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

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

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

1 Introduction

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

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

The difficulty in fitting GPs to big data is rooted in the repetitive inversion of the sample correlation matrix, \( R \), 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 shortcomings. One avenue of research has explored partitioning the input space (and hence the training data) via, e.g., trees [19] or Voronoi cells [20], and fitting an independent GP to each partition. While particularly useful for small to relatively large datasets that exhibit the nonstationary behavior, prediction using these methods results in discontinuity (at the partitions’ boundaries) and information loss (because the query point is associated with only one partition). Projected process approximation (PPA) [21] is another method where the information from \( n \) samples is distilled into \( m \ll n \) randomly (or sequentially) selected samples through conditional distributions. PPA is very sensitive to the \( m \) selected samples, however, and overestimates the variance [21]. In Bayesian committee machine (BCM) [22], the dataset is partitioned into \( m \) 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] methods. The \( m \) samples in these methods are chosen randomly or in a greedy fashion to minimize some cost function. While the many variants of subset selection may be useful in some applications, they waste information and are not applicable to very large datasets due to the computational and storage costs. Local methods also use subsets of the data because they fit a stationary GP (for each prediction) to a very small number of training data points that are closest to the query point. Locally approximate Gaussian process (LAGP) [28] is perhaps the most widely recognized local method where the subsets are selected either based on their proximity to the query point or to minimize the predictive variance. Despite being useful for nonstationary and relatively large datasets, local methods also waste some information and can be prohibitively expensive for repetitive use since local samples have to be found and a GP must be fitted for each prediction.

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

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

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

**2 Review on Gaussian Process Modeling**

Below, we describe how GP emulators (also known as surrogate, metamodels, or models) can replace a computer simulator. The procedure is identical if the data are obtained from physical experiments. Let us denote the output and inputs of a computer simulator by, respectively, \( y \) and the \( d \) dimensional vector \( x = [x_1, x_2, ..., x_d]^T \), where \( x \in \mathbb{R}^d \). Assume the input–output relation is a realization of the random process \( \eta(x) \):

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

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

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

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

\[
r(x, x') = \exp \left\{ -|x - x'|^2 \Omega(x - x') \right\} \tag{3}
\]

where \( \Omega = \text{diag}(10^6) \) and \( \omega = [\omega_{11}, \omega_{22}, ..., \omega_{dd}]^T \), \( -\infty < \omega_i < \infty \) 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 \( (x_i, y_i) \), 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., \( \beta = 0 \)) and employ MLE. These choices are widely practiced because a high predictive power is provided while computational costs are minimized [28,35–39].

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

\[
\log(L) = \frac{1}{2} \log(|R|) + \frac{1}{2} \log(\sigma^2) + \frac{1}{2} \sigma^2 (y - \hat{\beta})^T R^{-1} (y - \hat{\beta}) \tag{4}
\]

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

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

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

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

\[
\hat{\omega} = \arg \min_{\omega} n \log(\hat{\sigma}^2) + \log(|R|) = \arg \min_{\omega} \frac{1}{2} \log(\omega L) \tag{7}
\]

By numerically minimizing \( L \) in Eq. (7), one can find \( \hat{\omega} \). Many global optimization methods such as genetic algorithm (GA) [40], pattern searches [41,42], and particle swarm optimization [43] have been employed to solve for \( \hat{\omega} \) in Eq. (7). However, gradient-based optimization techniques are commonly preferred due to
their ease of implementation and superior computational efficiency [15,16,35]. To guarantee global optimality in this case, the optimization is done numerous times with different initial guesses. It is noted that, in practice, the search space of $\omega_0$ is generally limited to $[-20, 5]$ rather than $(-\infty, \infty)$ since the correlation exponentially changes as a function of $\omega_0$ (see also Fig. 3).

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

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

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

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

where $h = (1 - 1V^{-1}1)g(x')$.

If the training dataset has multiple outputs, one may fit either a single-response GP emulator to each response or a multiresponse GP to all the responses. We follow Ref. [44] and extend the above formulations to simulators with $q$ responses by placing a constant mean for each response (i.e., $\beta = [\hat{\beta}_1, \ldots, \hat{\beta}_q]^T$) and employing the separable covariance function:

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

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 $x$. The MLE approach described above can also be applied to multiresponse datasets in which case $\sigma^2$ will be replaced with $\Sigma$ (see Refs. [45–48] for details).

Finally, we note that GPs can address noise and smooth the data (i.e., avoid interpolation) via the so-called nugget or jitter parameter, $\delta$, in which case $R$ is replaced with $R_{\delta} = R + 0I_{\delta}$. If $\delta$ is used, the estimated (stationary) noise variance in the data would be $\delta^2$. We have recently developed an automatic method to robustly detect and estimate noise [35].

### 3 Globally Approximate Gaussian Process

Regardless of the optimization method used to solve for $\omega$, 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 a collection of independent GPs that use the same $\omega$ and share the training data among themselves.

To illustrate, we consider the function $y = x^4 - x^3 - 7x^2 + 3x + 5\sin(5x)$ over $-2 \leq x \leq 3$. The associated likelihood profile (i.e., $L$) is visualized in Fig. 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 [49,50] and plot the corresponding $L$. As illustrated in Fig. 1, even though a total of 20 curves are plotted, only four are visible since the five curves with the same $n$ are indistinguishable. (ii) As $n$ increases, $L$ is minimized at similar $\omega$’s.

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

Assuming a very large training dataset of size $n$ is available, we first randomly select a relatively small subset of size $n_0$ (e.g., $n_0 = 500$) and estimate $\omega_0$ with a gradient-based optimization technique. Then, we add $n_1$ random samples (e.g., $n_1 = 250$) to this subset and estimate $\omega_1$ while employing $\omega_0$ as the initial guess in the optimization. This process is stopped after $s$ steps when $\omega$ does not change noticeably (i.e., $\omega_s \approx \omega_{s+1}$) as more training data are used. The latest solution, denoted by $\omega_{\text{opt}}$, is then employed to build $m$ GP models, each with $n_0 + s + x_n_s$ samples chosen randomly from the entire training data such that $n = \sum_{i=1}^{m} n_i$. Here, we have assumed that the collection of these GPs (who have $\omega_{\text{opt}}$ as their hyperparameters) approximate a GP that is fitted to the entire training dataset and, correspondingly, call it GAGP. The algorithm of GAGP is presented in Fig. 2.

We point out the following important features regarding GAGP. First, we recommend using gradient-based optimizations throughout the entire process because (i) if $n_0$ is large enough (e.g., $n_0 > 500$), one would need to select only a few initial guesses to find the global minimizer of Eq. (7), i.e., $\omega_0$ (we suggest the method developed in Ref. [35] to estimate $\omega_0$); and (ii) we want to use $\omega_{s-1}$ as the initial guess for the optimization in the $s$th step to ensure fast convergence since the minimizer of $L$ changes slightly as the dataset size increases (see Fig. 1). Regarding the choice on $n_0$, note that it has to be small enough to avoid prohibitive computational time but large enough, so that (i) the global optimum changes slightly if $n_0 + s$ data points are used instead of $n_0$ data points, and (ii) most (if not all) of the local optima of $L$ are smoothed out. Second, for predicting the response, Eq. (8) is used for each of the $m$ GP models, and then, the results are averaged. In our experience, we observe very similar prediction results with different averaging (e.g., weighted averaging where the weights are proportional to inverse variance) or pooling (e.g., median) schemes. The best scheme for a particular problem can be found via CV, but we avoid this step to ease of use and generality. The advantages of employing a collection of models (in our case the $m$ GPs) in prediction are extensively demonstrated in the literature [14,22]. Third, the predictive power is not sensitive to $n_0, s$, and $n_1$ so long as large

![Fig. 1](https://example.com/fig1.png)  
**Fig. 1** The profile of $L$ 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.

![Fig. 2](https://example.com/fig2.png)  
**Fig. 2** Flowchart of globally approximate Gaussian process. It is assumed that a very large training dataset of size $n$ is available.
enough values are used for them. For novice users, we recommend starting with $n_0 = 500$, $s = 6$, and $n_i = 250$, and equally distributing the samples among the $m$ resulting GPs (we use these parameters in Sec. 5 and for all the examples in Sec. 4). For more experienced users, we provide a systematic way in Sec. 4 to choose these values based on GP’s inherent ability to estimate noise using the nugget variance. Finally, we point out that GAGP has a high predictive power and is applicable to very large datasets while maintaining a straightforward implementation because it only entails integrating a GP modeling package such as GPM [35] with the algorithm presented in Fig. 2.

4 Comparative Studies on Analytical Examples

To validate the performance of GAGP in regression, we compare its predictive power on five examples (Ex1–5) against recognized big data learners: LAGP [28], GBT [14], and feedforward NNs [51]. As shown in Eqs. (11)–(15), the examples cover a wide range of dimensionality and input–output complexity.

Ex1 is a 1D function with many local fluctuations due to the sin(5x) term. Ex2 is a 4D function that is primarily sensitive to the first two inputs. Ex3 is a 7D function that models the cycle time of a piston, which mainly depends on the first and fourth inputs. Ex4 is a very complex 10D function relating the stored potential in a material to the applied load and material properties; the inputs interact nonlinearly and all affect the response. Finally, Ex5 is a 20D function where the output is independent of $x_5$ and $x_{15}$.

- **Ex1**:

\[
y = x^4 - x^3 - 7x^2 + 3x + 5 \sin(5x) \\
-2 \leq x \leq 3
\]  

(11)

- **Ex2** [52]:

\[
y = \left(1 - \exp\left(-\frac{1}{2x_1}\right)\right)\left(\frac{x_1^2}{2} + 1900x_1^2 + 2092x_1 + 60\right) \\
\text{min}(x) = [0, 0, 2200, 85] \\
\text{max}(x) = [1, 1, 2400, 110]
\]  

(12)

- **Ex3** [53]:

\[
y = 2\pi \sqrt{\frac{x_1}{4}} \left(\frac{x_1}{x_2 + 19.62x_1 - x_4x_5} + 4x_4x_5} \right) \\
\text{min}(x) = [30, 0.005, 0.002, 1000, 9 \times 10^5, 290, 340] \\
\text{max}(x) = [60, 0.02, 0.01, 5000, 11 \times 10^5, 296, 360]
\]  

(13)

- **Ex4** [54]:

\[
y = \frac{9}{2}x_5^2 + \frac{9x_5x_6}{1 + x_7} \\
\epsilon = \left(\frac{x_1}{x_2 + 19.62x_1 - x_4x_5 + x_6x_7} + 4x_4x_5\right) \\
\text{min}(x) = [-0.1, -0.1, -0.1, -0.1, -0.1, 0.02, 1, 5, 0.1] \\
\text{max}(x) = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 5, 25, 1.5]
\]  

(14)

- **Ex5** [53]:

\[
y = \frac{5x_1^2}{1 + x_1} + 5(x_4 - x_20)^2 + x_5 + 40x_1^3 - 5x_1 + 0.05x_2 + 0.08x_3 \\
- 0.03x_4 + 0.03x_7 - 0.09x_9 - 0.01x_{10} - 0.07x_{11} + 0.25x_{13} \\
- 0.04x_{14} + 0.06x_{15} - 0.01x_{17} - 0.03x_{18} \\
- 0.5 \leq x_i \leq 0.5 \text{ for } i = 1, \ldots, 20
\]  

(15)

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

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

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

In engineering design, it is highly desirable to employ interpretable methods and tools that facilitate the knowledge discovery and decision-making processes. Contrary to many supervised learning techniques such as NNs and random forests that are black boxes, the structure of GPs can provide qualitative insights. To demonstrate, we rewrite Eq. (3) as $r(x, x') = \exp \left(-\sum_{i=1}^{d} 10^{\omega_i} (x_i - x'_i)^2\right)$. If $\omega_i < 0$ (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 $x'$ (see Fig. 3 for a 1D illustration). This contribution increases as the magnitude of $\omega_i$ increases. In a GP with a 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
Table 1 Root mean squared error with small noise

<table>
<thead>
<tr>
<th>Noise SD</th>
<th>LAGP</th>
<th>GBT</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>0.200</td>
</tr>
<tr>
<td>Ex2 (4D)</td>
<td>0.01</td>
<td>1.386</td>
<td>0.121</td>
<td>0.100</td>
</tr>
<tr>
<td>Ex3 (7D)</td>
<td>0.1</td>
<td>0.129</td>
<td>0.118</td>
<td>0.100</td>
</tr>
<tr>
<td>Ex4 (10D)</td>
<td>0.01</td>
<td>0.210</td>
<td>0.048</td>
<td>0.012</td>
</tr>
<tr>
<td>Ex5 (20D)</td>
<td>0.1</td>
<td>1.450</td>
<td>0.351</td>
<td>0.101</td>
</tr>
</tbody>
</table>

Note: Smallest errors are in bold.

Table 2 Root mean squared error with large noise

<table>
<thead>
<tr>
<th>Noise SD</th>
<th>LAGP</th>
<th>GBT</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>2.000</td>
</tr>
<tr>
<td>Ex2 (4D)</td>
<td>1</td>
<td>1.739</td>
<td>1.123</td>
<td>1.002</td>
</tr>
<tr>
<td>Ex3 (7D)</td>
<td>1</td>
<td>1.037</td>
<td>1.098</td>
<td>1.002</td>
</tr>
<tr>
<td>Ex4 (10D)</td>
<td>0.1</td>
<td>0.234</td>
<td>0.120</td>
<td>0.102</td>
</tr>
<tr>
<td>Ex5 (20D)</td>
<td>1</td>
<td>1.911</td>
<td>1.155</td>
<td>1.011</td>
</tr>
</tbody>
</table>

Note: Smallest errors are in bold.

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

<table>
<thead>
<tr>
<th>Noise SD</th>
<th>LAGP</th>
<th>GBT</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>2.000</td>
</tr>
<tr>
<td>Ex2 (4D)</td>
<td>1</td>
<td>1.739</td>
<td>1.123</td>
<td>1.002</td>
</tr>
<tr>
<td>Ex3 (7D)</td>
<td>1</td>
<td>1.037</td>
<td>1.098</td>
<td>1.002</td>
</tr>
<tr>
<td>Ex4 (10D)</td>
<td>0.1</td>
<td>0.234</td>
<td>0.120</td>
<td>0.102</td>
</tr>
<tr>
<td>Ex5 (20D)</td>
<td>1</td>
<td>1.911</td>
<td>1.155</td>
<td>1.011</td>
</tr>
</tbody>
</table>

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

Table 1 Root mean squared error with small noise

<table>
<thead>
<tr>
<th>Noise SD</th>
<th>LAGP</th>
<th>GBT</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>0.200</td>
</tr>
<tr>
<td>Ex2 (4D)</td>
<td>0.01</td>
<td>1.386</td>
<td>0.121</td>
<td>0.100</td>
</tr>
<tr>
<td>Ex3 (7D)</td>
<td>0.1</td>
<td>0.129</td>
<td>0.118</td>
<td>0.100</td>
</tr>
<tr>
<td>Ex4 (10D)</td>
<td>0.01</td>
<td>0.210</td>
<td>0.048</td>
<td>0.012</td>
</tr>
<tr>
<td>Ex5 (20D)</td>
<td>0.1</td>
<td>1.450</td>
<td>0.351</td>
<td>0.101</td>
</tr>
</tbody>
</table>

Note: Smallest errors are in bold.

Table 2 Root mean squared error with large noise

<table>
<thead>
<tr>
<th>Noise SD</th>
<th>LAGP</th>
<th>GBT</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>2.000</td>
</tr>
<tr>
<td>Ex2 (4D)</td>
<td>1</td>
<td>1.739</td>
<td>1.123</td>
<td>1.002</td>
</tr>
<tr>
<td>Ex3 (7D)</td>
<td>1</td>
<td>1.037</td>
<td>1.098</td>
<td>1.002</td>
</tr>
<tr>
<td>Ex4 (10D)</td>
<td>0.1</td>
<td>0.234</td>
<td>0.120</td>
<td>0.102</td>
</tr>
<tr>
<td>Ex5 (20D)</td>
<td>1</td>
<td>1.911</td>
<td>1.155</td>
<td>1.011</td>
</tr>
</tbody>
</table>

Note: Smallest errors are in bold.

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

<table>
<thead>
<tr>
<th>Noise SD</th>
<th>LAGP</th>
<th>GBT</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>2.000</td>
</tr>
<tr>
<td>Ex2 (4D)</td>
<td>1</td>
<td>1.739</td>
<td>1.123</td>
<td>1.002</td>
</tr>
<tr>
<td>Ex3 (7D)</td>
<td>1</td>
<td>1.037</td>
<td>1.098</td>
<td>1.002</td>
</tr>
<tr>
<td>Ex4 (10D)</td>
<td>0.1</td>
<td>0.234</td>
<td>0.120</td>
<td>0.102</td>
</tr>
<tr>
<td>Ex5 (20D)</td>
<td>1</td>
<td>1.911</td>
<td>1.155</td>
<td>1.011</td>
</tr>
</tbody>
</table>

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

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

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

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

Note that calculating the Sobol’s SIs involves evaluating \( f \) for hundreds of thousands of samples, while a GP can distill similar sensitivities from a dataset. To show this, for each \( \alpha \), 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 \( \hat{\omega}_2 \), implying \( x_1 \) is more important than \( x_2 \).
- As \( \alpha \) increases, \( \hat{\omega}_1 \) increases (\( x_1 \) becomes more important), while \( \hat{\omega}_2 \) changes negligibly (the underlying functional relation between \( x_2 \) and \( y \) does not depend on \( \alpha \)).
- For a given \( \alpha \), the estimates change insignificantly when \( n \) is increased.

The above feature is present in GAGP as well and depicted by the convergence histories for Ex3 and Ex5 in Figs. 4 and 5, respectively. Similar to Fig. 1, it is evident that the estimated roughness parameters do not change noticeably as more samples are used in training (only 6 of the 20 roughness parameters are plotted in Fig. 5 for a clearer illustration). The values of these parameters can determine which inputs (and to what extent) affect the output. For instance, in Ex5, \( \hat{\omega}_1 \) is very small so the output must be almost insensitive to \( x_1 \). In addition, since \( \hat{\omega}_2 \neq \hat{\omega}_2 \), it is expected that the corresponding inputs should affect \( y \) similarly. These observations agree with the analytical result between \( x \) and \( y \) in Ex5, where \( y \) is independent of \( x_1 \) and symmetric with respect to \( x_1 \) and \( x_2 \). By using GAGP, such information can also be extracted from a training dataset whose underlying functional relation is unknown and subsequently used for sensitivity analysis or dimensionality reduction (e.g., in Ex5, \( x_1 \) and \( x_10 \) can be excluded from the training data).

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

We close this section with some theoretical discussions related to the use of the same hyperparameters within a collection of
independent GPs. In Bayesian experimental design [55], multidimensional integrals ubiquitously arise when maximizing the expected information gain, i.e., \( E[I] \). By quantifying \( I \) using Kullback–Leibler divergence [56], it can be shown [57] that

\[
E[I] = \int \left( \log \left( \frac{p(y|x)}{p(y)} \right) \right) p(y|x) dy, \quad \text{where } \theta \text{ are the hyperparameters (to be estimated), } y \text{ are the observables in the } i \text{th experiment, and } p(\cdot) \text{ is the probability density function.} \]

The nested integral renders maximizing \( E[I] \) prohibitively expensive. To address this and facilitate integration, Laplace’s theorem is used to approximate \( p(\theta|y) \) via a multivariate Gaussian likelihood or log likelihood, such as in Eq. (4). Following the central limit theorem, the accuracy of this approximation increases as the number of data points increases since the likelihood would more closely resemble a unimodal multivariate Gaussian curve [58].

With GAGP, we essentially make the same approximation, i.e., Eq. (4) approximates a unimodal multivariate Gaussian curve in the log space whose minimizer insignificantly changes when the training data are massive (note that the function value, \( L \), does change; see Fig. 1).

5 Data-Driven Design of Metamaterials

To demonstrate the application of GAGP in engineering design, we employ it in a new data-driven method for the optimization of metamaterial unit cells using big data. Although various methods, e.g., TO and GA, have been applied to design metamaterials with prescribed properties, these are computationally intensive and suffer from sensitivity to the initial guess as well as disconnected boundaries when using multiple unit cells. A promising solution is to construct a large database of precomputed unit cells (also known as microstructures or building blocks) for efficient selection of well-connected unit cells from the database as well as inexpensively optimization of new unit cells [9–12]. However, with the exception of Ref. [12] where unit cells are parameterized via geometric features like beam thickness, research in this area thus far use high-dimensional geometric representations (e.g., signed distance functions [9] or voxels [11]) that increase the memory use high-dimensional geometric representations (e.g., signed distance features like beam thickness). As elaborated below, our method starts by building an initial dataset and then proceeds to efficiently cover the input (geometry) and output (property) spaces as widely as possible.

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

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

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

where \( x_{\text{new}} \) and \( x_{\text{old}} \) are the coordinates of the new and original pixel locations, \( x_i \) 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, respectively, and \( R_0 \) is the outer distortion radius. \( r_{\text{new}} \) can be expressed as follows:

\[
r_{\text{new}} = \begin{cases} 
\frac{1}{2} R_0 \left( 1 - \cot \left( \frac{\gamma}{2} - \beta \right) \right) & \text{if } \gamma > 0 \\
\frac{1}{2} R_0 \left( 1 - \cot \left( \frac{\gamma}{2} + \beta \right) \right) & \text{if } \gamma < 0 \\
r_{\text{old}} & \text{otherwise}
\end{cases}
\]
orthotropic metamaterials database can be uniquely characterized with the first 10–15 eigenvalues in the LB spectrum. (iii) The spectrum embodies some geometrical information, including perimeter, area, and Euler number. This can be beneficial for the construction of the machine learning model as less training data may be required to obtain an accurate model compared with voxel- or point-based representations. (iv) Similar shapes have close LB spectrum, which may also help the supervised learning task.

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

\[
\Delta f = -\lambda f
\]

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

\[
\Delta := \text{div}(\text{grad} f)
\]

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

\[
0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq < \infty
\]

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

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

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

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

5.3 Machine Learning: Linking LB Representation to Property via GAGP. Once the dataset is built, we follow the algorithm in Fig. 2 for machine learning, i.e., relating the LB representations of unit cells to their stiffness tensor. We use the same fitting parameters as in Sec. 4 (\( n_0 = 500, s = 6, n_1 = 250 \)), equally distribute the samples among the \( m = 88000/(500 \times 6 \times 250) = 44 \) GP, and use Eq. (10) to have a multiresponse model that leverages the correlation between the responses to have a higher predictive power. The convergence histories are provided in Fig. 9, where the trends are consistent with those in Sec. 4. It is observed that the
The 16 estimated roughness parameters do not change noticeably once more than 1000 samples are used in training. In particular, 3 of the 16 roughness estimates, which correspond to $\lambda_{14}$, $\lambda_{15}$, and $\lambda_{16}$ are very small, indicating that those LB descriptors do not affect the responses. The next largest estimate belongs to $\omega_1 \approx -8$, which corresponds to $\lambda_{13}$. The rest of the estimates are all between 2.5 and 3, implying that the first 12 eigenvalues (shape descriptors) affect the responses similarly and nonlinearly (since large $\omega_i$ indicates rough response changes along dimension $i$). These observations agree well with the fact that the higher order eigenvalues generally explain less variability in the data. The estimated noise variances (one per response) also converge, with $E_{xy}$ having the largest estimated noise variance in the data, which is potentially due to larger numerical errors in property estimation.

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

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

### 5.4 Data-Driven Unit Cell Optimization

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

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

$$
\min_{\lambda} \| E' - E^0 \|_{\infty}
$$

s.t. $\lambda_{i-1} \leq \lambda_i \leq 1.1 \lambda_i^0$

(24)

where $E'$ and $E^0$ are the vectorized forms of, respectively, the target and predicted stiffness tensors. $\lambda = [\lambda_1, \ldots, \lambda_{16}]$ and $\lambda^0$ are the LB spectra of, respectively, candidate unit cell for the target stiffness tensor and the unit cell closest to the prescribed properties in the property space ($\lambda_i$ is 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. Note that, to ensure fast convergence during GA, we limit the search space for GA using the LB spectrum of the unit cell in the training dataset whose properties are closest to the target values. For instance, in Ex1, the GAGP and NN results using the large dataset achieve the target \( E_x \), whereas the GP result with the small dataset differs from the target by around 12%. Ex2 shows a similar pattern, with the GAGP, NN, and GP results differing from the target \( E_x \) by 4%, 10%, and 16%, respectively. When the small dataset is used, the greater deviations from the target properties can be mainly attributed to insufficient training samples and the relatively small search space. This reinforces the need for a large database of unit cells in the data-driven design of metamaterials, along with an expedient machine learning method for big data. Moreover, unit cells designed with GAGP have smaller deviations than those with NN.

### 6 Conclusion and Future Works

In this work, we proposed a novel approach, GAGP, to enable GP modeling of massive datasets. Our method is centered on the observation that the hyperparameter estimates of a GP converge to some limit values, \( \tilde{\omega}_s \), as more training samples are added. We introduced an intuitive and straightforward method to find \( \tilde{\omega}_s \) and, subsequently, build a collection of independent GPs that all use the converged \( \tilde{\omega}_s \) as their hyperparameters. These GPs randomly distribute the entire training dataset among themselves, which allows inference based on the entire dataset by pooling the predictions from the individual GPs. The training cost of GAGP primarily depends on the initial optimization with \( n_s \) data points and the \( s \) optimizations thereafter. The former cost is the same as fitting a conventional GP with \( n_s \) samples. The latter is an additional cost but is generally manageable since a single initial guess close to the global optimum is available at each iteration. The cost of building \( m \) GPs is negligible compared to these optimization costs. The prediction cost, although being \( O(m \times n_s^2) \), is small enough for practical applications.

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

As a case study, we applied GAGP to a data-driven metamaterials unit cell design process that attains desired elastic properties by transforming the complex material design problem into a parametric one using spectral descriptors. After mapping reduced-dimensional geometric descriptors (LB spectrum) to properties through GAGP, unit cells with properties closest to the target values are discovered by using the LB spectrum in the spectrum space as the objective function. For faster convergence, the unit cell closest to the optimal LB spectrum in the spectrum space is taken as the initial guess in the reconstruction process.

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

From a quantitative point of view, our data-driven design method with the large database can, compared with the small dataset case, discover unit cells with properties that are closer to the target values. Note: MSE errors (and their variances) are obtained via 28,000 random repetitions.

### Acknowledgment

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

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

\( d \) = input dimensionality
\( n \) = number of training samples
\( q \) = output dimensionality
\( s \) = number of times that \( n_t \) samples are added to \( n_0 \)
\( x \) = vector of \( d \) inputs
\( y \) = vector of \( q \) outputs
\( L \) = objective function in MLE
\( \mathbf{R} \) = sample correlation matrix of size \( n \times n \)
\( \text{GP} \) = Gaussian process

MLE = maximum likelihood estimation
\( \delta \) = Nugget or jitter parameter
\( n_0 \) = number of initial random samples
\( n_s \) = number of random samples added to \( n_0 \) per iteration
\( \omega \) = roughness parameters of the correlation function
\( \omega_{0\sigma} \) = estimate of \( \omega \) via MLE with very large training data

References


