GAUSSIAN PROCESS EMULATION FOR BIG DATA IN 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†  
Northwestern University  
Evanston, IL, USA

ABSTRACT
Our main contribution is to introduce a novel method for Gaussian process (GP) modeling of massive datasets. The key idea is to build an ensemble of independent GPs that use the same hyperparameters but distribute the entire training dataset among themselves. This is motivated by our observation that estimates of the GP hyperparameters change negligibly as the size of the training data exceeds a certain level, which can be found in a systematic way. For inference, the predictions from all GPs in the ensemble are pooled to efficiently exploit the entire training dataset for prediction. We name our modeling approach globallly approximate Gaussian process (GAGP), which, unlike most largescale supervised learners such as neural networks and trees, is easy to fit and can interpret the model behavior. These features make it particularly useful in engineering design with big data. We use analytical examples to demonstrate that GAGP achieves very high predictive power that matches or 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 accomplished by employing GAGP in inverse optimization.

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

† Corresponding Author. Wilson Cook Professor in Engineering Design. Email: weichen@northwestern.edu

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 [13] (NNs) and gradient boosted trees [14] (GBT) are difficult to interpret and hence less 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 smooth the data to address noise (i.e., 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, \( R \), 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 (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 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 \( R \) 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 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 an ensemble of independent GPs that utilize converged roughness parameters as their hyperparameters. This is based on an empirical observation that the estimates of the GP hyperparameters negligibly change 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 \( \mathbf{x} = [x_{1(1)}, x_{2(1)}, \ldots, x_{d(1)}]^T \) where \( \mathbf{x} \in \mathbb{R}^d \). Assume the input-output relation is a realization of the random process \( \eta(\mathbf{x}) \):

\[
\eta(\mathbf{x}) = \sum_{i=1}^{n} \beta_i f_i(\mathbf{x}) + \xi(\mathbf{x}),
\]

where \( f_i(\mathbf{x}) \)'s are some pre-determined set of basis functions, \( \beta = [\beta_1, \ldots, \beta_n]^T \) are unknown weights, and \( \xi(\mathbf{x}) \) is a zero-mean GP characterized with its parametric covariance function, \( c(\cdot, \cdot) \):

\[
cov(\xi(\mathbf{x}), \xi(\mathbf{x}')) = c(\mathbf{x}, \mathbf{x}') = \sigma^2 r(\mathbf{x}, \mathbf{x}'),
\]

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

\[
r(\mathbf{x}, \mathbf{x}') = \exp(-((\mathbf{x} - \mathbf{x}')^T \Omega (\mathbf{x} - \mathbf{x}'))),
\]

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

With the formulation in Eq. (1) and given the \( n \) training pairs of \((\mathbf{x}_i, \mathbf{y}_i)\), GP modeling requires finding a point estimate for \( \beta \), \( \omega \), and \( \sigma^2 \) 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., \( \sum_{i=1}^{n} \beta_i f_i(\mathbf{x}) = \beta \)) 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:

\[
[\hat{\beta}, \hat{\sigma}^2, \hat{\omega}] = \arg\min_{\beta, \sigma^2, \omega} \left\{ \frac{n}{2} \log(\sigma^2) + \frac{1}{2} \log(|\mathbf{R}|) + \frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{1}\beta)^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{1}\beta) \right\},
\]

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

\[
\hat{\beta} = \left[ \mathbf{1}^T \mathbf{R}^{-1} \mathbf{1} \right]^{-1} \mathbf{1}^T \mathbf{R}^{-1} \mathbf{y},
\]

\[
\hat{\sigma}^2 = \frac{1}{n} (\mathbf{y} - \mathbf{1}\hat{\beta})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{1}\hat{\beta}).
\]

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

\[
\hat{\omega} = \arg\min_{\omega} n \log(\hat{\sigma}^2) + \log(|\mathbf{R}|) = \arg\min_{\omega} L.
\]

By numerically minimizing \( L \), one can find \( \hat{\omega} \). Many global optimization methods such as genetic algorithms [36], pattern searches [37, 38], and particle swarm optimization [39] have been employed to solve for \( \hat{\omega} \) 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 \( \mathbf{x}' \):

\[
\hat{y}(\mathbf{x}') = \hat{\beta} + \mathbf{g}^T(\mathbf{x}') \mathbf{V}^{-1} (\mathbf{y} - \mathbf{1}\hat{\beta}),
\]

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

\[
cov(y^*, y') = c(x', x'') - \mathbf{g}^T(x') \mathbf{V}^{-1} \mathbf{g}(x') + \mathbf{h}^T (\mathbf{1}^T \mathbf{V}^{-1} \mathbf{1})^{-1} \mathbf{h},
\]

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

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 \( q \) responses by placing a constant mean for each response (i.e., \( \beta = [\beta_{1(1)}, \ldots, \beta_{q(1)}]^T \)) and employing the separable covariance function:

\[
cov(\xi(\mathbf{x}), \xi(\mathbf{x}')) = c(x', x'') = \Sigma \otimes r(x', x''),
\]

where \( \otimes \) denotes the Kronecker product and \( \Sigma \) is the \( q \times q \) process covariance matrix with its off-diagonal elements representing the covariance between the corresponding responses at any fixed \( \mathbf{x} \). The MLE approach described above can also be applied to multi-response datasets in which case \( \sigma \) will be replaced with \( \Sigma \) (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, \( \delta \), in which case \( \mathbf{R} \) is replaced with \( \mathbf{R}_\delta = \mathbf{R} + \delta \mathbf{I}_{n,n} \). If \( \delta \) 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 \( \hat{\omega} \), each evaluation of \( L \) in Eq. (7) requires inverting the \( n \times n \)
matrix $R$. For very large $n$, there are two main challenges associated with this inversion: computational cost of approximately $O(n^3)$ and singularity of $R$ (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 $\tilde{\omega}$ and share the training data among themselves. To illustrate, we take the following function over $-2 \leq x \leq 3$:

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

The associated likelihood profile (i.e., $L$) is visualized in Figure 1 as a function of $\omega$ 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 [45, 46] and plot the corresponding $L$. As illustrated in Figure 1, even though a total of 20 curves are plotted, only four are visible since the curves with the same $n$ are indistinguishable. (ii) As $n$ increases, $L$ is minimized at similar $\omega$’s.

![Figure 1](image1.png)

Figure 1 The profile of $\frac{L}{n}$ as a function of $\omega$ 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) 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 $\tilde{\omega}_0$ 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 $\tilde{\omega}_1$, while employing $\tilde{\omega}_0$ as the initial guess in the optimization. This process is stopped after $s$ steps when $\tilde{\omega}$ does not change noticeably (i.e., $\tilde{\omega}_s \approx \tilde{\omega}_{s-1}$) as more training data are used. At this point, the latest solution, denoted by $\tilde{\omega}_s$, 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 $\tilde{\omega}_0$ 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., $\tilde{\omega}_0$; and (ii) we want to use $\tilde{\omega}_{i-1}$ 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 $\tilde{\omega}_0$, 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.

![Flowchart](image2.png)

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

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]. As shown in Eqs.
\( y = x^4 - x^3 - 7x^2 + 3x + 5\sin(5x), \quad -2 \leq x \leq 3, \) \hspace{1cm} (12)

\[ y = \frac{1 - \exp(-x^2/2)}{2\sqrt{\pi}}x^3 + 1000x^2 + 2092x + 60, \]
\[ x_{4x^4 + 500x^2 + 4x+20}, \]
\[ \min(x) = [0, 0, 2200, 85], \]
\[ \max(x) = [1, 1, 2400, 110], \]

\[ y = 2\pi \sqrt{x_4 + \frac{x_1}{4x^3 + x_5 + x_6}}, \quad \] (14)
\[ \min(x) = [30, 0.005, 0.002, 1000, 9 \times 10^4, 290, 340], \]
\[ \max(x) = [60, 0.02, 0.01, 5000, 11 \times 10^4, 296, 360], \]

\[ y = \frac{5x_1^2}{1 + x_1} + 5(x_4 - x_2)^2 + x_5 + 40x_3^9 - 5x_1 + 0.05x_2 + 0.08x_3 - 0.03x_6 + 0.03x_7 - 0.09x_9 - 0.01x_{10} - 0.07x_{11} + 0.25x_{13}^2 - 0.04x_{14} + 0.06x_{15} - 0.01x_{17} - 0.03x_{18}, \quad \] (15)
\[ -0.5 \leq x_i \leq 0.5 \quad \text{for } i = 1, \ldots, 20. \]

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

<table>
<thead>
<tr>
<th>Noise SD</th>
<th>LAGP</th>
<th>XGB</th>
<th>NN</th>
<th>GAGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex1 (1D)</td>
<td>0.2</td>
<td>1.271</td>
<td>0.209</td>
<td>\textbf{0.200}</td>
</tr>
<tr>
<td>Ex2 (4D)</td>
<td>0.1</td>
<td>1.386</td>
<td>0.121</td>
<td>\textbf{0.100}</td>
</tr>
<tr>
<td>Ex3 (7D)</td>
<td>0.1</td>
<td>0.129</td>
<td>0.118</td>
<td>\textbf{0.100}</td>
</tr>
<tr>
<td>Ex4 (20D)</td>
<td>0.1</td>
<td>1.450</td>
<td>0.351</td>
<td>\textbf{0.101}</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Noise SD</th>
<th>LAGP</th>
<th>XGB</th>
<th>NN</th>
<th>GAGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex1 (1D)</td>
<td>2</td>
<td>2.270</td>
<td>2.062</td>
<td>\textbf{2.000}</td>
</tr>
<tr>
<td>Ex2 (4D)</td>
<td>1</td>
<td>1.739</td>
<td>1.123</td>
<td>\textbf{1.002}</td>
</tr>
<tr>
<td>Ex3 (7D)</td>
<td>1</td>
<td>1.037</td>
<td>1.098</td>
<td>\textbf{1.002}</td>
</tr>
<tr>
<td>Ex4 (20D)</td>
<td>1</td>
<td>1.911</td>
<td>1.155</td>
<td>\textbf{1.011}</td>
</tr>
</tbody>
</table>

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 provide qualitative insights. To demonstrate, we rewrite Eq. (3) as \( r(x, x') = \exp\left\{-\sum_{i=1}^{d} 10\omega_i(x_i - x_i')^2\right\}. \) If \( \omega_i \ll 0 \) (e.g., \( \omega_i = -10 \)), then variations along the \( i \)th dimension (i.e., \( x_i) \) do not contribute to the summation and, subsequently, to the correlation between \( x \) and \( x' \). This contribution increases as the magnitude of \( \omega_i \) increases. In a GP with constant mean of \( \beta \), all the effect of inputs on the output is captured through \( r(x, x') \). Hence, as \( \omega_i \) decreases, the effect of \( x_i \) on the output decreases as well. We illustrate this feature with a 2D example as follows. Assume \( f = y = f(x_1, x_2; \alpha) = \sin(2x_1x_2) + \alpha \cos(\alpha x_1^2), \quad -\pi \leq x_1, x_2 \leq \pi \) for \( \alpha = 2, 4, 6 \). Three points regarding \( f \) are highlighted:

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

Table 3. These indices are in \([0, 1]\) range with higher values indicating more sensitivity to the input. Here, SI of \( x_1 \) is always 1 but SI of \( x_2 \) decreases as \( \alpha \) increases. This trend indicates that the relative importance of \( x_1 \) on \( y \) increases as \( \alpha \) increases.

We now demonstrate that a GP can distill the above features from a training dataset. To this end, for each \( \alpha \), 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 $\alpha$, $\hat{\omega}_1$ is larger than $\hat{\omega}_2$ which implies that $x_1$ is more important than $x_2$.
- As $\alpha$ increases, $\hat{\omega}_1$ increases while $\hat{\omega}_2$ only changes insignificantly. This shows that, as $\alpha$ increases, $x_1$ becomes more important (since $\hat{\omega}_1$ increases), and that the underlying functional relation between $x_2$ and $y$ does not depend on $\alpha$ (since $\hat{\omega}_2$ does not change).
- For a given $\alpha$, the estimates change negligibly when $n$ is increased.

### Table 3 Effect of sample size and nonlinearity on hyperparameter estimates in the pedagogical 2D example. The Sobol’s total sensitivity indices (SIs) are also included.

<table>
<thead>
<tr>
<th></th>
<th>$n = 1000$</th>
<th>$n = 2000$</th>
<th>Total SI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\hat{\omega}_1$</td>
<td>$\hat{\omega}_2$</td>
<td>$\hat{\omega}_1$</td>
</tr>
<tr>
<td></td>
<td>2.39</td>
<td>-1.98</td>
<td>2.38</td>
</tr>
<tr>
<td></td>
<td>3.59</td>
<td>-2.11</td>
<td>3.54</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

To demonstrate the above 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, $\omega_8$ is very small so the output must be almost insensitive to $x_8$. Additionally, since $\omega_4 \equiv \omega_8$, it is also expected that the corresponding inputs should affect $y$ similarly. These observations completely agree with the analytical relation between $x$ 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 $\hat{\omega}$, 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 stiffness tensor of unit cells change 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 to those
obtained using a conventional GP model fitted on a smaller database.

5.1 Metamaterials Database Generation

We propose a novel two-stage pipeline method inspired by [11] to generate a large training dataset of unit cells and their corresponding homogenized properties. For demonstration, our primary properties of interest are the effective stiffness tensor components, $E_x$, $E_y$, and $E_{xy}$. 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 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 SIMP (Solid Isotropic Material with Penalization) TO method [50] to find the orthotropic unit cells corresponding to each target. This stage generates 358 valid structures, while the remaining 642 points do not result in feasible unit cells, mainly because the uniform sampling places some design targets in theoretically infeasible regions. Moreover, TO may fail to meet the design targets due to sensitivity to the initial shape which is difficult to guess without prior knowledge. These 358 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.

$$\text{Poisson's ratio} = \frac{v_{xy}}{v_{xx}}$$

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. This perturbation technique is performed iteratively to efficiently expand the property space by avoiding expensive TO in achieving prescribed properties. Specifically, the following radial distortion model is used to perturb an existing shape:

$$x_{\text{new}} = \begin{cases} x_c + \frac{r_{\text{new}}}{r_{\text{old}}} (x_{\text{old}} - x_c) & \text{if } r_{\text{old}} \leq R_0 \\ x_{\text{old}} & \text{if } r_{\text{old}} > R_0 \end{cases}$$

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

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

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. (16) for two reasons. First, its parameters (i.e., $R_0$, $\gamma$, 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:

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

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 Figure 5 and Figure 6), $\rho$ 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. After each iteration, newly generated unit cells are checked and discarded if they contain infeasible features such as isolated parts. The perturbation is repeated until the boundary of property space does not expand significantly and the points inside the boundaries are relatively dense. In this second stage, the database is expanded from 358 to 8800 unit cells that cover a wider range of properties (see Figure 6).

5.2 Unit Cell Dimension Reduction via Spectral Shape Descriptors

In the previous section, each unit cell in the database is represented by $50 \times 50$ 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 Shape-DNA, 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:

\[
\Delta f = -\lambda f, \tag{20}
\]

where the Laplace-Beltrami operator \( \Delta \) is defined as:

\[
\Delta : \text{div(grad } f). \tag{21}
\]

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

\[
0 \leq \lambda_1 \leq \lambda_2 \leq \ldots < \infty. \tag{22}
\]

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:

\[
\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = -\lambda f \quad \text{in } \Omega,
\]

\[
f = 0 \quad \text{on } \tau,
\]

where \( \Omega \) and \( \tau \) 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 \([54]\); see Figure 7. It is noted that our 88000 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. 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 @3.50 GHz). Since this computation is performed once and can be run in parallel, the runtime is small.

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

### 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 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 \( m = \frac{88000}{500+6\times250} = 44 \) 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 \( \omega_{13} \equiv -8 \) which corresponds to \( \lambda_{13} \). 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 \( \omega_k \) indicates rough response changes along dimension \( k \)). 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 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 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 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 4 with the converged noise estimates in Figure 8).

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.

Table 4 MSE errors on 28000 random samples. The mean and variance of MSE are calculated over 20 random repetitions.

<table>
<thead>
<tr>
<th></th>
<th>Mean of MSE</th>
<th>Variance of MSE (× 10^6)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E_x</td>
<td>E_y</td>
</tr>
<tr>
<td>GP</td>
<td>0.048</td>
<td>0.007</td>
</tr>
<tr>
<td>GAGP</td>
<td>0.008</td>
<td>0.001</td>
</tr>
</tbody>
</table>

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 stiffness tensor components. 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 steps: 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 use GA in this search process, which is formulated as:

\[
\min_{\lambda} \| E^\ell - E^p \|_\infty \\
\text{s.t. } \lambda_{i-1} \leq \lambda_i \\
0.9 \lambda_{i-1} \leq \lambda_i \leq 1.1 \lambda_{i-1},
\]

(24)

where \( E^\ell \) and \( E^p \) are the vectors of, respectively, the target and predicted stiffness tensors, and \( \lambda = [\lambda_1, ..., \lambda_{16}] \) and \( \lambda^p \) are the LB spectra of the current unit cell and the unit cell closest to the prescribed properties in the property space respectively, with \( \lambda_i \) being the \( i^{th} \) order eigenvalue. We choose GA for optimization since the GAGP 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 \( E^\ell \).

After obtaining the optimal LB spectrum, we use a level set method to reconstruct the corresponding unit cell based on [55], with the squared residuals of the LB spectrum as the optimization objective. 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 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 88000 structures). The results are visualized in Figure 9 and demonstrate that the unit cells identified from GAGP are more geometrically diverse than those obtained via GP. This is likely 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. It is also noted that the unit cells designed 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 using the large dataset 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, \( \tilde{\theta}_\infty \), as more training samples are used. We introduce an intuitive and straightforward method to find \( \tilde{\theta}_\infty \) and, subsequently, build an ensemble of independent GPs that all use the converged \( \tilde{\theta}_\infty \) 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 designs. 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

[16] Tao, S., Shintani, K., Bostanabad, R., Chan, Y.-C., Yang, G., Meingast, H., and Chen, W., "Enhanced Gaussian Process


