DRAFT
Proceedings of the ASME 2019
International Design Engineering Technical Conferences
and Computers and Information in Engineering Conferences
IDETC/CIE2019
August 18-21, 2019, Anaheim, CA, USA

IDETC2019-98027

GLOBALLY APPROXIMATE GAUSSIAN PROCESSES FOR BIG DATA WITH AN APPLICATION TO DATA-DRIVEN METAMATERIALS DESIGN

Ramin Bostanabad
Northwestern University
Evanston, IL, USA

Yu-Chin Chan
Northwestern University
Evanston, IL, USA

Liwei Wang
Shanghai Jiao Tong University
Shanghai, China

Ping Zhu
Shanghai Jiao Tong University
Shanghai, China

Wei Chen[1]
Northwestern University
Evanston, IL, USA

ABSTRACT
We introduce a novel method to enable Gaussian process (GP) modeling of massive datasets, called globally approximate Gaussian process (GAGP). Unlike most largescale 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 an ensemble of independent GPs that distribute the entire training dataset among themselves and use the same hyperparameters. This is based on the observation that the GP hyperparameter estimates negligibly change as the size of the training data exceeds a certain level that can be estimated in a systematic way. For inference, the predictions from all GPs in the ensemble are pooled which allows to efficiently exploit the entire training dataset for prediction. Through analytical examples, we demonstrate that GAGP achieves very high predictive power that matches (and in some cases exceeds) that of state-of-the-art machine learning methods. We illustrate the application of GAGP in engineering design with a problem on data-driven metamaterials design where it is used 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.

Keywords: Gaussian processes, Supervised learning, Big data, Metamaterials

NOMENCLATURE

n	Number of training samples
GP	Gaussian process
d	Input dimensionality
$\boldsymbol{x}$	Vector of d inputs
q	Output dimensionality
$\boldsymbol{y}$	Vector of q outputs
$\boldsymbol{R}$	Sample correlation matrix of size $n \times n$
$\boldsymbol{\omega}$	Roughness parameters of the correlation function
MLE	Maximum likelihood estimation
L	Objective function in MLE
δ	Nugget or jitter parameter
n_0	Number of initial random samples
n_s	Number of random samples added to n_0 per iteration
s	Number of times that n_s samples are added to n_0
$\widehat{\boldsymbol{\omega}}_\infty$	Estimate of $\boldsymbol{\omega}$ via MLE with very large training data

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].

[1] Corresponding Author. Wilson Cook Professor in Engineering Design. Email: weichen@northwestern.edu

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 [13] (NNs) and gradient boosted trees [14] (GBT) are difficult to interpret and hence not suitable for engineering design. Gaussian process (GP) models (aka 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 address noise and smooth the data (i.e., to avoid interpolation) [15], and are very interpretable (i.e., they 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 nonstationary behavior, prediction via 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 and overestimates the variance [21]. In Bayesian committee machine (BCA) [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 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] (SR) 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 [28] (LAGP) 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 cannot 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 to enable GP modeling of massive datasets. The main idea of our approach is to build an ensemble of independent GPs that utilize a converged roughness parameter as their hypermeters. This is based on an empirical observation that the estimates of the GP hyperparameters don't change much as the size of the training data exceeds certain level. While having some common aspects with a few of the abovementioned works, our method is more massively scalable, can leverage multicore or GPU (graphical processing unit) computations [29, 30], and is applicable to very high-dimensional data with or without noise.

As mentioned earlier, big data has 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 discover 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 (in case multiple unit cells are simultaneously designed). We take a data-driven approach to address these challenges 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 GP modeling approach to link the geometrical descriptors of unit cells and 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 largescale supervised learning methods on four analytical problems. We demonstrate an application of the GP approach 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 (aka surrogates, metamodels, or models) can replace a computer simulator. The procedure is identical if the data is obtained from physical experiments. Let us denote the output and inputs of a computer simulator by, respectively, y and the d dimensional vector $\boldsymbol{x} = [x_{(1)}, x_{(2)}, \dots, x_{(d)}]^T$ 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 pre-determined set of basis functions, $\boldsymbol{\beta} = [\beta_{(1)}, \dots, \beta_{(h)}]^T$ are unknown weights, and $\xi(\boldsymbol{x})$ is a zero-mean GP characterized with its parametric covariance function, $c(\cdot,\cdot)$:

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

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 Gaussian correlation function being the most widely used one:

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

where $\boldsymbol{\Omega} = diag(\mathbf{10}^{\boldsymbol{\omega}})$ and $\boldsymbol{\omega} = [\omega_{(1)}, \omega_{(2)}, \dots, \omega_{(d)}]^T$, $-\infty < \omega_i < \infty$ 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 minimizing the computational costs [28, 31-35].

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

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

where $\log(\cdot)$ is the natural logarithm 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 $i,j = 1, \dots, n$. Setting the partial derivatives with respect to β and σ^2 to zero yields:

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

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

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

$$\widehat{\boldsymbol{\omega}} = \underset{\boldsymbol{\omega}}{\text{argmin}}\ nlog(\hat{\sigma}^2) + \log(|\boldsymbol{R}|) = \underset{\boldsymbol{\omega}}{\text{argmin}}\ L. \tag{7}$$

By numerically minimizing L in Eq. (7) one can find $\widehat{\boldsymbol{\omega}}$. Many global optimization methods such as genetic algorithms [36], pattern searches [37, 38], and particle swarm optimization [39] have been employed to solve for $\widehat{\boldsymbol{\omega}}$ in Eq. (7). However, gradient-based optimization techniques are commonly preferred due to their ease of implementation and superior computational efficiency [15, 16, 31]. To guarantee global optimality in this case, the optimization is done numerous times with different initial guesses.

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

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

where $\boldsymbol{g}(\boldsymbol{x}^*)$ is an $n \times 1$ vector with i^{th} 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 $\hat{\sigma}^2 r(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\boldsymbol{y} = [y_1, \dots, y_n]^T$ are the responses in the training dataset, and $\mathbf{1}$ is an $n \times 1$ vector of ones. The posterior covariance between the responses at the two inputs $\boldsymbol{x}^*$ and $\boldsymbol{x}'$ reads:

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

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

If the training dataset has multiple outputs, one may fit either a single-response GP emulator to each response or a multi-response GP (hereafter denoted by MRGP) to all the responses. We follow [40] and extend the above formulations to simulators with dy responses by placing a constant mean for each response (i.e., $\boldsymbol{\beta} = [\beta_{(1)}, \dots, \beta_{(q)}]^T$) and employing the separable covariance function:

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

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 multi-response datasets in which case σ will be replaced with $\boldsymbol{\Sigma}$ (see [41-44] for the 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 [31].

3 GLOBALLY APPROXIMATE GAUSSIAN PROCESS

Regardless of the optimization method used to solve for $\widehat{\boldsymbol{\omega}}$, 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 an ensemble of independent GPs that use the same $\widehat{\boldsymbol{\omega}}$ and share the training data among themselves. To illustrate, we take the following function over $-2 \leq x \leq 3$:

$$y = x^4 - x^3 - 7x^2 + 3x + 5\sin(5x)\,. \qquad (11)$$

The associated likelihood profile (i.e., L) is visualized in **Figure 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 generated five independent training samples via Sobol sequence [45, 46] and plotted the corresponding L. As illustrated, the five curves with the same n are indistinguishable (i.e., 20 curves are plotted in **Figure 1**). (ii) As n increases, L is minimized at similar ω's.

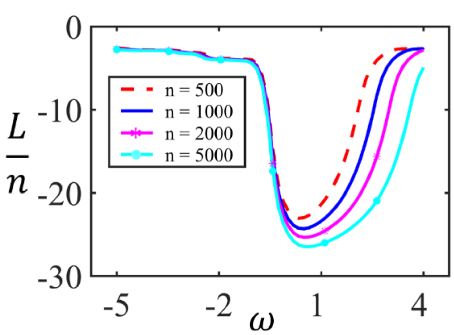

Figure 1 The profile of $\frac{L}{n}$ as a function of ω for various values of n. For each n, five curves are plotted, which are indistinguishable.

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) 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 $\widehat{\boldsymbol{\omega}}_0$ with a gradient-based optimization technique. Then, we add n_s random samples (e.g., $n_s = 250$ or $n_s = 500$) to this subset and estimate $\widehat{\boldsymbol{\omega}}_1$ while employing $\widehat{\boldsymbol{\omega}}_0$ as the initial guess in the optimization. This process is stopped after s steps when $\widehat{\boldsymbol{\omega}}$ does not change noticeably (i.e., $\widehat{\boldsymbol{\omega}}_s \cong \widehat{\boldsymbol{\omega}}_{s-1}$) as more training data are used. At this point, the latest solution, denoted by $\widehat{\boldsymbol{\omega}}_\infty$, is employed to build m GP models each with $n_k \geq n_0 + s \times n_s$ randomly chosen samples (from the entire training data) where $n = \sum_{k=1}^{m} n_k$. Here, we have assumed that the collection of these GPs (who have $\widehat{\boldsymbol{\omega}}_\infty$ as their hyperparameters) approximate a GP that is fitted to the entire training dataset and, correspondingly, call it globally approximate Gaussian process (GAGP). The algorithm of GAGP is summarized in **Figure 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., $\widehat{\boldsymbol{\omega}}_0$; and (ii) we want to use $\widehat{\boldsymbol{\omega}}_{i-1}$ as the initial guess for the optimization in the i^{th} step. This latter choice ensures fast convergence since the minimizer of L changes slightly as the dataset size increases (see **Figure 1**). To estimate $\widehat{\boldsymbol{\omega}}_0$, we recommend the method developed in [31]. 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 have observed very similar prediction results with different averaging schemes (e.g., weighted averaging where the weights are proportional to inverse variance). The advantages of employing an ensemble of models (in our case the m GPs) in prediction is 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 enough values are used for them. For novice users, we recommend starting with $n_0 = 500$, $s = 6$, $n_s = 250$, and equally distributing the samples among the m 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 by the nugget variance. Lastly, it is pointed out that while GAGP has a high predictive power and is applicable to very large datasets, its implementation is very straightforward because it only entails integrating a GP modeling package such as GPM [31] with the algorithm in **Figure 2**.

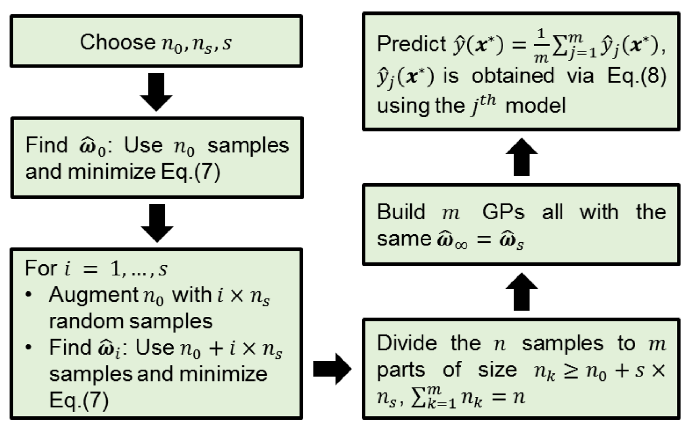

Figure 2 Flowchart of globally approximate Gaussian process, GAGP.

4 COMPARATIVE STUDIES ON ANALYTICAL EXAMPLES

To validate the performance of GAGP in regression, we compare its predictive power on four examples (Ex1-4) against recognized big data learners: locally approximate Gaussian process (LAGP) [28], gradient boosted trees (XGB) [14], and feed-forward neural networks (NNs) [47]. The examples cover a wide range of dimensionality and input-output complexity. Their analytical equations are as follows:

Ex1: $y = x^4 - x^3 - 7x^2 + 3x + 5\sin(5x)$
$-2 \le x \le 3$,

Ex2 [48]: $y = \frac{\left(1-\exp\left(-\frac{1}{2x_2}\right)\right)(x_3x_1^3+1900x_1^2+2092x_1+60)}{x_4x_1^3+500x_1^2+4x_1+20}$
$\min(\boldsymbol{x}) = [0,\ 0,\ 2200,\ 85]$
$\max(\boldsymbol{x}) = [1,\ 1,\ 2400,\ 110]$,

Ex3 [49]: $y = 2\pi \sqrt{\frac{x_1}{x_4 + \frac{4x_3x_4x_5x_6}{\left(x_5x_2+19.62x_1-\frac{x_4x_3}{x_2}\right)^2+4x_4x_5\left(\frac{x_6}{x_7}\right)x_7}}}$
$\min(\boldsymbol{x}) = [30,\ 0.005,\ 0.002,\ 1000, 9\times10^4, 290, 340]$
$\max(\boldsymbol{x}) = [60,\ 0.02,\ 0.01,\ 5000, 11\times10^4, 296, 360]$,

Ex4 [49]: $y = \frac{5x_{12}}{1+x_1} + 5(x_4 - x_{20})^2 + x_5 + 40x_{19}^3 - 5x_1 + 0.05x_2 + 0.08x_3 - 0.03x_6 + 0.03x_7 - 0.09x_9 - 0.01x_{10} - 0.07x_{11} + 0.25x_{13}^2 - 0.04x_{14} + 0.06x_{15} - 0.01x_{17} - 0.03x_{18}$
$0.5 \le x_i \le 0.5$ for $i = 1, \dots 20$,

For each example, two independent and unique datasets of size 30000 are generated with Sobol sequence [46], where the first one is used for training while the second one 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. 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).

Table 1 Root mean squared error (RMSE) with small noise. Smallest errors are in bold.

	Noise SD	LAGP	XGB	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 (6D)	0.1	0.129	0.118	**0.100**	**0.100**
Ex4 (20D)	0.1	1.450	0.351	**0.101**	0.103

Table 2 Root mean squared error (RMSE) with large noise. Smallest errors are in bold.

	Noise SD	LAGP	XGB	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 (6D)	1	1.037	1.098	**1.002**	**1.002**
Ex4 (20D)	1	1.911	1.155	1.011	**1.001**

We use CV to ensure the best performance is achieved for LAGP, XGB, and NN. For GAGP, we use $n_0 = 500$, $s = 6$, $n_s = 250$, and equally distribute the samples among the $m = \frac{30000}{500+6\times250} = 15$ 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 XGB, (ii) GAGP and NN both recover the true amount of added noise in the data, and (iii) GAGP achieves very similar results to NN. 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, XGB, and NN). Potential parameters include n_0, s, n_s, and $f_i(\boldsymbol{x})$ (see Eq. (1)). 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 process. Contrary to many supervised learning techniques such as NNs and random forests that are black-boxes, the structure of GPs can easily provide quantitative insights into the problem. To demonstrate this feature in GAGP, the convergence histories for Ex3 and Ex4 are plotted in **Figure 3** and **Figure 4**, respectively. Similar to **Figure 1**, it is evident that the estimated roughness parameters do not change noticeably as more samples are used in training (only 6 out of the 20 roughness parameters are plotted in **Figure 4** for a clearer illustration). The values of these parameters can determine which inputs (and to what extent) affect the output. For instance, in Ex4, ω_8 is very small so the output must be almost insensitive to x_8. Additionally, since $\omega_4 \cong \omega_{20}$, it is also expected that the corresponding inputs should affect y similarly. These observations completely agree with the analytical relation between $\boldsymbol{x}$ and y in Ex4 where y is independent of x_8 and is symmetric with respect to x_4 and x_{20}.

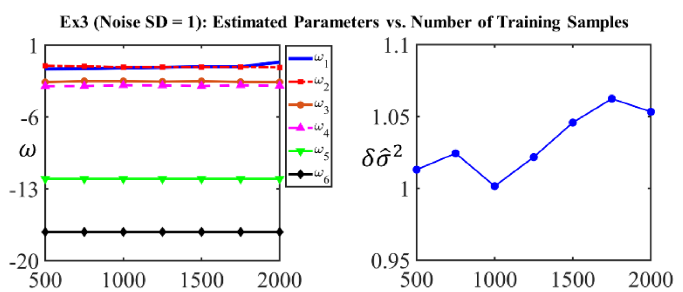

Figure 3 Convergence history in example 3 as the number of training samples is increased from 500 to 2000.

The estimated variance, $\delta\hat{\sigma}^2$, in both examples fluctuates very closely around the true noise variance. $\delta\hat{\sigma}^2$ provides a useful quantitative measure for the expected predictive power (e.g., RMSE in future uses of the model). Additionally, similar to $\hat{\boldsymbol{\omega}}$, its convergence history helps in determining whether sufficient samples have been used in training. Firstly, the number of training samples should be increased until $\delta\hat{\sigma}^2$ does not fluctuate noticeably. Secondly, via k-fold CV during training, one should ideally recover the noise variance by calculating the RMSE associated with predicting the samples in the i^{th} 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 **Figure 4** were large, we should have increased s (from 6 to, e.g., 10) or n_s (from 250 to, e.g., 500).

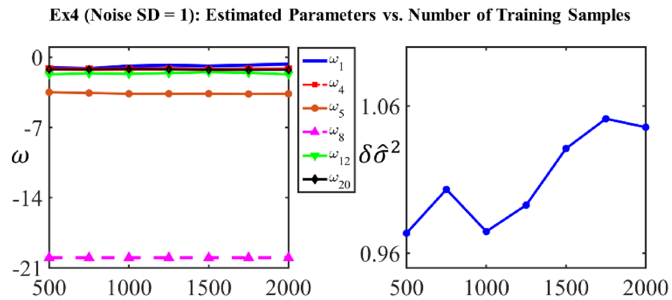

Figure 4 Convergence history in example 4 as the number of training samples is increased from 500 to 2000. For clearer demonstration, only six out of the twenty roughness parameters are plotted.

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 genetic algorithm (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 if multiple unit cells exist. A promising solution is to construct a large database of precomputed unit cells (aka microstructures or building blocks), enabling efficient selection of well-connected unit cells from the database and inexpensive optimization of new unit cells [9-12]. However, with the exception of [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. Reducing the dimension of the unit cell is therefore 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 Young's moduli of unit cells change as a function of their reduced-dimension 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 to those obtained using a conventional GP model fitted on a smaller database.

5.1 Metamaterials Database Generation

For demonstration, we focus on cases where the Young's moduli are the primary properties of interest, i.e., E_x , E_y, and E_{xy}. We propose a novel two-stage pipeline method inspired by [11] to generate a large training dataset of unit cells and their corresponding moduli. As explained below, our method starts by building an initial dataset and then proceeds to better cover the input (geometry) and output (property) spaces.

In stage one, to construct the initial dataset, we select some design targets in the property space (the 3D space spanned by E_x , E_y , and E_{xy}) and then use the SIMP (solid isotropic material with penalization) method [50] to find the corresponding orthotropic unit cells through TO. As the bounds of the property space are unknown *a priori*, we uniformly choose 1000 points in $[0,1]^3$. In this stage, 358 valid structures are generated while the remaining 642 points did not result in feasible unit cells. These initial structures are shown in **Figure 5**, where the Poisson's ratio is used instead of E_{xy} for a better illustration of the space.

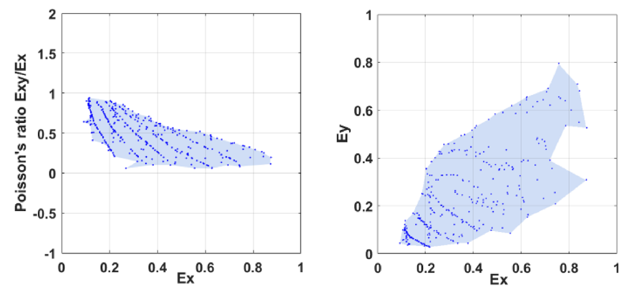

Figure 5 The property space of the initial database with 358 structures.

In stage two, we further populate the initial database via a stochastic shape perturbation algorithm that generates distorted structures with slightly different properties from the original ones. Specifically, the following radial distortion model is used to perturb an existing shape:

$$x_{new} = \begin{cases} x_c + \frac{r_{new}}{r_{old}}(x_{old} - x_c) & if \quad r_{old} \le R_0 \\ x_{old} & if \quad r_{old} > R_0 \end{cases}, \tag{15}$$

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, and R_0 is the outer distortion radius. r_{new} can be expressed as:

$$r_{new} = \begin{cases} \frac{1}{2} R_0 \left(1 - cot\left(\frac{\gamma}{2}\right) - \beta\right) & if \quad \gamma > 0 \\ \frac{1}{2} R_0 \left(1 - cot\left(\frac{\gamma}{2}\right) + \beta\right) & if \quad \gamma < 0 \\ r_{old} & otherwise \end{cases}, \tag{16}$$

where

$$\beta = \sqrt{\frac{2}{sin^2(\gamma/_2)} - \left(1 + cot\left(\frac{\gamma}{2}\right) - \frac{2r_{old}}{R_0}\right)^2}, \qquad (17)$$

and $\gamma \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ is the angle that controls the amount of distortion. Considering the orthorhombic 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. (15) for two reasons. First, its parameters (i.e., 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 the available unit cells:

$$Score = \frac{1}{(d + \varepsilon)2^{\rho}}, \qquad (18)$$

where d is the Euclidean (L2) distance between the moduli of each unit cell to the boundaries of the region enclosing all the moduli (see **Figure 5** and **Figure 6**), 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 $\rho = 0.05$), and $\varepsilon \ll 1$ is used to avoid singularity. Then, we select the N points with the highest scores for stochastic perturbation. The properties of newly generated structures are calculated via numerical homogenization [51] and added to the current property space for the next iteration. With our two-stage method, the database is expanded from 358 to 88,000 unit cells that cover a wider range of properties (see **Figure 6**).

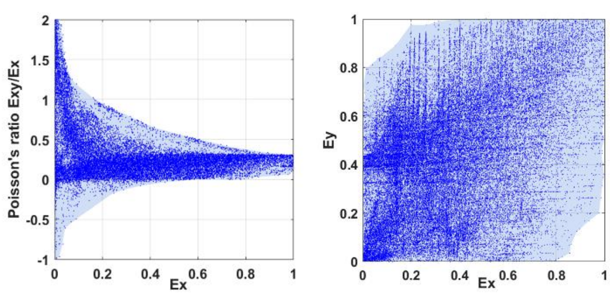

Figure 6 The property space of the expanded database with 88,000 structures.

5.2 Unit Cell Dimension Reduction via Spectral Shape Descriptors

In the previous section, each unit cell in the database is represented by voxels (or pixels). For dimensionality reduction, we use spectral shape descriptors as they retain geometric and physical information. Specifically, we use the LB spectrum, also known as ShapeDNA, which can be directly calculated for any unit cell shape [52, 53].

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 to 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 [52], the Helmholtz equation reads as:

$$\Delta f = -\lambda f, \qquad (19)$$

where the Laplace-Beltrami operator Δ is defined as:

$$\Delta := div(grad\ \ f). \qquad (20)$$

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

$$0 \leq \lambda_1 \leq \lambda_2 \leq \cdots < \infty. \qquad (21)$$

We focus on the LB spectrum of a 2D shape under Dirichlet boundary conditions. In this case, the Helmholtz equation reduces to a Laplacian eigenvalue problem with the Dirichlet boundary condition:

$$\begin{aligned} \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} &= -\lambda f \qquad in\ \ \Omega \\ f &= 0 \qquad on\ \ \tau \end{aligned}, \qquad (22)$$

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

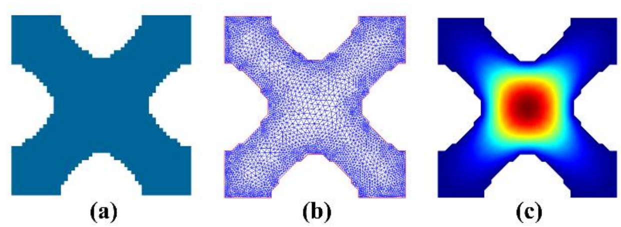

Figure 7 LB spectrum calculation: **(a)** Original structure, **(b)** Finite element mesh, and **(c)** The first eigenfunction.

Finally, the finite element method is employed to obtain the LB spectrum of unit cells [54]; see **Figure 7**. It is noted that our 88,000 structures can be uniquely determined with only the first 16 orders of LB spectrum, reducing the input dimension from $50 \times 50 = 2500$ pixels to 16 scalar descriptors.

5.3 Machine Learning - Linking LB Representation to Property via GAGP

Once the dataset is built, we follow the algorithm in **Figure 2** for machine learning, i.e., relating the LB representations of unit cells to their moduli. We use the same fitting parameters as in Sec. 4 ($n_0 = 500,\ s = 6,\ n_s = 250$), equally distribute the samples among the $m = \frac{88000}{500+6\times250} = 44$ GPs, and use Eq. (10) to have a multi-response model that leverages the correlation between the responses to have a higher predictive power. The convergence histories are provided in **Figure 8**, where the trends are consistent with those in Sec. 4. It is observed that the 16 estimated roughness parameters do not change noticeably once more than 1000 samples are used in training. In particular, 3 out of the 16 roughness estimates (corresponding to $\lambda_{14} = \lambda_{15} = \lambda_{16}$) are very small, indicating that the corresponding shape descriptors do not affect the responses; see Eq. (3). 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, indicating that the first twelve 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.

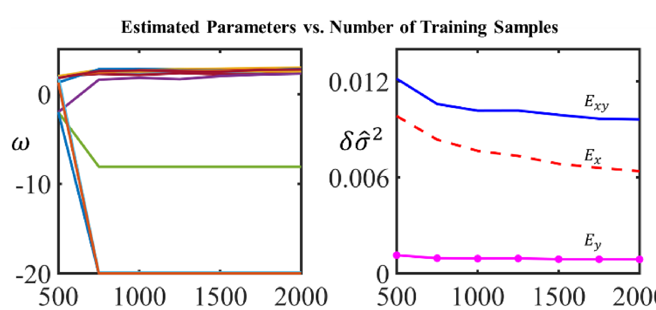

Figure 8 Convergence history as the number of training samples is increased from 500 to 2000. The 16 colored lines in the left panel indicate the histories of the 16 hyperparameters.

To illustrate the effect of expanding the training data from 385 to 88000, we randomly select 28000 samples from the data 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 original 385 samples and a GAGP fitted to the rest of the data (i.e., to 60000 samples, resulting in $m = \frac{60000}{500+6\times250} = 30$ models). To account for randomness, we repeat this process 20 times. The results are summarized in **Table 3** 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 60000 samples, 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 3** with the converged noise estimates in **Figure 8**).

Table 3 MSE errors on 28000 random samples. The mean and variance of MSE are calculated over 20 random repetitions.

	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

5.4 Data-Driven Unit Cell Optimization

In this section, we demonstrate how our GAGP model can be employed in an inverse optimization scheme to realize unit cells with target moduli. Establishing such an inverse link is highly desirable in structure design as it allows to achieve desired elastic properties efficiently, obviating the tedious and expensive trial and error in TO. Additionally, though not demonstrated in this work, such a link can provide multiple candidate unit cells with the same properties that, in turn, enables tiling different unit cells into a macrostructure while ensuring boundary compatibility.

Our data-driven optimization scheme has two stages: The search for the optimal LB spectrum and the reconstruction of the unit cell given the LB spectrum. Using our GAGP (or GP) model, we directly search for the LB spectrum of the unit cell with the desired properties. We employ GA in this search process, which is formulated as:

$$\begin{gathered}\min_{\boldsymbol{\lambda}} \|\boldsymbol{E}^t - \boldsymbol{E}^p\|_\infty \\ s.t.\quad \lambda_{i-1} \le \lambda_i \\ 0.9\lambda_i^0 \le \lambda_i \le 1.1\lambda_i^0,\end{gathered} \tag{23}$$

where $\boldsymbol{E}^t$ and $\boldsymbol{E}^p$ are the vectors of, respectively, the target and predicted moduli, and $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_{16}]$ and $\boldsymbol{\lambda}^{\mathbf{0}}$ are the LB spectra of the current unit cell and the unit cell closest to the prescribed properties in the property space respectively, with λ_i being the i$^{\text{th}}$ order eigenvalue. We choose GA for optimization since the GAGP (or GP) model is cheap to run and GA ensures global optimality for multivariate and constrained problems. The search space for GA is defined by the LB spectrum of the unit cell in the training dataset whose properties are closest $\boldsymbol{E}^t$.

After obtaining the optimal LB spectrum, we use the level set method to reconstruct the corresponding unit cell. The reconstruction process is based on [55] but the optimization objective is changed to the squared residuals of the LB spectrum. For a 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 objective is to design structures with desired E_x, E_y, and E_{xy} (see targets in **Figure**

9). In each example, two unit cells are designed: one with the GP model (fitted to the initial set of 358) and one with the GAGP model (fitted to all 88,000 structures). The results are visualized in **Figure 9** and demonstrate that the unit cells identified from GAGP are more diverse than those obtained via GP. This diversity is a direct result of populating the original dataset with perturbed structures and, in turn, providing the GA search process with a wider range of initial seeds. It is also noted that the unit cells reconstructed with GP are similar in shape but different in the size of the center hole, which leads to a significant change in properties.

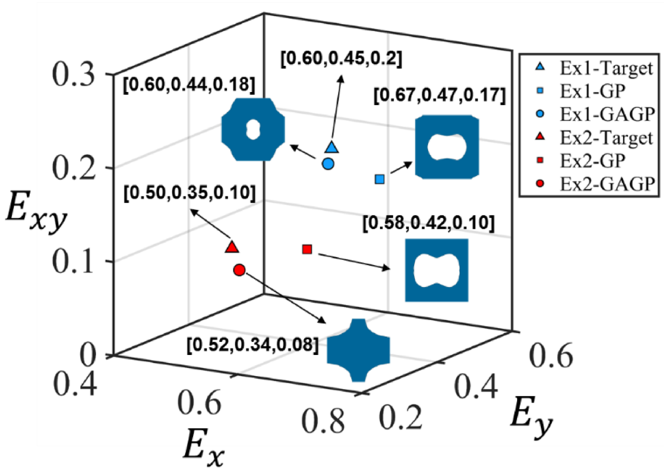

Figure 9 Reconstructed unit cells in the two examples. The results are visualized in the property space, $[E_x, E_y, E_{xy}]$.

From a quantitative point of view, it is observed that our data-driven design method with the large database can, as compared to the small dataset case, discover unit cells with properties that are closer to the target values. For instance, in Ex1, the GAGP result with large data set achieves the target E_x, whereas the GP result from the small dataset differs from the target by around 12%. Ex2 shows a similar pattern, with the GAGP and GP results differing from the target E_x by 4% 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.

6 CONCLUSION AND FUTURE WORKS

We propose a novel method to enable Gaussian process modeling of massive datasets. The central idea of our method, named globally approximate Gaussian process (GAGP), is based on the observation that the hyperparameter estimates of a GP converge to some limit values, $\hat{\boldsymbol{\omega}}_\infty$, as more training samples are used. We introduce an intuitive and straightforward method to find $\hat{\boldsymbol{\omega}}_\infty$ and, subsequently, build an ensemble 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 to make inference based on the entire dataset by pooling the predictions from the individual GPs.

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 neural networks 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. In our approach, we assume that the noise is stationary 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 achieves desired elastic properties by transforming the complex material design problem into a parametric one. 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.

7 ACKNOWLEDGMENTS

The authors are grateful to Prof. K. Svanberg, from Royal Institute of Technology, Sweden, for providing a copy of the MMA code for metamaterial property calculations. Support from NSF grants (ACI 1640840 and OAC 1835782) and AFOSR FA9550-18-1-0381 are greatly appreciated. Ms. Yu-Chin Chan would like to acknowledge the NSF Graduate Research Fellowship Program under grant DGE-1842165.

8 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," Computational Mechanics, 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," Manufacturing letters, 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," J Mech Design, 140(11), p. 111408.
[4] Bessa, M. A., Bostanabad, R., Liu, Z., Hu, A., Apley, D. W., Brinson, C., Chen, W., and Liu, Wing 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," Journal of microscopy, 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., August 26–29, 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, American Society of Mechanical Engineers, Quebec City, Quebec, Canada.
[8] Bostanabad, R., Bui, A. T., Xie, W., Apley, D. W., and Chen, W., 2016, "Stochastic microstructure characterization and reconstruction via supervised learning," Acta Materialia, 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 Transactions on Graphics, 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 Transactions on Graphics, 34(4), p. 135.
[11] Zhu, B., Skouras, M., Chen, D., and Matusik, W., 2017, "Two-Scale Topology Optimization with Microstructures," ACM Transactions on Graphics, 36(5), p. 164.
[12] Chen, D., Skouras, M., Zhu, B., and Matusik, W., 2018, "Computational discovery of extremal microstructure families," Science Advances, 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," Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 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," Scientific Reports, 7(1), p. 15259.
[16] Tao, S., Shintani, K., Bostanabad, R., Chan, Y.-C., Yang, G., Meingast, H., and Chen, W., "Enhanced Gaussian Process Metamodeling and Collaborative Optimization for Vehicle Suspension Design Optimization," Proc. ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers.
[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," Composites Science and Technology, 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," Journal of the American Statistical Association, 103(483), pp. 1119-1130.
[20] Kim, H.-M., Mallick, B. K., and Holmes, C., 2005, "Analyzing nonstationary spatial data using piecewise Gaussian processes," Journal of the American Statistical Association, 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 computation, 12(11), pp. 2719-2741.
[23] Herbrich, R., Lawrence, N. D., and Seeger, M., "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., "Fast forward selection to speed up sparse Gaussian process regression," Proc. Artificial Intelligence and Statistics 9.
[25] Williams, C. K., and Seeger, M., "Using the Nyström method to speed up kernel machines," Proc. Advances in neural information processing systems, pp. 682-688.
[26] Rasmussen, C. E., and Quinonero-Candela, J., "Healing the relevance vector machine through augmentation," Proc. 22nd international conference on Machine learning, ACM, pp. 689-696.
[27] Wahba, G., 1990, Spline models for observational data, 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," Journal of Manufacturing Science and Engineering, 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," Computational Mechanics.
[31] 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.
[32] MacDonald, B., Ranjan, P., and Chipman, H., 2015, "GPfit: AnRPackage for Fitting a Gaussian Process Model to Deterministic Simulator Outputs," Journal of Statistical Software, 64(12).
[33] Plumlee, M., and Apley, D. W., 2017, "Lifted Brownian Kriging Models," Technometrics, 59(2), pp. 165-177.
[34] 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.
[35] Sacks, J., Schiller, S. B., and Welch, W. J., 1989, "Designs for Computer Experiments," Technometrics, 31(1), pp. 41-47.
[36] Toal, D. J. J., Bressloff, N. W., and Keane, A. J., 2008, "Kriging hyperparameter tuning strategies," Aiaa Journal, 46(5), pp. 1240-1252.
[37] Audet, C., and Dennis Jr, J. E., 2002, "Analysis of generalized pattern searches," SIAM Journal on optimization, 13(3), pp. 889-903.
[38] Zhao, L., Choi, K., and Lee, I., 2011, "Metamodeling method using dynamic kriging for design optimization," AIAA journal, 49(9), pp. 2034-2046.
[39] Toal, D. J., Bressloff, N., Keane, A., and Holden, C., 2011, "The development of a hybridized particle swarm for kriging hyperparameter tuning," Engineering optimization, 43(6), pp. 675-699.
[40] 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.
[41] Arendt, P. D., Apley, D. W., and Chen, W., 2012, "Quantification of Model Uncertainty: Calibration, Model Discrepancy, and Identifiability," Journal of Mechanical Design, 134(10), p. 100908.
[42] Arendt, P. D., Apley, D. W., Chen, W., Lamb, D., and Gorsich, D., 2012, "Improving identifiability in model calibration using multiple responses," J Mech Design, 134(10), p. 100909.
[43] 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," The Annals of Statistics, pp. 1874-1906.
[44] Conti, S., and O'Hagan, A., 2010, "Bayesian emulation of complex multi-output and dynamic computer models," Journal of statistical planning and inference, 140(3), pp. 640-651.
[45] Sobol, I. M., 1967, "On the distribution of points in a cube and the approximate evaluation of integrals," Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 7(4), pp. 784-802.
[46] Sobol, I. M., 1998, "On quasi-Monte Carlo integrations," Math Comput Simulat, 47(2-5), pp. 103-112.
[47] Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., and Tibshirani, R., 2009, The elements of statistical learning, Springer.
[48] Bastos, L. S., and O'Hagan, A., 2009, "Diagnostics for Gaussian process emulators," Technometrics, 51(4), pp. 425-438.
[49] 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," Quality Engineering, 19(4), pp. 327-338.
[50] Xia, L., and Breitkopf, P., 2015, "Design of materials using topology optimization and energy-based homogenization approach in Matlab," Structural and Multidisciplinary Optimization, 52(6), pp. 1229-1241.
[51] Andreassen, E., and Andreasen, C. S., 2014, "How to determine composite material properties using numerical homogenization," Comp Mater Sci, 83, pp. 488-495.
[52] Reuter, M., Wolter, F.-E., and Peinecke, N., 2006, "Laplace–Beltrami spectra as 'Shape-DNA' of surfaces and solids," Computer-Aided Design, 38, pp. 342-366.
[53] 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 Recognition, 46, pp. 449-461.
[54] Su, S., 2010, "Numerical approaches on shape optimization of elliptic eigenvalue problems and shape study of human brains," The Ohio State University.
[55] Zhu, S., 2018, "Effective shape optimization of Laplace eigenvalue problems using domain expressions of Eulerian derivatives," Journal of Optimization Theory and Applications, 176(1), pp. 17-34.

