Ramin Bostanabad
Department of Mechanical Engineering,
Northwestern University,
2145 Sheridan Road,
Evanston, IL 60208
e-mail: bostanabad@u.northwestern.edu

Yu-Chin Chan
Department of Mechanical Engineering,
Northwestern University,
2145 Sheridan Road,
Evanston, IL 60208
e-mail: ychan@u.northwestern.edu

Liwei Wang
State Key Laboratory of Mechanical
System and Vibration,
School of Mechanical Engineering,
Shanghai Jiao Tong University,
800 Dongchuan Road,
Shanghai 200240, China
e-mail: iridescence@sjtu.edu.cn

Ping Zhu
State Key Laboratory of Mechanical
System and Vibration,
School of Mechanical Engineering,
Shanghai Jiao Tong University,
800 Dongchuan Road,
Shanghai 200240, China
e-mail: pzhu@sjtu.edu.cn

Wei Chen[1]
Department of Mechanical Engineering,
Northwestern University,
2145 Sheridan Road,
Evanston, IL 60208
e-mail: weichen@northwestern.edu

Globally Approximate Gaussian Processes for Big Data With Application to Data-Driven Metamaterials Design

We introduce a novel method for Gaussian process (GP) modeling of massive datasets called globally approximate Gaussian process (GAGP). Unlike most large-scale supervised learners such as neural networks and trees, GAGP is easy to fit and can interpret the model behavior, making it particularly useful in engineering design with big data. The key idea of GAGP is to build a collection of independent GPs that use the same hyperparameters but randomly distribute the entire training dataset among themselves. This is based on our observation that the GP hyperparameter approximations change negligibly as the size of the training data exceeds a certain level, which can be estimated systematically. For inference, the predictions from all GPs in the collection are pooled, allowing the entire training dataset to be efficiently exploited for prediction. Through analytical examples, we demonstrate that GAGP achieves very high predictive power matching (and in some cases exceeding) that of state-of-the-art supervised learning methods. We illustrate the application of GAGP in engineering design with a problem on data-driven metamaterials, using it to link reduced-dimension geometrical descriptors of unit cells and their properties. Searching for new unit cell designs with desired properties is then achieved by employing GAGP in inverse optimization. [DOI: 10.1115/1.4044257]

Keywords: Gaussian processes, supervised learning, big data, metamaterials, design automation, design optimization, design representation, metamodeling

1 Introduction

Fueled by recent advancements in high-performance computing as well as data acquisition and storage capabilities (e.g., online repositories), data-driven methods are increasingly employed in engineering design [1–3] to efficiently explore the design space of complex systems by obviating the need for expensive experiments or simulations. For emerging material systems, in particular, large datasets have been successfully leveraged to design heterogeneous materials [4–8] and mechanical metamaterials [9–12].

Key to data-driven design is to develop supervised learners that can distill as much useful information from massive datasets as possible. However, most large-scale learners such as deep neural networks (NNs) [13] and gradient boosted trees (GBT) [14] are difficult to interpret and hence less suitable for engineering design. Gaussian process (GP) models (also known as Kriging) have many attractive features that underpin their widespread use in engineering design. For example, GPs interpolate the data, have a natural and intuitive mechanism to smooth the data to address noise (i.e., to avoid interpolation) [15], and are very interpretable (i.e., provide insight into input–output relations) [16,17]. In addition, they quantify prediction uncertainty and have analytical conditional distributions that enable, e.g., tractable adaptive sampling or Bayesian analysis [18]. However, conventional GPs are not readily applicable to large datasets and have been mostly confined to engineering design with small data. The goal of our work is to bridge the gap between big data and GPs while achieving high predictive accuracy.

The difficulty in fitting GPs to big data is rooted in the repetitive inversion of the sample correlation matrix, $\boldsymbol{R}$, whose size equals the number of training samples, n. Given the practical features and popularity of GPs, considerable effort has been devoted to resolving their scalability shortcoming. One avenue of research has explored partitioning the input space (and hence the training data) via, e.g., trees [19] or Voronoi cells [20], and fitting an independent GP to each partition. While particularly useful for small to relatively large datasets that exhibit the nonstationary behavior, prediction using these methods results in discontinuity (at the partitions' boundaries) and information loss (because the query point is associated with only one partition). Projected process approximation (PPA) [21] is another method where the information from n samples is distilled into $m \ll n$ randomly (or sequentially) selected samples through conditional distributions. PPA is very sensitive to the m selected samples, however, and overestimates the variance [21]. In Bayesian committee machine (BCM) [22], the dataset is partitioned into p mutually exclusive and collectively exhaustive parts with independent GP priors, and then, the predictions from all the GPs are pooled together in a Bayesian setting. While theoretically very attractive, BCM does not scale well with the dataset size and is computationally very expensive.

Another avenue of research has pursued subset selection. For example, a simple strategy is to only use $m \ll n$ samples to train a

[1]Corresponding author.

Contributed by the Design Automation Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received March 1, 2019; final manuscript received July 11, 2019; published online July 17, 2019. Assoc. Editor: Samy Missoum.

GP [23,24], where the m samples are selected either randomly or sequentially based on maximizing some criteria such as information gain or differential entropy score. Reduced-rank approximation of $\boldsymbol{R}$ with $m \ll n$ samples is another option for subset selection and has been used in the Nystrom [25] and subset of regressors [26,27] methods. The m samples in these methods are chosen randomly or in a greedy fashion to minimize some cost function. While the many variants of subset selection may be useful in some applications, they waste information and are not applicable to very large datasets due to the computational and storage costs. Local methods also use subsets of the data because they fit a stationary GP (for each prediction) to a very small number of training data points that are closest to the query point. Locally approximate Gaussian process (LAGP) [28] is perhaps the most widely recognized local method where the subsets are selected either based on their proximity to the query point or to minimize the predictive variance. Despite being useful for nonstationary and relatively large datasets, local methods also waste some information and can be prohibitively expensive for repetitive use since local samples have to be found and a GP must be fitted for each prediction.

Although the recent works have made significant progress in bridging the gap between GPs and big data, GPs still struggle to achieve the accuracy of the state-of-the-art large-scale supervised learners such as NNs and trees. Motivated by this limitation, we develop a computationally stable and inexpensive approach for GP modeling of massive datasets. The main idea of our approach is to build a collection of independent GPs that utilize a converged roughness parameter as their hyperparameters. This is based on an empirical observation that the estimates of the GP hyperparameters change negligibly as the size of the training data exceeds a certain level. While having some common aspects with a few of the abovementioned works, our method is more massively scalable, can leverage multicore or graphical processing unit computations [29,30], and is applicable to very high-dimensional data with or without noise.

As mentioned earlier, big data have enticed new design methods for complex systems such as metamaterials [9–12], which possess superior properties through their hierarchical structure that consists of repeated unit cells. While traditional methods like topology optimization (TO) provide a systematic computational platform to find metamaterials with unprecedented properties, they have many challenges that are primarily due to the high-dimensional design space (i.e., the geometry of unit cells), computational costs, local optimality, and spatial discontinuities across unit cell boundaries (when multiple unit cells are simultaneously designed). Techniques for TO such as varying the volume fraction or size of one unit cell to maintain continuous boundaries [31,32], adding connectivity constraints [33], and substructuring [34] have recently been proposed but cannot fully address all of the above challenges. Instead, we take a data-driven approach by first building a large training database of many unit cells and their corresponding properties. Unlike previous data-driven works that represent unit cells as signed distance fields [9] or voxels [11], we drastically reduce the input dimension in our dataset by characterizing the unit cells via spectral shape descriptors based on the Laplace–Beltrami (LB) operator. Then, we employ our globally approximate Gaussian process (GAGP) modeling approach to link the LB descriptors of unit cells to their properties and, in turn, efficiently discover new unit cells with desired properties.

The rest of the paper is organized as follows. We first review some preliminaries on GP modeling in Sec. 2 and then introduce our novel idea in Sec. 3. In Sec. 4, we validate the accuracy of our approach by comparing its performance against three popular and large-scale supervised learning methods on five analytical problems. We demonstrate an application of GAGP to our data-driven design method for metamaterials in Sec. 5 and conclude the paper in Sec. 6.

2 Review on Gaussian Process Modeling

Below, we describe how GP emulators (also known as surrogates, metamodels, or models) can replace a computer simulator. The procedure is identical if the data are obtained from physical experiments. Let us denote the output and inputs of a computer simulator by, respectively, y and the d dimensional vector $\boldsymbol{x} = [x_{(1)}, x_{(2)}, \ldots, x_{(d)}]^T$, where $\boldsymbol{x} \in \mathbb{R}^d$. Assume the input–output relation is a realization of the random process $\eta(\boldsymbol{x})$:

$$\eta(\boldsymbol{x}) = \sum_{i=1}^{h} \beta_{(i)} f_i(\boldsymbol{x}) + \xi(\boldsymbol{x}) \tag{1}$$

where $f_i(\boldsymbol{x})$'s are some predetermined set of basis functions, $\boldsymbol{\beta} = [\beta_{(1)}, \ldots., \beta_{(h)}]^T$ are unknown weights, and $\xi(\boldsymbol{x})$ is a zero-mean GP characterized with its parametric covariance function, $c(\cdot, \cdot)$:

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

where $r(\cdot)$ is the correlation function having the property $r(\boldsymbol{x}, \boldsymbol{x}) = 1$ and σ^2 is the process variance. Various correlation functions have been developed in the literature, with the most widely used one being the Gaussian correlation function:

$$r(\boldsymbol{x}, \boldsymbol{x}') = \exp\{-(\boldsymbol{x} - \boldsymbol{x}')^T \boldsymbol{\Omega} (\boldsymbol{x} - \boldsymbol{x}')\} \tag{3}$$

where $\boldsymbol{\Omega} = diag(10^{\boldsymbol{\omega}})$ and $\boldsymbol{\omega} = [\omega_{(1)}, \omega_{(2)}, \ldots, \omega_{(d)}]^T$, $-\infty < \omega_i < \infty$ are the roughness or scale parameters. The collection of σ^2 and $\boldsymbol{\omega}$ are called the hyperparameters.

With the formulation in Eq. (1) and given the n training pairs of $(\boldsymbol{x}_i, y_i)$, GP modeling requires finding a point estimate for $\boldsymbol{\beta}$, $\boldsymbol{\omega}$, and σ^2 via either maximum likelihood estimation (MLE) or cross-validation (CV). Alternatively, Bayes' rule can be employed to find the posterior distributions if there is prior knowledge on these parameters. Herein, we use a constant process mean (i.e., $\sum_{i=1}^{h} \beta_i f_i(\boldsymbol{x}) = \beta$) and employ MLE. These choices are widely practiced because a high predictive power is provided while computational costs are minimized [28,35–39].

MLE requires maximizing the multivariate Gaussian likelihood function, or equivalently:

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

where $\log(\cdot)$ is the natural logarithm, $\boldsymbol{1}$ is an $n \times 1$ vector of ones, and $\boldsymbol{R}$ is the $n \times n$ correlation matrix with (i, j)th element $R_{ij} = r(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $i, j = 1, \ldots, n$. Setting the partial derivatives with respect to β and σ^2 to zero yields:

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

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

Plugging these values into Eq. (4) and eliminating the constants:

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

By numerically minimizing L in Eq. (7), one can find $\hat{\boldsymbol{\omega}}$. Many global optimization methods such as genetic algorithm (GA) [40], pattern searches [41,42], and particle swarm optimization [43] have been employed to solve for $\hat{\boldsymbol{\omega}}$ in Eq. (7). However, gradient-based optimization techniques are commonly preferred due to

their ease of implementation and superior computational efficiency [15,16,35]. To guarantee global optimality in this case, the optimization is done numerous times with different initial guesses. It is noted that, in practice, the search space of ω_i is generally limited to $[-20, 5]$ rather than $(-\infty, \infty)$ since the correlation exponentially changes as a function of ω_i (see also Fig. 3).

On completion of MLE, the following closed-form formula can be used to predict the response at any $\boldsymbol{x}^*$:

$$\hat{y}(\boldsymbol{x}^*) = \hat{\beta} + \boldsymbol{g}^T(\boldsymbol{x}^*)\boldsymbol{V}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\beta}) \tag{8}$$

where $\boldsymbol{g}(\boldsymbol{x}^*)$ is an $n \times 1$ vector with ith element $c(\boldsymbol{x}_i, \boldsymbol{x}^*) = \hat{\sigma}^2 r(\boldsymbol{x}_i, \boldsymbol{x}^*)$, $\boldsymbol{V}$ is the covariance matrix with (i, j)th element $\hat{\sigma}^2 r(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and $\boldsymbol{y} = [y_1, \ldots, y_n]^T$ are the responses in the training dataset. The posterior covariance between the responses at $\boldsymbol{x}^*$ and $\boldsymbol{x}'$ reads:

$$\text{cov}(y^*, y') = c(\boldsymbol{x}^*, \boldsymbol{x}') - \boldsymbol{g}^T(\boldsymbol{x}^*)\boldsymbol{V}^{-1}\boldsymbol{g}(\boldsymbol{x}') + \boldsymbol{h}^T(\boldsymbol{1}^T\boldsymbol{V}^{-1}\boldsymbol{1})^{-1}\boldsymbol{h} \tag{9}$$

where $\boldsymbol{h} = (\boldsymbol{1} - \boldsymbol{1}^T\boldsymbol{V}^{-1}\boldsymbol{g}(\boldsymbol{x}'))$.

If the training dataset has multiple outputs, one may fit either a single-response GP emulator to each response or a multiresponse GP to all the responses. We follow Ref. [44] and extend the above formulations to simulators with q responses by placing a constant mean for each response (i.e., $\boldsymbol{\beta} = [\beta_{(1)}, \ldots, \beta_{(q)}]^T$) and employing the separable covariance function:

$$\text{cov}(\xi(\boldsymbol{x}), \xi(\boldsymbol{x}')) = c(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\Sigma} \otimes r(\boldsymbol{x}, \boldsymbol{x}') \tag{10}$$

where $\otimes$ denotes the Kronecker product and $\boldsymbol{\Sigma}$ is the $q \times q$ process covariance matrix with its off-diagonal elements representing the covariance between the corresponding responses at any fixed $\boldsymbol{x}$. The MLE approach described above can also be applied to multiresponse datasets in which case σ^2 will be replaced with $\boldsymbol{\Sigma}$ (see Refs. [45–48] for details).

Finally, we note that GPs can address noise and smooth the data (i.e., avoid interpolation) via the so-called nugget or jitter parameter, δ, in which case $\boldsymbol{R}$ is replaced with $\boldsymbol{R}_\delta = \boldsymbol{R} + \delta \boldsymbol{I}_{n \times n}$. If δ is used, the estimated (stationary) noise variance in the data would be $\delta\hat{\sigma}^2$. We have recently developed an automatic method to robustly detect and estimate noise [35].

3 Globally Approximate Gaussian Process

Regardless of the optimization method used to solve for $\hat{\boldsymbol{\omega}}$, each evaluation of L in Eq. (7) requires inverting the $n \times n$ matrix $\boldsymbol{R}$. For very large n, there are two main challenges associated with this inversion: computational cost of approximately $O(\alpha n^3)$ and singularity of $\boldsymbol{R}$ (since the samples get closer as n increases). To address these issues and enable GP modeling of big data, our essential idea is to build a collection of independent GPs that use the same $\hat{\boldsymbol{\omega}}$ and share the training data among themselves.

To illustrate, we consider the function $y = x^4 - x^3 - 7x^2 + 3x + 5\sin(5x)$ over $-2 \le x \le 3$. The associated likelihood profile (i.e., L) is visualized in Fig. 1 as a function of ω for various values of n. Two interesting phenomena are observed in this figure: (i) With large n, the profile of L does not alter as the training samples change. To observe this, for each n, we generate five independent training samples via Sobol sequence [49,50] and plot the corresponding L. As illustrated in Fig. 1, even though a total of 20 curves are plotted, only four are visible since the five curves with the same n are indistinguishable. (ii) As n increases, L is minimized at similar ω's.

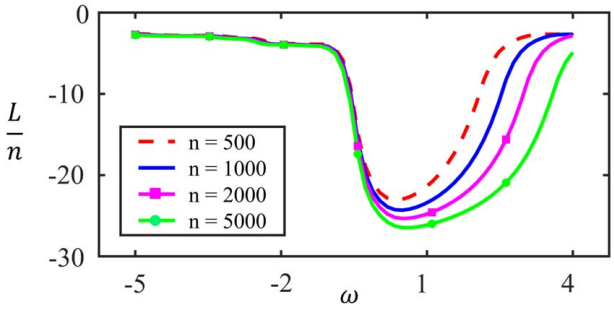

Fig. 1 The profile of $\frac{L}{n}$ as a function of ω for various values of n. For each n, five curves are plotted, but only four are visible since the curves with the same n are indistinguishable.

While we visualize the above two points with a simple 1D function, our studies indicate that they hold in general (i.e., irrespective of problem dimensionality and the absence or presence of noise; see Sec. 4) as long as the number of training samples is large. Therefore, we propose the following approach for GP modeling of large datasets.

Assuming a very large training dataset of size n is available, we first randomly select a relatively small subset of size n_0 (e.g., $n_0 = 500$) and estimate $\hat{\boldsymbol{\omega}}_0$ with a gradient-based optimization technique. Then, we add n_s random samples (e.g., $n_s = 250$) to this subset and estimate $\hat{\boldsymbol{\omega}}_1$ while employing $\hat{\boldsymbol{\omega}}_0$ as the initial guess in the optimization. This process is stopped after s steps when $\hat{\boldsymbol{\omega}}$ does not change noticeably (i.e., $\hat{\boldsymbol{\omega}}_s \cong \hat{\boldsymbol{\omega}}_{s-1}$) as more training data are used. The latest solution, denoted by $\hat{\boldsymbol{\omega}}_\infty$, is then employed to build m GP models, each with $n_k \ge n_0 + s \times n_s$ samples chosen randomly from the entire training data such that $n = \sum_{k=1}^{m} n_k$. Here, we have assumed that the collection of these GPs (who have $\hat{\boldsymbol{\omega}}_\infty$ as their hyperparameters) approximate a GP that is fitted to the entire training dataset and, correspondingly, call it GAGP. The algorithm of GAGP is presented in Fig. 2.

We point out the following important features regarding GAGP. First, we recommend using gradient-based optimizations throughout the entire process because (i) if n_0 is large enough (e.g., $n_0 > 500$), one would need to select only a few initial guesses to find the global minimizer of Eq. (7), i.e., $\hat{\boldsymbol{\omega}}_0$ (we suggest the method developed in Ref. [35] to estimate $\hat{\boldsymbol{\omega}}_0$); and (ii) we want to use $\hat{\boldsymbol{\omega}}_{i-1}$ as the initial guess for the optimization in the ith step to ensure fast convergence since the minimizer of L changes slightly as the dataset size increases (see Fig. 1). Regarding the choice on n_0, note that it has to be small enough to avoid prohibitive computational time but large enough, so that (i) the global optimum changes slightly if $n_0 + s$ data points are used instead of n_0 data points, and (ii) most (if not all) of the local optima of L are smoothed out. Second, for predicting the response, Eq. (8) is used for each of the m GP models, and then, the results are averaged. In our experience, we observe very similar prediction results with different averaging (e.g., weighted averaging where the weights are proportional to inverse variance) or pooling (e.g., median) schemes. The best scheme for a particular problem can be found via CV, but we avoid this step to ensure ease of use and generality. The advantages of employing a collection of models (in our case the m GPs) in prediction are extensively demonstrated in the literature [14,22]. Third, the predictive power is not sensitive to n_0, s, and n_s so long as large

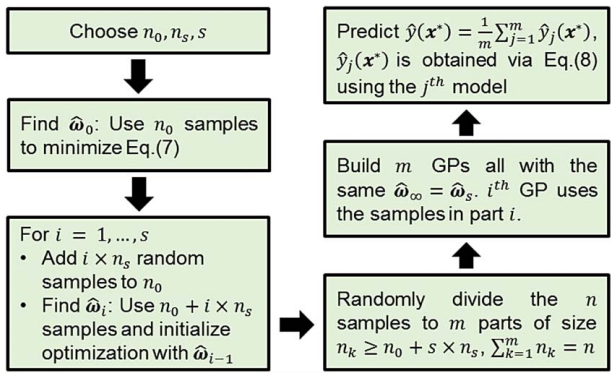

Fig. 2 Flowchart of globally approximate Gaussian process. It is assumed that a very large training dataset of size n is available.

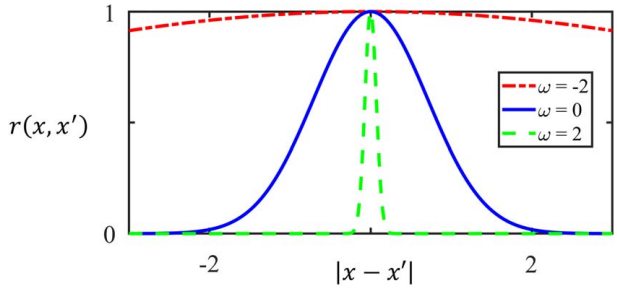

Fig. 3 Effect of ω on the correlation between x and x′ in 1D. See Eq. (3) for the functional form of $r(x, x')$.

enough values are used for them. For novice users, we recommend starting with $n_0 = 500$, $s = 6$, and $n_s = 250$, and equally distributing the samples among the m resulting GPs (we use these parameters in Sec. 5 and for all the examples in Sec. 4). For more experienced users, we provide a systematic way in Sec. 4 to choose these values based on GP's inherent ability to estimate noise using the nugget variance. Finally, we point out that GAGP has a high predictive power and is applicable to very large datasets while maintaining a straightforward implementation because it only entails integrating a GP modeling package such as GPM [35] with the algorithm presented in Fig. 2.

4 Comparative Studies on Analytical Examples

To validate the performance of GAGP in regression, we compare its predictive power on five examples (Ex1–5) against recognized big data learners: LAGP [28], GBT [14], and feed-forward NNs [51]. As shown in Eqs. (11)–(15), the examples cover a wide range of dimensionality and input–output complexity. Ex1 is a 1D function with many local fluctuations due to the $\sin(5x)$ term. Ex2 is a 4D function that is primarily sensitive to the first two inputs. Ex3 is a 7D function that models the cycle time of a piston, which mainly depends on the first and fourth inputs. Ex4 is a very complex 10D function relating the stored potential in a material to the applied load and material properties; the inputs interact nonlinearly and all affect the response. Finally, Ex5 is a 20D function where the output is independent of x_8 and x_{16}.

- Ex1:

$$y = x^4 - x^3 - 7x^2 + 3x + 5\sin(5x)$$
$$-2 \le x \le 3 \tag{11}$$

- Ex2 [52]:

$$y = \frac{\left(1 - \exp\left(-\frac{1}{2x_2}\right)\right)(x_3 x_1^3 + 1900x_1^2 + 2092x_1 + 60)}{x_4 x_1^3 + 500x_1^2 + 4x_1 + 20}$$
$$\min(\boldsymbol{x}) = [0,\ 0,\ 2200,\ 85]$$
$$\max(\boldsymbol{x}) = [1,\ 1,\ 2400,\ 110] \tag{12}$$

- Ex3 [53]:

$$y = 2\pi\sqrt{\frac{x_1}{x_4 + \frac{4x_3x_4x_5x_6}{\left(x_5x_2 + 19.62x_1 - \frac{x_4x_3}{x_2}\right)^2 + 4x_4x_5\left(\frac{x_6}{x_7}\right)x_7}}}$$
$$\min(\boldsymbol{x}) = [30,\ 0.005,\ 0.002,\ 1000,\ 9\times10^4,\ 290,\ 340]$$
$$\max(\boldsymbol{x}) = [60,\ 0.02,\ 0.01,\ 5000,\ 11\times10^4,\ 296,\ 360] \tag{13}$$

- Ex4 [54]:

$$y = \frac{9}{2}x_9\varepsilon_m^2 + \frac{x_8x_{10}}{1+x_7}\left[\frac{\varepsilon_{eq}}{x_{10}}\right]^{1+x_7}$$
$$\boldsymbol{\varepsilon} = \begin{pmatrix} x_1 & x_6 & x_5 \\ x_6 & x_2 & x_4 \\ x_5 & x_4 & x_3 \end{pmatrix}, \varepsilon_m = \frac{1}{3}Tr(\boldsymbol{\varepsilon}),$$
$$\varepsilon_d = \boldsymbol{\varepsilon} - \varepsilon_m\mathbf{1}, \varepsilon_{eq} = \sqrt{\frac{2}{3}(\varepsilon_d{:}\varepsilon_d)}$$
$$\min(\boldsymbol{x}) = [-0.1,\ -0.1,\ -0.1,\ -0.1,\ -0.1,\ -0.1,\ 0.02,\ 1,\ 5,\ 0.1]$$
$$\max(\boldsymbol{x}) = [0.1,\ 0.1,\ 0.1,\ 0.1,\ 0.1,\ 0.1,\ 0.1,\ 5,\ 25,\ 1.5] \tag{14}$$

- Ex5 [53]:

$$y = \frac{5x_{12}}{1+x_1} + 5(x_4 - x_{20})^2 + x_5 + 40x_{19}^3 - 5x_1 + 0.05x_2 + 0.08x_3$$
$$- 0.03x_6 + 0.03x_7 - 0.09x_9 - 0.01x_{10} - 0.07x_{11} + 0.25x_{13}^2$$
$$- 0.04x_{14} + 0.06x_{15} - 0.01x_{17} - 0.03x_{18}$$
$$-0.5 \le x_i \le 0.5 \quad \text{for } i = 1,\ \ldots 20 \tag{15}$$

For each example, two independent and unique datasets of size 30,000 are generated with Sobol sequence [50], where the first one is used for training and the second for validation. In each example, Gaussian noise is added to both the training and validation outputs. We consider two noise levels to test the sensitivity of the results where the noise standard deviation (SD) is determined based on each example's output range (e.g., the outputs in Ex1 and Ex4 fall in the [−20, 5] and [0, 1.8] ranges, respectively). As we measure performance by root mean squared error (RMSE), the noise SD should be recovered on the validation dataset (i.e., the RMSE would ideally equal noise SD).

We use CV to ensure the best performance is achieved for LAGP, GBT, and NN. For GAGP, we choose $n_0 = 500$, $s = 6$, and $n_s = 250$ and equally distribute the samples among the $m = 30000/(500 + 6\times250) = 15$ GPs (i.e., each GP has 2000 samples). The results are summarized in Table 1 (for small noise SD) and Table 2 (for large noise SD) and indicate that (i) GAGP consistently outperforms LAGP and GBT, (ii) both GAGP and NN recover the true amount of added noise with high accuracy, and (iii) GAGP achieves very similar results to NN. Given the large number of data points, the effect of sample-to-sample randomness on the results is very small and hence not reported.

We highlight that the performance of GAGP in each case could have been improved even further by tuning its parameters via CV (which was done for LAGP, GBT, and NN). Potential parameters include n_0, s, n_s, and $f_i(\boldsymbol{x})$. However, we *intentionally* avoid this tuning to demonstrate GAGP's flexibility, generality, and ease of use.

In engineering design, it is highly desirable to employ interpretable methods and tools that facilitate the knowledge discovery and decision-making processes. Contrary to many supervised learning techniques such as NNs and random forests that are black boxes, the structure of GPs can provide qualitative insights. To demonstrate, we rewrite Eq. (3) as $r(\boldsymbol{x}, \boldsymbol{x}') = \exp\left\{-\sum_{i=1}^{d} 10^{\omega_i}(x_{(i)} - x'_{(i)})^2\right\}$. If $\omega_i \ll 0$ (e.g., $\omega_i = -10$), then variations along the ith dimension (i.e., $x_{(i)}$) do not contribute to the summation and, subsequently, to the correlation between $\boldsymbol{x}$ and $\boldsymbol{x}'$ (see Fig. 3 for a 1D illustration). This contribution increases as the magnitude of ω_i increases. In a GP with a constant mean of β, all the effect of inputs on the output is captured through $r(\boldsymbol{x}, \boldsymbol{x}')$. Hence, as ω_i decreases, the effect of x_i on the output decreases as well. We illustrate this feature with a 2D example as

Table 1 Root mean squared error with small noise

	Noise SD	LAGP	GBT	NN	GAGP
Ex1 (1D)	0.2	1.271	0.209	**0.200**	**0.200**
Ex2 (4D)	0.1	1.386	0.121	**0.100**	0.103
Ex3 (7D)	0.1	0.129	0.118	**0.100**	**0.100**
Ex4 (10D)	0.01	0.210	0.048	0.012	**0.011**
Ex5 (20D)	0.1	1.450	0.351	**0.101**	0.103

Note: Smallest errors are in bold.

Table 2 Root mean squared error with large noise

	Noise SD	LAGP	GBT	NN	GAGP
Ex1 (1D)	2	2.270	2.062	**2.000**	**2.000**
Ex2 (4D)	1	1.739	1.123	**1.002**	1.009
Ex3 (7D)	1	1.037	1.098	**1.002**	**1.002**
Ex4 (10D)	0.1	0.234	0.120	**0.102**	**0.102**
Ex4 (20D)	1	1.911	1.155	1.011	**1.001**

Note: Smallest errors are in bold.

Table 3 Effect of sample size and underlying function's nonlinearity on hyperparameter estimates

		$\alpha=2$	$\alpha=4$	$\alpha=6$
$n=1000$	$\hat{\omega}_1$	2.39	3.11	3.59
	$\hat{\omega}_2$	−1.98	−2.12	−2.11
$n=2000$	$\hat{\omega}_1$	2.38	3.10	3.54
	$\hat{\omega}_2$	−1.92	−2.18	−2.22
Total SI	x_1	1.00	1.00	1.00
	x_2	0.18	0.05	0.03

Note: The Sobol's total sensitivity indices (SIs) are also included.

follows. Assume $y=f(x_1,\ x_2;\alpha)=\sin(2x_1x_2)+\alpha\cos(\alpha x_1^2)$, $-\pi\le x_1,\ x_2\le\pi$ for $\alpha=2$, 4, 6. Three points regarding f are highlighted:

(1) x_1 is more important than x_2 since both $\sin(2x_1x_2)$ and $\alpha\cos(\alpha x_1^2)$ depend on x_1 (note that $\alpha\neq 0$), while x_2 only affects the first term.
(2) As α increases, the relative importance of x_1 (compared with x_2) increases because the amplitude of $\alpha\cos(\alpha x_1^2)$ increases.
(3) As α increases, y depends on x_1 with growing nonlinearity because the frequency of $\alpha\cos(\alpha x_1^2)$ increases.

The first two points can be verified by calculating Sobol's total sensitivity indices (SIs) for x_1 and x_2 in f; see Table 3. These indices range from 0 to 1, with higher values indicating more sensitivity to the input. Here, the SI of x_1 is always 1, but the SI of x_2 decreases as α increases, indicating that the *relative* importance of x_1 on y increases as α increases.

Note that calculating the Sobol's SIs involves evaluating f for hundreds of thousands of samples, while a GP can distill similar sensitivities from a dataset. To show this, for each α, we fit two GPs: one with $n=1000$ training data and the other with $n=2000$. The hyperparameter estimates are summarized in Table 3 and indicate that:

- For each α, $\hat{\omega}_1$ is larger than $\hat{\omega}_2$, implying x_1 is more important than x_2.
- As α increases, $\hat{\omega}_1$ increases (x_1 becomes more important), while $\hat{\omega}_2$ changes negligibly (the underlying functional relation between x_2 and y does not depend on α).
- For a given α, the estimates change insignificantly when n is increased.

The above feature is present in GAGP as well and depicted by the convergence histories for Ex3 and Ex5 in Figs. 4 and 5, respectively. Similar to Fig. 1, it is evident that the estimated roughness parameters do not change noticeably as more samples are used in training (only 6 of the 20 roughness parameters are plotted in Fig. 5 for a clearer illustration). The values of these parameters can determine which inputs (and to what extent) affect the output. For instance, in Ex5, ω_8 is very small so the output must be almost insensitive to x_8. In addition, since $\omega_4\cong\omega_{20}$, it is expected that the corresponding inputs should affect y similarly. These observations agree with the analytical relation between $\boldsymbol{x}$ and y in Ex5, where y is independent of x_8 and symmetric with respect to x_4 and x_{20}. By using GAGP, such information can also be extracted from a training dataset whose underlying functional relation is unknown and subsequently used for sensitivity analysis or dimensionality reduction (e.g., in Ex5, x_8 and x_{16} can be excluded from the training data).

In Figs. 4 and 5, the estimated variance, $\delta\hat{\sigma}^2$, varies closely around the true noise variance. It provides a useful quantitative measure for the expected predictive power (e.g., RMSE in future uses of the model). In addition, like $\hat{\boldsymbol{\omega}}$, its convergence history helps in determining whether sufficient samples have been used in training. First, the number of training samples should be increased until $\delta\hat{\sigma}^2$ does not fluctuate noticeably. Second, via k-fold CV during training, the true noise variance should ideally be recovered by calculating the RMSE associated with predicting the samples in the ith fold (when fold i is not used in training). If these two values differ significantly, s (or n_s) should be increased. For instance, if the fluctuations on the right panel in Fig. 5 had been large or far from the noise variance, we would have increased s (from 6 to, e.g., 10) or n_s (from 250 to, e.g., 500).

We close this section with some theoretical discussions related to the use of the same hyperparameters within a collection of

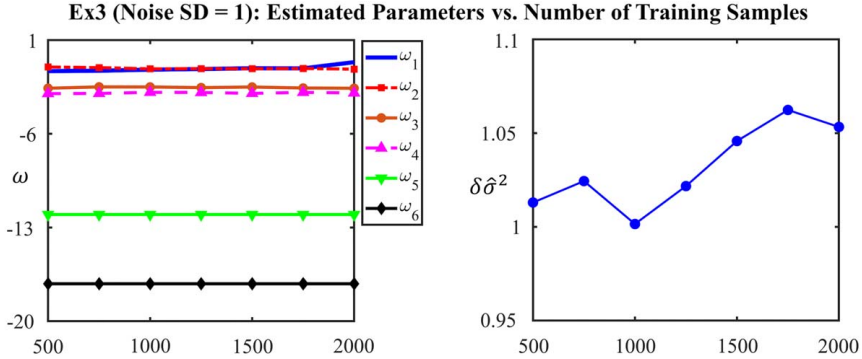

Fig. 4 Convergence history in Ex3 as the number of training samples is increased from 500 to 2000

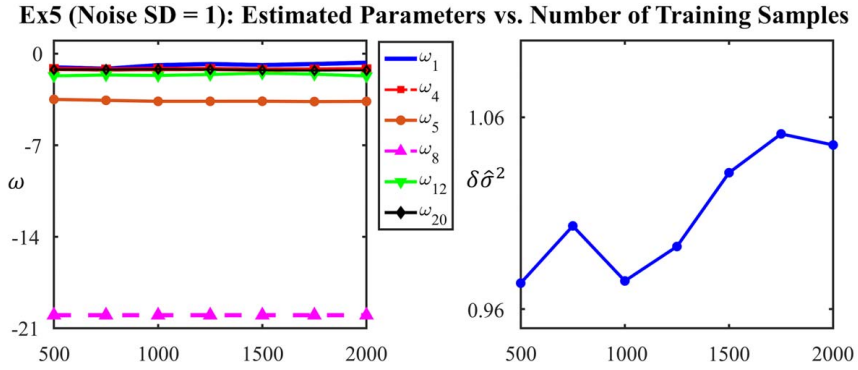

Fig. 5 Convergence history in Ex5 as the number of training samples is increased from 500 to 2000. For clearer demonstration, only 6 of the 20 roughness parameters are plotted.

independent GPs. In Bayesian experimental design [55], multidimensional integrals ubiquitously arise when maximizing the expected information gain, i.e., $E[I]$. By quantifying I using Kullback–Leibler divergence [56], it can be shown [57] that $E[I] = \int \left(\int \log\left(\frac{p(\boldsymbol{\theta}|\mathbf{y}_i)}{p(\boldsymbol{\theta})}\right) p(\boldsymbol{\theta}|\mathbf{y}_i) d\boldsymbol{\theta} \right) p(\mathbf{y}_i) d\mathbf{y}_i$, where $\boldsymbol{\theta}$ are the hyperparameters (to be estimated), $\mathbf{y}_i$ are the observables in the ith experiment, and $p(\cdot)$ is the probability density function. The nested integral renders maximizing $E[I]$ prohibitively expensive. To address this and facilitate integration, Laplace's theorem is used to approximate $p(\boldsymbol{\theta}|\mathbf{y}_i)$ via a multivariate Gaussian likelihood or log likelihood, such as in Eq. (4). Following the central limit theorem, the accuracy of this approximation increases as the number of data points increases since the likelihood would more closely resemble a unimodal multivariate Gaussian curve [58]. With GAGP, we essentially make the same approximation, i.e., Eq. (4) approximates a unimodal multivariate Gaussian curve in the log space whose minimizer insignificantly changes when the training data are massive (note that the function value, L, does change; see Fig. 1).

5 Data-Driven Design of Metamaterials

To demonstrate the application of GAGP in engineering design, we employ it in a new data-driven method for the optimization of metamaterial unit cells using big data. Although various methods, e.g., TO and GA, have been applied to design metamaterials with prescribed properties, these are computationally intensive and suffer from sensitivity to the initial guess as well as disconnected boundaries when using multiple unit cells. A promising solution is to construct a large database of precomputed unit cells (also known as microstructures or building blocks) for efficient selection of well-connected unit cells from the database as well as inexpensive optimization of new unit cells [9–12]. However, with the exception of Ref. [12] where unit cells are parameterized via geometric features like beam thickness, research in this area thus far use high-dimensional geometric representations (e.g., signed distance functions [9] or voxels [11]) that increase the memory demand and the complexity of constructing machine learning models that link structures to properties. Therefore, reducing the dimension of the unit cell is a crucial step.

In this work, we reduce the dimension of the unit cells in our metamaterial database with spectral shape descriptors based on the LB operator. We then employ GAGP to learn how the effective stiffness tensor of unit cells changes as a function of their LB descriptors. After the GAGP model is fitted, we use it to discover unit cells with desired properties through inverse optimization. Furthermore, to present the advantages of a large unit cell database and GAGP, we compare the results with those obtained using an NN model fitted to the same dataset and a conventional GP model fitted to a smaller dataset.

5.1 Metamaterials Database Generation. We propose a novel two-stage pipeline inspired by Ref. [11] to generate a large training dataset of unit cells and their corresponding homogenized properties. For demonstration, our primary properties of interest are the components of the stiffness tensor, E_x, E_y, and E_{xy}. As elaborated below, our method starts by building an initial dataset and then proceeds to efficiently cover the input (geometry) and output (property) spaces as widely as possible.

To construct the initial dataset in stage one, we select design targets in the property space (the 3D space spanned by E_x, E_y, and E_{xy}). As the bounds of the property space are unknown a priori, we sample 1000 points uniformly distributed in $[0, 1]^3$. Then, we use the solid isotropic material with penalization TO method [59] to find the orthotropic unit cells corresponding to each target. This stage generates 358 valid structures. The remaining 642 points do not result in feasible unit cells mainly because (i) the uniform sampling places some design targets in theoretically infeasible regions and (ii) the TO method may fail to meet targets due to sensitivity to the initial shape, which is difficult to guess without prior knowledge. The properties of these 358 structures are shown in Fig. 6, where the Poisson's ratio is used instead of E_{xy} for a better illustration of the space.

In stage two, we expand the initial database via an iterative stochastic shape perturbation algorithm that by-passes TO by generating distorted structures with slightly different properties from the original ones. Specifically, the following radial distortion model is used to perturb an existing shape:

$$x_{new} = \begin{cases} x_c + \dfrac{r_{new}}{r_{old}}(x_{old} - x_c) & \text{if } r_{old} \le R_0 \\ x_{old} & \text{if } r_{old} > R_0 \end{cases} \tag{16}$$

where x_{new} and x_{old} are the coordinates of the new and original pixel locations, x_c is the coordinate vector of the distortion center, r_{new} and r_{old} are the new and original distances to the distortion center, respectively, and R_0 is the outer distortion radius. r_{new} can be expressed as follows:

$$r_{new} = \begin{cases} \dfrac{1}{2}R_0\left(1 - \cot\left(\dfrac{\gamma}{2}\right) - \beta\right) & \text{if } \gamma > 0 \\ \dfrac{1}{2}R_0\left(1 - \cot\left(\dfrac{\gamma}{2}\right) + \beta\right) & \text{if } \gamma < 0 \\ r_{old} & \text{otherwise} \end{cases} \tag{17}$$

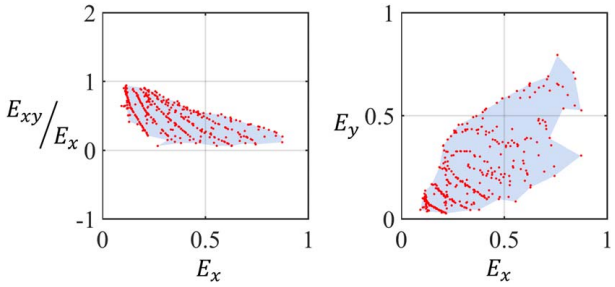

Fig. 6 The property space of the initial database with 358 structures

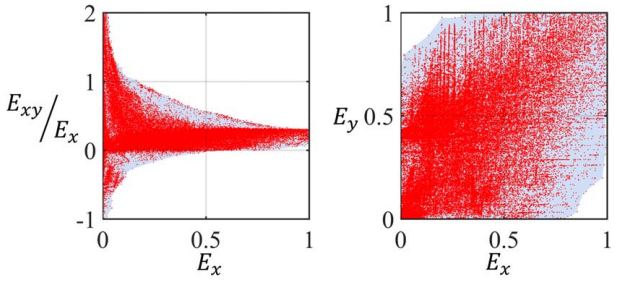

Fig. 7 The property space of the expanded database with 88,000 structures. The shaded regions indicate the boundary of the property space.

where

$$\beta = \sqrt{\frac{2}{\sin^2\left(\frac{\gamma}{2}\right)} - \left(1 + \cot\left(\frac{\gamma}{2}\right) - \frac{2r_{old}}{R_0}\right)^2} \tag{18}$$

and $\gamma \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ is the angle that controls the amount of distortion. Considering the orthotropic symmetry of the unit cells, only a quarter of the original structure is distorted and then reassembled to realize the full structure. We adopt the distortion model in Eq. (16) for two reasons. First, its parameters (R_0, γ, and x_c) have clear interpretations and hence can be easily tuned. In our case, they are all set as random variables with standard uniform distributions to generate a wide range of structures. Second, it preserves the topology of the original unit cell and introduces negligible artifacts (e.g., disconnections and checkerboard patterns) upon perturbation.

To better cover the property space, the database is populated iteratively. In each iteration, we first calculate the following score for all available unit cells:

$$\text{Score} = \frac{1}{(d+\varepsilon)2^{\rho}} \tag{19}$$

where d is the Euclidean (L2) distance between the stiffness tensor components of each unit cell to the boundaries of the region enclosing the current property space (see the shaded areas in Figs. 6 and 7), ρ is the number of data points inside the neighborhood within a given radius in the property space (in our experience, sampling is more uniform when the radius equals 0.05), and $\varepsilon \ll 1$ is used to avoid singularity. Then, we select the N points with the highest scores for stochastic perturbation. After each iteration, newly generated unit cells are verified and discarded if they contain infeasible features such as isolated parts. The properties of new feasible structures are then calculated via numerical homogenization [60] and added to the dataset. The perturbation is repeated until the boundary of property space does not expand significantly ($\Delta d < 0.1$), and the points inside the boundaries are relatively dense (average $\rho > 500$). By using this strategy, the database is expanded in our test case from 358 to 88,000 unit cells that cover a wider range of properties (see Fig. 7). We note that the machine learner (GAGP in our case) may also be used to detect the infeasible structures; we do not consider this in the current work but may include it in our future studies.

5.2 Unit Cell Dimension Reduction via Spectral Shape Descriptors. In the previous section, each unit cell in the database is represented by 50×50 pixels. For dimension reduction, we use spectral shape descriptors as they retain geometric and physical information. Specifically, we use the LB spectrum, also known as Shape-DNA, which can be directly calculated for any unit cell shape [61,62].

The LB spectrum is an effective descriptor for the metamaterials database for several reasons: (i) It has a powerful discrimination ability and has been successfully applied to shape matching and classification in computer vision despite being one of the simplest spectral descriptors. (ii) All of the complex structures in our orthotropic metamaterials database can be uniquely characterized with the first 10–15 eigenvalues in the LB spectrum. (iii) The spectrum embodies some geometrical information, including perimeter, area, and Euler number. This can be beneficial for the construction of the machine learning model as less training data may be required to obtain an accurate model compared with voxel- or point-based representations. (iv) Similar shapes have close LB spectrum, which may also help the supervised learning task.

The calculation of the LB spectrum for each unit cell is as follows. For a real-valued function f defined on a Riemannian manifold [61], the Helmholtz equation reads as follows:

$$\Delta f = -\lambda f \tag{20}$$

where the Laplace–Beltrami operator Δ is defined as follows:

$$\Delta := div(gradf) \tag{21}$$

The eigenvalues of the Helmholtz equation are the LB spectrum and denoted as follows:

$$0 \le \lambda_1 \le \lambda_2 \le \cdots < \infty \tag{22}$$

Our unit cells reside in a 2D planar domain and can be considered as a special case of Riemannian manifolds. Therefore, we focus on the LB spectrum of a 2D shape under Dirichlet boundary conditions, which reduce the Helmholtz equation to a Laplacian eigenvalue problem:

$$\begin{aligned} \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} &= -\lambda f \quad \text{in } \Omega \\ f &= 0 \quad \text{on } \tau \end{aligned} \tag{23}$$

where Ω and τ are the interior and boundaries of the domain of interest, respectively.

Finally, the finite element method is employed to obtain the LB spectrum of unit cells [63]; see Fig. 8. It is noted that our 88,000 structures can be uniquely determined with only the first 16 non-zero eigenvalues, reducing the input dimension from $50 \times 50 = 2500$ pixels to 16 scalar descriptors. In general, the computation of the LB spectrum takes only a few seconds per unit cell on a single CPU (Intel(R) Xeon(R) Gold 6144 CPU at 3.50 GHz). Since these computations are performed once and in parallel, the runtime is acceptable.

5.3 Machine Learning: Linking LB Representation to Property via GAGP. Once the dataset is built, we follow the algorithm in Fig. 2 for machine learning, i.e., relating the LB representations of unit cells to their stiffness tensor. We use the same fitting parameters as in Sec. 4 ($n_0 = 500$, $s = 6$, $n_s = 250$), equally distribute the samples among the $m = 88000/(500 + 6 \times 250) = 44$ GPs, and use Eq. (10) to have a multiresponse model that leverages the correlation between the responses to have a higher predictive power. The convergence histories are provided in Fig. 9, where the trends are consistent with those in Sec. 4. It is observed that the

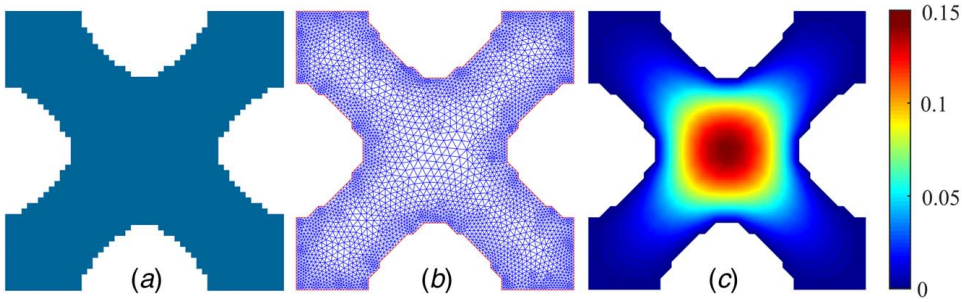

Fig. 8 Steps in the LB spectrum calculation: (*a*) original structure, (*b*) finite element mesh, and (*c*) the eigenfunction corresponding to the first eigenvalue, λ_1

16 estimated roughness parameters do not change noticeably once more than 1000 samples are used in training. In particular, 3 of the 16 roughness estimates, which correspond to λ_{14}, λ_{15}, and λ_{16} are very small, indicating that those LB descriptors do not affect the responses. The next largest estimate belongs to $\omega_{13} \cong -8$, which corresponds to λ_{13}. The rest of the estimates are all between 2.5 and 3, implying that the first 12 eigenvalues (shape descriptors) affect the responses similarly and nonlinearly (since large ω_i indicates rough response changes along dimension i). These observations agree well with the fact that the higher order eigenvalues generally explain less variability in the data. The estimated noise variances (one per response) also converge, with E_{xy} having the largest estimated noise variance in the data, which is potentially due to larger numerical errors in property estimation.

To illustrate the effect of expanding the training data from 385 to 88,000, we randomly select 28,000 samples for validation. Then, we evaluate the mean squared error (MSE) of the following two models on this test set: a conventional GP fitted to the initial 385 samples and a GAGP fitted to the rest of the data (i.e., to 60,000 samples, resulting in $m = 60{,}000/(500 + 6 \times 250) = 30$ models). To account for randomness, we repeat this process 20 times. The results are summarized in Table 4 and demonstrate that (i) increasing the dataset size (stage two in Sec. 5.1) creates a supervised learner with a higher predictive power (compare the mean of MSEs for GP and GAGP). (ii) GAGP is more robust to variations than GP (compare the variance of MSEs for GP and GAGP). (iii) With 60,000samples, the predictive power of GAGP is slightly lower than the case where the entire dataset is used in training (compare mean of MSEs for GAGP in Table 4 with the converged noise estimates in Fig. 9).

To assess the robustness of GAGP to data size, we repeat the above procedure but with 20,000 and 40,000 training samples instead of 60,000. For fair comparisons, the same validation sample size of 28,000 is used for each. The results are summarized in Table 5 and, by comparing them with those of GAGP in Table 4, indicate that increasing the sample size from 20,000 to 60,000 increases the predictive power and robustness. Note that, since $[n_0, n_s, s]$ are not changed when fitting GAGP, using more samples increases m, the number of GPs.

5.4 Data-Driven Unit Cell Optimization. Finally, we illustrate the benefits of the GAGP model in an inverse optimization scheme to realize unit cells with target stiffness tensor components and compare the results with those designed using other techniques. Establishing such an inverse link is highly desirable in structure design as it allows to efficiently achieve target elastic properties by avoiding expensive finite element simulations and tedious trial and error in TO. In addition, although not demonstrated in this work, such a link can provide multiple candidate unit cells with the same properties that, in turn, enable tiling different unit cells into a macrostructure while ensuring boundary compatibility.

Our data-driven optimization scheme has two steps: the search for the optimal LB spectrum and the reconstruction of the unit cell given the LB spectrum. By using the GAGP (or another) model, we directly search for the LB spectrum of the unit cell with the desired properties. The search problem is formulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{\lambda}} \|\boldsymbol{E}^t - \boldsymbol{E}^p\|_\infty \\ &\text{s.t.} \quad \lambda_{i-1} \le \lambda_i \\ &0.9\lambda_i^0 \le \lambda_i \le 1.1\lambda_i^0 \end{aligned} \tag{24}$$

where $\boldsymbol{E}^t$ and $\boldsymbol{E}^p$ are the vectorized forms of, respectively, the target and predicted stiffness tensors. $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_{16}]$ and $\boldsymbol{\lambda}^0$ are the LB spectra of, respectively, candidate unit cell for the target stiffness tensor and the unit cell closest to the prescribed properties in the property space (λ_i is the ith order eigenvalue). We choose GA for

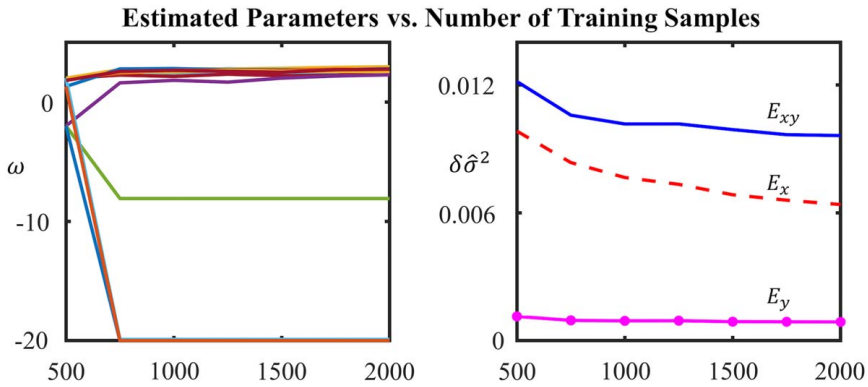

Fig. 9 Convergence history as the number of training samples is increased from 500 to 2000. The 16 colored lines in the left panel indicate the estimation histories of the 16 hyperparameters.

Table 4 MSE errors on 28,000 random samples for GP and GAGP

	Mean of MSE			Variance of MSE ($\times 10^6$)		
	E_x	E_y	E_{xy}	E_x	E_y	E_{xy}
GP	0.048	0.007	0.028	39	5.5	0.45
GAGP	0.008	0.001	0.011	0.12	0.0007	0.04

Note: The mean and variance of MSE are calculated over 20 random repetitions.

Table 5 Effect of sample size on GAGP performance

Sample size	Mean of MSE			Variance of MSE ($\times 10^6$)		
	E_x	E_y	E_{xy}	E_x	E_y	E_{xy}
20,000	0.0090	0.0024	0.017	0.30	0.004	0.08
40,000	0.0084	0.0016	0.013	0.16	0.001	0.05

Note: MSE errors (and their variances) are obtained via 28,000 random samples in 20 repetitions.

optimization since the GAGP model is cheap to run and GA ensures global optimality for multivariate and constrained problems. Note that, to ensure fast convergence during GA, we limit the search space for GA using the LB spectrum of the unit cell in the training dataset whose properties are closest to $\boldsymbol{E}^t$.

After obtaining the optimal LB spectrum, we use a level set method to reconstruct the corresponding unit cell [64] while employing squared residuals of the LB spectrum as the objective function. For faster convergence, the unit cell closest to the optimal LB spectrum in the spectrum space is taken as the initial guess in the reconstruction process.

In the following two examples, the goal is to design structures with desired E_x, E_y, and E_{xy} (see the target properties in Fig. 10). For each example, three unit cells are designed using different models: GAGP, NN, and GP (GAGP and NN use the entire dataset while GP uses the initial one with 358 structures). The results are visualized in Fig. 10 and demonstrate that the unit cells identified from GAGP and NN are more geometrically diverse than those obtained via GP. This is a direct result of populating the large dataset with perturbed structures and, in turn, providing the GA search process with a wider range of initial seeds. While we utilized our entire database for GAGP and NN in an attempt to provide more diversity for new designs, a smaller or less diverse training dataset could potentially achieve similar results; such a study is left for future work. We also note that the unit cells designed with GP are similar in shape but different in the size of the center hole, which leads to the significant change in properties.

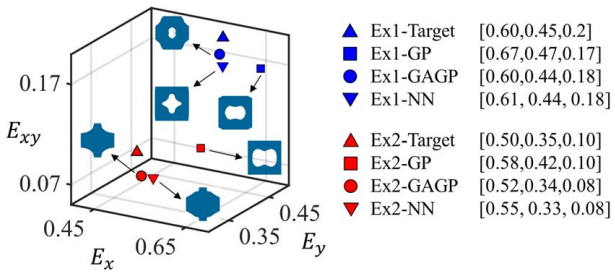

Fig. 10 Reconstructed unit cells in the three examples via GAGP, GP, and NN. The results are visualized in the property space and the triplets in the legend correspond to [E_x, E_y, E_{xy}].

From a quantitative point of view, our data-driven design method with the large database can, compared with the small dataset case, discover unit cells with properties that are closer to the target values. For instance, in Ex1, the GAGP and NN results using the large dataset achieve the target E_x, whereas the GP result with the small dataset differs from the target by around 12%. Ex2 shows a similar pattern, with the GAGP, NN, and GP results differing from the target E_x by 4%, 10%, and 16%, respectively. When the small dataset is used, the greater deviations from the target properties can be mainly attributed to insufficient training samples and the relatively small search space. This reinforces the need for a large database of unit cells in the data-driven design of metamaterials, along with an expedient machine learning method for big data. Moreover, unit cells designed with GAGP have smaller deviations than those with NN.

6 Conclusion and Future Works

In this work, we proposed a novel approach, GAGP, to enable GP modeling of massive datasets. Our method is centered on the observation that the hyperparameter estimates of a GP converge to some limit values, $\hat{\boldsymbol{\omega}}_\infty$, as more training samples are added. We introduced an intuitive and straightforward method to find $\hat{\boldsymbol{\omega}}_\infty$ and, subsequently, build a collection of independent GPs that all use the converged $\hat{\boldsymbol{\omega}}_\infty$ as their hyperparameters. These GPs randomly distribute the entire training dataset among themselves, which allows inference based on the entire dataset by pooling the predictions from the individual GPs. The training cost of GAGP primarily depends on the initial optimization with n_s data points and the s optimizations thereafter. The former cost is the same as fitting a conventional GP with n_s samples. The latter is an additional cost but is generally manageable since a single initial guess close to the global optimum is available at each iteration. The cost of building m GPs is negligible compared with these optimization costs. The prediction cost, although being m times larger than a conventional GP, is small enough for practical applications.

With analytical examples, we demonstrated that GAGP achieves very high predictive power that matches, and in some cases exceeds, that of state-of-the-art machine learning methods such as NNs and boosted trees. Unlike these latter methods, GAGP is easy to fit and interpret, which makes it particularly useful in engineering design with big data. Although the predictive power of GAGP increases as the size of the training data increases, so does the cost of fitting and training; it may be necessary to choose part of the data if resources are limited. We also note that, throughout, we assumed that the training samples are not ordered or highly correlated. If they are, randomization and appropriate transformations are required. In addition, we assumed stationary noise with an unknown variance. Considering nonstationary noise variance would be an interesting and useful extension for GAGP. Thrifty sample selection for model refinement (instead of randomly taking subsets of training data) can also improve the predictive power of GAGP and is planned for our future works.

As a case study, we applied GAGP to a data-driven metamaterials unit cell design process that attains desired elastic properties by transforming the complex material design problem into a parametric one using spectral descriptors. After mapping reduced-dimensional geometric descriptors (LB spectrum) to properties through GAGP, unit cells with properties close to the target values are discovered by finding the optimal LB spectrum with inverse optimization. This framework provides a springboard for a salient new approach to systematically and efficiently design metamaterials with optimized boundary compatibility, spatially varying properties, and multiple functionalities.

Acknowledgment

The authors are grateful to Professor K. Svanberg from the Royal Institute of Technology, Sweden, for providing a copy of the MMA

code for metamaterial design. Support from the National Science Foundation (NSF) (Grant Nos. ACI 1640840 and OAC 1835782; Funder ID: 10.13039/501100008982) and the Air Force Office of Scientific Research (AFOSR FA9550-18-1-0381; Funder ID: 10.13039/100000181) are greatly appreciated. Ms. Yu-Chin Chan would like to acknowledge the NSF Graduate Research Fellowship Program (Grant No. DGE-1842165).

Nomenclature

d = input dimensionality
n = number of training samples
q = output dimensionality
s = number of times that n_s samples are added to n_0
$\boldsymbol{x}$ = vector of d inputs
$\boldsymbol{y}$ = vector of q outputs
L = objective function in MLE
$\boldsymbol{R}$ = sample correlation matrix of size $n \times n$
GP = Gaussian process
MLE = maximum likelihood estimation
δ = Nugget or jitter parameter
n_0 = number of initial random samples
n_s = number of random samples added to n_0 per iteration
$\boldsymbol{\omega}$ = roughness parameters of the correlation function
$\hat{\boldsymbol{\omega}}_\infty$ = estimate of $\boldsymbol{\omega}$ via MLE with very large training data

References

[1] Yan, W., Lin, S., Kafka, O. L., Lian, Y., Yu, C., Liu, Z., Yan, J., Wolff, S., Wu, H., and Ndip-Agbor, E., 2018, "Data-Driven Multi-Scale Multi-Physics Models to Derive Process–Structure–Property Relationships for Additive Manufacturing," Comput. Mech., **61**(5), pp. 521–541.
[2] Mozaffar, M., Paul, A., Al-Bahrani, R., Wolff, S., Choudhary, A., Agrawal, A., Ehmann, K., and Cao, J., 2018, "Data-Driven Prediction of the High-Dimensional Thermal History in Directed Energy Deposition Processes via Recurrent Neural Networks," Manuf. Lett., **18**, pp. 35–39.
[3] Ghumman, U. F., Iyer, A., Dulal, R., Munshi, J., Wang, A., Chien, T., Balasubramanian, G., and Chen, W., 2018, "A Spectral Density Function Approach for Active Layer Design of Organic Photovoltaic Cells," ASME J. Mech. Des., **140**(11), p. 111408.
[4] Bessa, M. A., Bostanabad, R., Liu, Z., Hu, A., Apley, D. W., Brinson, C., Chen, W., and Liu, W. K., 2017, "A Framework for Data-Driven Analysis of Materials Under Uncertainty: Countering the Curse of Dimensionality," Comput. Method Appl. M., **320**, pp. 633–667.
[5] Bostanabad, R., Chen, W., and Apley, D. W., 2016, "Characterization and Reconstruction of 3D Stochastic Microstructures via Supervised Learning," J. Microsc., **264**(3), pp. 282–297.
[6] Bostanabad, R., Zhang, Y., Li, X., Kearney, T., Brinson, L. C., Apley, D. W., Liu, W. K., and Chen, W., 2018, "Computational Microstructure Characterization and Reconstruction: Review of the State-of-the-Art Techniques," Prog. Mater. Sci., **95**, pp. 1–41.
[7] Li, X., Yang, Z., Brinson, L. C., Choudhary, A., Agrawal, A., and Chen, W., 2018, "A Deep Adversarial Learning Methodology for Designing Microstructural Material Systems," ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Quebec City, Quebec, Canada, Aug. 26–29.
[8] Bostanabad, R., Bui, A. T., Xie, W., Apley, D. W., and Chen, W., 2016, "Stochastic Microstructure Characterization and Reconstruction via Supervised Learning," Acta Mater., **103**, pp. 89–102.
[9] Schumacher, C., Bickel, B., Rys, J., Marschner, S., Daraio, C., and Gross, M., 2015, "Microstructures to Control Elasticity in 3D Printing," ACM Trans. Graph., **34**(4), p. 136.
[10] Panetta, J., Zhou, Q., Malomo, L., Pietroni, N., Cignoni, P., and Zorin, D., 2015, "Elastic Textures for Additive Fabrication," ACM Trans. Graph., **34**(4), p. 135.
[11] Zhu, B., Skouras, M., Chen, D., and Matusik, W., 2017, "Two-Scale Topology Optimization With Microstructures," ACM Trans. Graph., **36**(5), p. 164.
[12] Chen, D., Skouras, M., Zhu, B., and Matusik, W., 2018, "Computational Discovery of Extremal Microstructure Families," Sci. Adv., **4**(1), p. eaao7005.
[13] LeCun, Y., Bengio, Y., and Hinton, G., 2015, "Deep Learning," Nature, **521**(7553), pp. 436–444.
[14] Chen, T., and Guestrin, C., "Xgboost: A Scalable Tree Boosting System," 2016, Proceedings of 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, Aug. 13–17, pp. 785–794.
[15] Hassaninia, I., Bostanabad, R., Chen, W., and Mohseni, H., 2017, "Characterization of the Optical Properties of Turbid Media by Supervised Learning of Scattering Patterns," Sci. Rep., **7**(1), p. 15259.
[16] Tao, S., Shintani, K., Bostanabad, R., Chan, Y.-C., Yang, G., Meingast, H., and Chen, W., 2017, "Enhanced Gaussian Process Metamodeling and Collaborative Optimization for Vehicle Suspension Design Optimization," ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Cleveland, OH, Aug. 6–9, Paper No. DETC2017-67976.
[17] Bostanabad, R., Liang, B., Gao, J., Liu, W. K., Cao, J., Zeng, D., Su, X., Xu, H., Li, Y., and Chen, W., 2018, "Uncertainty Quantification in Multiscale Simulation of Woven Fiber Composites," Comput. Method Appl. M., **338**, pp. 506–532.
[18] Zhang, W., Bostanabad, R., Liang, B., Su, X., Zeng, D., Bessa, M. A., Wang, Y., Chen, W., and Cao, J., 2019, "A Numerical Bayesian-Calibrated Characterization Method for Multiscale Prepreg Preforming Simulations With Tension-Shear Coupling," Compos. Sci. Technol., **170**, pp. 15–24.
[19] Gramacy, R. B., and Lee, H. K. H., 2008, "Bayesian Treed Gaussian Process Models With an Application to Computer Modeling," J. Am. Stat. Assoc., **103**(483), pp. 1119–1130.
[20] Kim, H.-M., Mallick, B. K., and Holmes, C., 2005, "Analyzing Nonstationary Spatial Data Using Piecewise Gaussian Processes," J. Am. Stat. Assoc., **100**(470), pp. 653–668.
[21] Rasmussen, C. E., 2006, *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA.
[22] Tresp, V., 2000, "A Bayesian Committee Machine," Neural Comput., **12**(11), pp. 2719–2741.
[23] Herbrich, R., Lawrence, N. D., and Seeger, M., 2003, "Fast Sparse Gaussian Process Methods: The Informative Vector Machine," Proc. Advances in Neural Information Processing Systems, pp. 625–632.
[24] Seeger, M., Williams, C., and Lawrence, N., 2003, "Fast Forward Selection to Speed Up Sparse Gaussian Process Regression," Proceedings of Artificial Intelligence and Statistics 9.
[25] Williams, C. K., and Seeger, M., 2001, "Using the Nyström Method to Speed Up Kernel Machines," Proceedings of Advances in Neural Information Processing Systems, pp. 682–688.
[26] Rasmussen, C. E., and Quinonero-Candela, J., 2005, "Healing the Relevance Vector Machine Through Augmentation," Proceedings of 22nd International Conference on Machine Learning, Aug. 7–11, pp. 689–696.
[27] Wahba, G., 1990, Spline Models for Observational Data, Vol. 59, Siam.
[28] Gramacy, R. B., and Apley, D. W., 2015, "Local Gaussian Process Approximation for Large Computer Experiments," J. Comput. Graph. Stat., **24**(2), pp. 561–578.
[29] Garcia, D. J., Mozaffar, M., Ren, H. Q., Correa, J. E., Ehmann, K., Cao, J., and You, F. Q., 2019, "Sustainable Manufacturing With Cyber-Physical Discrete Manufacturing Networks: Overview and Modeling Framework," ASME J. Manuf. Sci. Eng., **141**(2), p. 021013.
[30] Mozaffar, M., Ndip-Agbor, E., Stephen, L., Wagner, G. J., Ehmann, K., and Cao, J., 2019, "Acceleration Strategies for Explicit Finite Element Analysis of Metal Powder-Based Additive Manufacturing Processes Using Graphical Processing Units," Comput. Mech., pp. 1–16.
[31] Han, Y., and Lu, W. F., 2018, "A Novel Design Method for Nonuniform Lattice Structures Based on Topology Optimization," ASME J. Mech. Des., **140**(9), p. 091403.
[32] Li, D., Dai, N., Tang, Y., Dong, G., and Zhao, Y. F., 2019, "Design and Optimization of Graded Cellular Structures With Triply Periodic Level Surface-Based Topological Shapes," ASME J. Mech. Des., **141**(7), p. 071402.
[33] Du, Z., Zhou, X.-Y., Picelli, R., and Kim, H. A., 2018, "Connecting Microstructures for Multiscale Topology Optimization With Connectivity Index Constraints," ASME J. Mech. Des., **140**(11), p. 111417.
[34] Fu, J., Xia, L., Gao, L., Xiao, M., and Li, H., 2019, "Topology Optimization of Periodic Structures With Substructuring," ASME J. Mech. Des., **141**(7), p. 071403.
[35] Bostanabad, R., Kearney, T., Tao, S., Apley, D. W., and Chen, W., 2018, "Leveraging the Nugget Parameter for Efficient Gaussian Process Modeling," Int. J. Numer Meth. Eng., **114**(5), pp. 501–516.
[36] MacDonald, B., Ranjan, P., and Chipman, H., 2015, "GPfit: An RPackage for Fitting a Gaussian Process Model to Deterministic Simulator Outputs," J. Stat. Software, **64**, p. 12.
[37] Plumlee, M., and Apley, D. W., 2017, "Lifted Brownian Kriging Models," Technometrics, **59**(2), pp. 165–177.
[38] Ranjan, P., Haynes, R., and Karsten, R., 2011, "A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data," Technometrics, **53**(4), pp. 366–378.
[39] Sacks, J., Schiller, S. B., and Welch, W. J., 1989, "Designs for Computer Experiments," Technometrics, **31**(1), pp. 41–47.
[40] Toal, D. J. J., Bressloff, N. W., and Keane, A. J., 2008, "Kriging Hyperparameter Tuning Strategies," AIAA J., **46**(5), pp. 1240–1252.
[41] Audet, C., and Dennis, J. E., Jr, 2002, "Analysis of Generalized Pattern Searches," SIAM J. Optim., **13**(3), pp. 889–903.
[42] Zhao, L., Choi, K., and Lee, I., 2011, "Metamodeling Method Using Dynamic Kriging for Design Optimization," AIAA J., **49**(9), pp. 2034–2046.
[43] Toal, D. J., Bressloff, N., Keane, A., and Holden, C., 2011, "The Development of a Hybridized Particle Swarm for Kriging Hyperparameter Tuning," Eng. Optim., **43**(6), pp. 675–699.
[44] Conti, S., Gosling, J. P., Oakley, J. E., and O'Hagan, A., 2009, "Gaussian Process Emulation of Dynamic Computer Codes," Biometrika, **96**(3), pp. 663–676.
[45] Arendt, P. D., Apley, D. W., and Chen, W., 2012, "Quantification of Model Uncertainty: Calibration, Model Discrepancy, and Identifiability," ASME J. Mech. Des., **134**(10), p. 100908.
[46] Arendt, P. D., Apley, D. W., Chen, W., Lamb, D., and Gorsich, D., 2012, "Improving Identifiability in Model Calibration Using Multiple Responses," ASME J. Mech. Des., **134**(10), p. 100909.

[47] Bayarri, M., Berger, J., Cafeo, J., Garcia-Donato, G., Liu, F., Palomo, J., Parthasarathy, R., Paulo, R., Sacks, J., and Walsh, D., 2007, "Computer Model Validation With Functional Output," Ann. Stat., **35**(5), pp. 1874–1906.
[48] Conti, S., and O'Hagan, A., 2010, "Bayesian Emulation of Complex Multi-Output and Dynamic Computer Models," J. Stat. Plann. Inference, **140**(3), pp. 640–651.
[49] Sobol, I. M., 1967, "On the Distribution of Points in a Cube and the Approximate Evaluation of Integrals," Zh. Vychisl. Mat. Mat. Fiz., **7**(4), pp. 784–802.
[50] Sobol, I. M., 1998, "On Quasi-Monte Carlo Integrations," Math. Comput. Simulat., **47**(2–5), pp. 103–112.
[51] Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., and Tibshirani, R., 2009, *The Elements of Statistical Learning*, Springer, New York.
[52] Bastos, L. S., and O'Hagan, A., 2009, "Diagnostics for Gaussian Process Emulators," Technometrics, **51**(4), pp. 425–438.
[53] Ben-Ari, E. N., and Steinberg, D. M., 2007, "Modeling Data From Computer Experiments: An Empirical Comparison of Kriging With MARS and Projection Pursuit Regression," Qual. Eng., **19**(4), pp. 327–338.
[54] Le, B., Yvonnet, J., and He, Q. C., 2015, "Computational Homogenization of Nonlinear Elastic Materials Using Neural Networks," Int. J. Numer Meth. Eng., **104**(12), pp. 1061–1084.
[55] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B., 2013, *Bayesian Data Analysis*, CRC Press, Boca Raton, FL.
[56] Kullback, S., 1997, *Information Theory and Statistics*, Dover Publications, New York.
[57] Long, Q., Scavino, M., Tempone, R., and Wang, S., 2013, "Fast Estimation of Expected Information Gains for Bayesian Experimental Designs Based on Laplace Approximations," Comput. Method Appl. M., **259**(Supplement C), pp. 24–39.
[58] Tierney, L., and Kadane, J. B., 1986, "Accurate Approximations for Posterior Moments and Marginal Densities," J. Am. Stat. Assoc., **81**(393), pp. 82–86.
[59] Xia, L., and Breitkopf, P., 2015, "Design of Materials Using Topology Optimization and Energy-Based Homogenization Approach in Matlab," Struct. Multidiscip. Optim., **52**(6), pp. 1229–1241.
[60] Andreassen, E., and Andreasen, C. S., 2014, "How to Determine Composite Material Properties Using Numerical Homogenization," Comp. Mater. Sci., **83**, pp. 488–495.
[61] Reuter, M., Wolter, F.-E., and Peinecke, N., 2006, "Laplace–Beltrami Spectra as 'Shape-DNA' of Surfaces and Solids," Comput.-Aided Des., **38**(4), pp. 342–366.
[62] Lian, Z., Godil, A., Bustos, B., Daoudi, M., Hermans, J., Kawamura, S., Kurita, Y., Lavoué, G., Van Nguyen, H., Ohbuchi, R., Ohkita, Y., Ohishi, Y., Porikli, F., Reuter, M., Sipiran, I., Smeets, D., Suetens, P., Tabia, H., and Vandermeulen, D., 2013, "A Comparison of Methods for Non-Rigid 3D Shape Retrieval," Pattern Recognit., **46**(1), pp. 449–461.
[63] Su, S., 2010, "Numerical Approaches on Shape Optimization of Elliptic Eigenvalue Problems and Shape Study of Human Brains," Ph.D. thesis, The Ohio State University, Columbus, OH.
[64] Zhu, S., 2018, "Effective Shape Optimization of Laplace Eigenvalue Problems Using Domain Expressions of Eulerian Derivatives," J. Optim. Theory Appl., **176**(1), pp. 17–34.