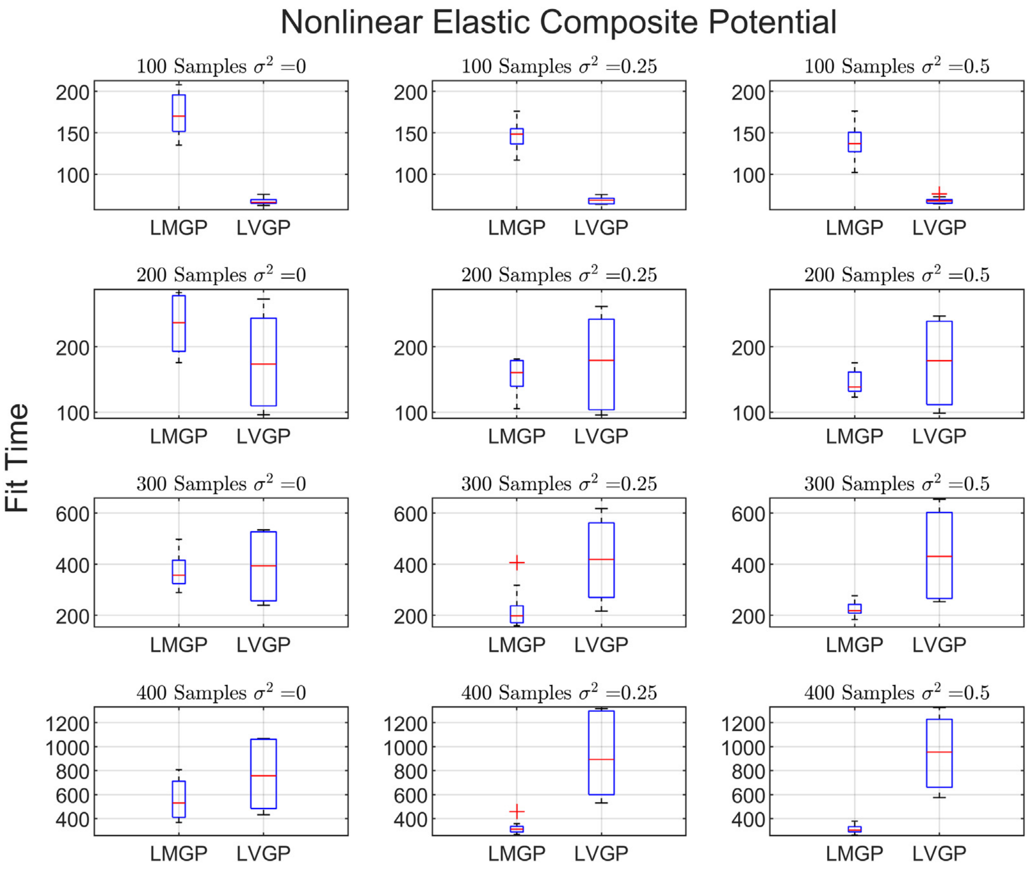


Fig. 13. Training costs for the borehole function: The rows and columns are organized based on training dataset size and noise variance, respectively.

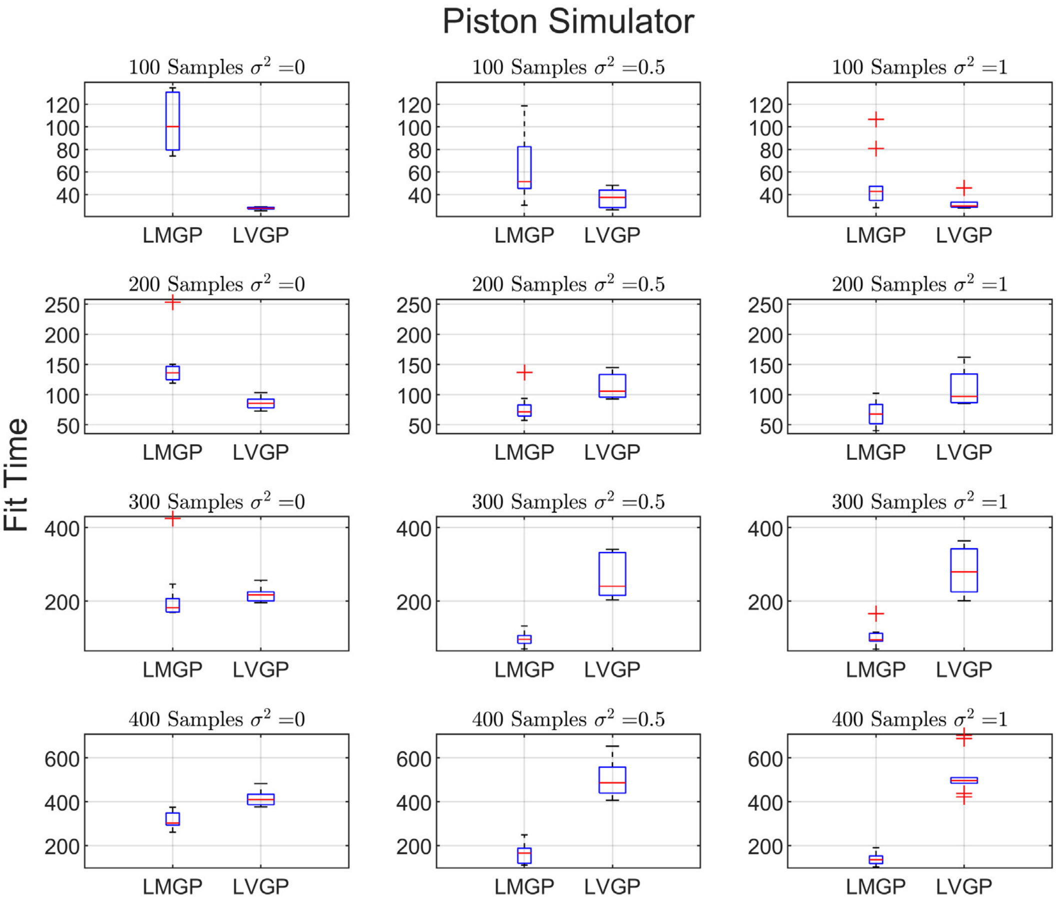


Fig. 14. Training costs for the Piston simulator function: The rows and columns are organized based on training dataset size and noise variance, respectively.

References

[1] C.E. Rasmussen, Gaussian Processes for Machine Learning, 2006.

[2] S. Tao, K. Shintani, R. Bostanabad, Y.-C. Chan, G. Yang, H. Meingast, W. Chen, Enhanced gaussian process metamodeling and collaborative optimization for vehicle suspension design optimization, in: ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, 2017.

[3] R. Bostanabad, Y.-C. Chan, L. Wang, P. Zhu, W. Chen, Globally approximate gaussian processes for big data with application to data-driven metamaterials design, J. Mech. Des. 141 (11) (2019).

[4] D.G. Giovanis, M.D. Shields, Data-driven surrogates for high dimensional models using gaussian process regression on the grassmann manifold, Comput. Methods Appl. Mech. Engrg. 370 (2020) 113269.

[5] M. Plumlee, D.W. Apley, Lifted brownian kriging models, Technometrics 59 (2) (2017) 165–177.

[6] R.B. Gramacy, H.K.H. Lee, Bayesian treed gaussian process models with an application to computer modeling, J. Amer. Statist. Assoc. 103 (483) (2012) 1119–1130.

[7] R.B. Gramacy, D.W. Apley, Local gaussian process approximation for large computer experiments, J. Comput. Graph. Statist. 24 (2) (2015) 561–578.

[8] X. Du, H. Xu, F. Zhu, A data mining method for structure design with uncertainty in design variables, Comput. Struct. 244 (2021) 106457.

[9] P. Perdikaris, M. Raissi, A. Damianou, N.D. Lawrence, G.E. Karniadakis, Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling, Proc. Math. Phys. Eng. Sci. 473 (2198) (2017) 20160751.

[10] M. Raissi, Parametric gaussian process regression for big data, 2017, arXiv preprint arXiv:1704.03144.

[11] M. Raissi, P. Perdikaris, G.E. Karniadakis, Machine learning of linear differential equations using gaussian processes, J. Comput. Phys. 348 (2017) 683–693.

[12] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[13] C.M. Bishop, Neural Networks for Pattern Recognition., Oxford University Press, 1995.

[14] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, M.A. Bessa, Deep learning predicts path-dependent plasticity, Proc. Natl. Acad. Sci. USA 116 (52) (2019) 26414–26420.

[15] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Comput. Methods Appl. Mech. Engrg. 379 (2021) 113741.

[16] E. Haghighat, R. Juanes, Sciann: A keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113552.

[17] S. Saha, et al., Hierarchical deep learning neural network (hidenn): An artificial intelligence (ai) framework for computational science and engineering, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113452.

[18] Y. Suh, R. Bostanabad, Y. Won, Deep learning predicts boiling heat transfer, Sci. Rep. 11 (1) (2021) 5622.

[19] H. Wang, A. Planas, R. Chandramowlishwaran, R. Bostanabad, Train once and use forever: Solving boundary value problems in unseen domains with pre-trained deep learning models, 2021, arXiv preprint arXiv:2104.10873.

[20] T. Chen, C. Guestrin, Xgboost: A Scalable Tree Boosting System. in Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, ACM, 2016.

[21] E. Alpaydin, Introduction To Machine Learning, MIT Press, 2014.

[22] T. Therneau, B. Atkinson, B. Ripley, Rpart: Recursive Partitioning and Regression Trees, 2014.

[23] R. Bostanabad, et al., Uncertainty quantification in multiscale simulation of woven fiber composites, Comput. Methods Appl. Mech. Engrg. 338 (2018) 506–532.

[24] W. Zhang, R. Bostanabad, B. Liang, X. Su, D. Zeng, M.A. Bessa, Y. Wang, W. Chen, J. Cao, A numerical bayesian-calibrated characterization method for multiscale prepreg preforming simulations with tension-shear coupling, Compos. Sci. Technol. 170 (2019) 15–24.

[25] M. Kennedy, Predicting the output from a complex computer code when fast approximations are available, Biometrika 87 (1) (2000) 1–13.

[26] S. Conti, J.E. Gosling, A. O'Hagan, Gaussian process emulation of dynamic computer codes, Biometrika 96 (3) (2009) 663–676.

[27] M. Plumlee, Bayesian calibration of inexact computer models, J. Amer. Statist. Assoc. (2016) in press.

[28] M. Plumlee, V.R. Joseph, H. Yang, Calibrating functional parameters in the ion channel models of cardiac cells, J. Amer. Statist. Assoc. 111 (514) (2016) 500–509.

[29] H. Lam, X. Zhang, M. Plumlee, Improving prediction from stochastic simulation via model discrepancy learning, in: 2017 Winter Simulation Conference (WSC), 2017.

[30] Y. Wang, A. Iyer, W. Chen, J.M. Rondinelli, Featureless adaptive optimization accelerates functional electronic materials design, Appl. Phys. Rev. 7 (4) (2020) 041403.

[31] C. He, J. Ge, B. Zhang, J. Gao, S. Zhong, W.K. Liu, D. Fang, A hierarchical multiscale model for the elastic–plastic damage behavior of 3d braided composites at high temperature, Compos. Sci. Technol. 196 (2020) 108230.

[32] Y. Zhang, S. Tao, W. Chen, D.W. Apley, A latent variable approach to gaussian process modeling with qualitative and quantitative factors, Technometrics 62 (3) (2019) 291–302.

[33] R.R. Schmidt, E.E. Cruz, M. Iyengar, Challenges of data center thermal management, IBM J. Res. Dev. 49 (4.5) (2005) 709–723.

[34] S. Conti, A. O'Hagan, Bayesian emulation of complex multi-output and dynamic computer models, J. Statist. Plann. Inference 140 (3) (2010) 640–651.

[35] R. Bostanabad, T. Kearney, S. Tao, D.W. Apley, W. Chen, Leveraging the nugget parameter for efficient gaussian process modeling, Internat. J. Numer. Methods Engrg. 114 (5) (2018) 501–516.

[36] R. Jin, W. Chen, T.W. Simpson, Comparative studies of metamodelling techniques under multiple modelling criteria, Struct. Multidiscip. Optim. 23 (1) (2001) 1–13.

[37] H. Xu, Constructing oscillating function-based covariance matrix to allow negative correlations in gaussian random field models for uncertainty quantification, J. Mech. Des. 142 (7) (2020).

[38] R.G. Gallager, Stochastic processes: Theory for Applications, Cambridge University Press, 2013.

[39] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[40] D.J.J. Toal, N.W. Bressloff, A.J. Keane, C.M.E. Holden, The development of a hybridized particle swarm for kriging hyperparameter tuning, Eng. Optim. 43 (6) (2011) 675–699.

[41] C. Zhu, R.H. Byrd, P. Lu, J. Nocedal, Algorithm 778: L-bfgs-b, ACM Trans. Math. Software 23 (4) (1997) 550–560.

[42] R.B. Gramacy, H.K.H. Lee, Cases for the nugget in modeling computer experiments, Stat. Comput. 22 (3) (2010) 713–722.

[43] P.D. Arendt, D.W. Apley, W. Chen, D. Lamb, D. Gorsich, Improving identifiability in model calibration using multiple responses, J. Mech. Des. 134 (10) (2012) 100909.

[44] P.D. Arendt, D.W. Apley, W. Chen, Quantification of model uncertainty: Calibration, model discrepancy, and identifiability, J. Mech. Des. 134 (10) (2012) 100908.

[45] H. Xu, C.-H. Chuang, R.-J. Yang, Mixed-variable metamodeling methods for designing multi-material structures, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, 2016.

[46] P.Z.G. Qian, H. Wu, C.F.J. Wu, Gaussian process models for computer experiments with qualitative and quantitative factors, Technometrics 50 (3) (2008) 383–396.

[47] X. Deng, C.D. Lin, K.W. Liu, R.K. Rowe, Additive gaussian process for computer models with qualitative and quantitative factors, Technometrics 59 (3) (2017) 283–292.

[48] Y. Zhang, W.I. Notz, Computer experiments with qualitative and quantitative variables: A review and reexamination, Qual. Eng. 27 (1) (2014) 2–13.

[49] L. Wang, S. Tao, P. Zhu, W. Chen, Data-driven topology optimization with multiclass microstructures using latent variable gaussian process, J. Mech. Des. 143 (3) (2021) 031708.

[50] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders.

[51] C. Soize, R. Ghanem, Probabilistic learning on manifolds constrained by nonlinear partial differential equations for small datasets, Comput. Methods Appl. Mech. Engrg. 380 (2021) 113777.

[52] M.D. Morris, T.J. Mitchell, D. Ylvisaker, Bayesian design and analysis of computer experiments: Use of derivatives in surface prediction, Technometrics 35 (3) (1993) 243–255.

[53] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, S. Tarantola, Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index, Comput. Phys. Comm. 181 (2) (2010) 259–270.

[54] P.G. Constantine, E. Dow, Q. Wang, Active subspace methods in theory and practice: Applications to kriging surfaces, SIAM J. Sci. Comput. 36 (4) (2014) A1500–A1524.

[55] N. Wycoff, M. Binois, S.M. Wild, Sequential learning of active subspaces, 2019, arXiv preprint arXiv:1907.11572.

[56] R.D. Cook, L. Ni, Sufficient dimension reduction via inverse regression: A minimum discrepancy approach, J. Amer. Statist. Assoc. 100 (470) (2005) 410–428.

[57] K.-C. Li, Sliced inverse regression for dimension reduction, J. Amer. Statist. Assoc. 86 (414) (1991) 316–327.

[58] F. Chiaromonte, R.D. Cook, B. Li, Sufficient dimensions reduction in regressions with categorical predictors, Ann. Statist. 30 (2) (2002) 475–497.

[59] E.N. Ben-Ari, D.M. Steinberg, Modeling data from computer experiments: An empirical comparison of kriging with mars and projection pursuit regression, Qual. Eng. 19 (4) (2007) 327–338.

[60] B.A. Le, J. Yvonnet, Q.C. He, Computational homogenization of nonlinear elastic materials using neural networks, Internat. J. Numer. Methods Engrg. 104 (12) (2015) 1061–1084.

[61] H. Moon, Design and Analysis of Computer Experiments for Screening Input Variables, The Ohio State University, 2010.

[62] H. Dette, A. Pepelyshev, Generalized latin hypercube design for computer experiments, Technometrics 52 (4) (2010) 421–429.

[63] Sobol' I.y.M., On the distribution of points in a cube and the approximate evaluation of integrals, Zh. Vychisl. Mat. Mat. Fiz. 7 (4) (1967) 784–802.

[64] Sobol' I.y.M., On sensitivity estimation for nonlinear mathematical models, Mat. Model. 2 (1) (1990) 112–118.

[65] I.M. Sobol, On quasi-monte carlo integrations, Math. Comput. Simulation 47 (2–5) (1998) 103–112.

[66] D. Harrison, D.L. Rubinfeld, Hedonic housing prices and the demand for clean air, J. Environ. Econ. Manag. 5 (1) (1978) 81–102.

[67] D. Dua, C. Graff, Uci Machine Learning Repository, School of Information and Computer Sciences, University of California, Irvine, 2019;, Available from.

[68] T. Pearce, M. Zaki, A. Brintrup, A. Neel, Uncertainty in neural networks: Bayesian ensembling, 2018, arXiv preprint arXiv:1810.05546.

[69] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: international conference on machine learning, 2016.

[70] A.A. Bataineh, D. Kaur, A comparative study of different curve fitting algorithms in artificial neural network using housing dataset. IEEE.

[71] C.-U. Yeom, K.-C. Kwak, Performance evaluation of automobile fuel consumption using a fuzzy-based granular model with coverage and specificity, Symmetry 11 (12) (2019) 1480.

[72] P.V. Balachandran, D. Xue, J. Theiler, J. Hogden, T. Lookman, Adaptive strategies for materials design using uncertainties, Sci. Rep. 6 (2016) 19660.

[73] M.F. Cover, O. Warschkow, M.M. Bilek, D.R. McKenzie, A comprehensive survey of m(2)ax phase elastic properties, J. Phys.: Condens. Matter 21 (30) (2009) 305403.

[74] Y.C. Zhou, H.Y. Dong, X.H. Wang, S.Q. Chen, Electronic structure of the layered ternary carbides ti2snc and ti2gec, J. Phys.: Condens. Matter 12 (46) (2000) 9617.

[75] T. El-Raghy, M.W. Barsoum, A. Zavaliangos, S.R. Kalidindi, Processing and mechanical properties of ti3sic2: Ii, effect of grain size and deformation temperature, J. Am. Ceram. Soc. 82 (10) (1999) 2855–2860.

[76] Y. Zhang, D.W. Apley, W. Chen, Bayesian optimization for materials design with mixed quantitative and qualitative variables, Sci. Rep. 10 (1) (2020) 4924.

[77] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A.G. Wilson, E. Bakshy, Botorch: A framework for efficient monte-carlo bayesian optimization, Adv. Neural Inf. Process. Syst. 33 (2020).

[78] J.T. Wilson, R. Moriconi, F. Hutter, M.P. Deisenroth, The reparameterization trick for acquisition functions, 2017, arXiv preprint arXiv:1712.00424.

[79] J. Wu, S. Toscano-Palmerin, P.I. Frazier, A.G. Wilson, Practical multi-fidelity bayesian optimization for hyperparameter tuning. PMLR.