Received: 2 May 2017 | Revised: 13 October 2017 | Accepted: 6 December 2017

DOI: 10.1002/nme.5751

RESEARCH ARTICLE

WILEY

Leveraging the nugget parameter for efficient Gaussian process modeling

Ramin Bostanabad[1] | Tucker Kearney[1] | Siyu Tao[1] | Daniel W. Apley[2] | Wei Chen[1,3]

[1]Department of Mechanical Engineering, Northwestern University, Evanston, Illinois, USA

[2]Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, USA

[3]Department of Materials Science and Engineering, Northwestern University, Evanston, Illinois, USA

Correspondence
Wei Chen, Department of Mechanical Engineering, Northwestern University, Evanston, IL 60208 USA; or Department of Materials Science and Engineering, Northwestern University, Evanston, IL 60208 USA.
Email: weichen@northwestern.edu

Funding information
National Science Foundation, Grant/Award Number: 1537641; Air Force Office of Scientific Research (AFOSR), Grant/Award Number: FA9550-12-1-0458

Summary

Gaussian process (GP) metamodels have been widely used as surrogates for computer simulations or physical experiments. The heart of GP modeling lies in optimizing the log-likelihood function with respect to the hyperparameters to fit the model to a set of observations. The complexity of the log-likelihood function, computational expense, and numerical instabilities challenge this process. These issues limit the applicability of GP models more when the size of the training data set and/or problem dimensionality increase. To address these issues, we develop a novel approach for fitting GP models that significantly improves computational expense and prediction accuracy. Our approach leverages the smoothing effect of the nugget parameter on the log-likelihood profile to track the evolution of the optimal hyperparameter estimates as the nugget parameter is adaptively varied. The new approach is implemented in the **R** package **GPM** and compared to a popular GP modeling **R** package (**GPfit)** for a set of benchmark problems. The effectiveness of the approach is also demonstrated using an engineering problem to learn the constitutive law of a hyperelastic composite where the required level of accuracy in estimating the response gradient necessitates a large training data set.

KEYWORDS

computer experiments, Gaussian process, hyperparameter estimation, ill-conditioned matrix, nugget parameter

1 | INTRODUCTION

Since the rise of simulation-based science and engineering, the field of surrogate modeling has become invaluable to replace expensive computer simulations or real experiments with fast and accurate surrogate models. The use of Gaussian processes (GPs), proposed by Sacks et al,[1] as surrogates has been of particular interest. GP models can interpolate the data by viewing the response surface as a realization of a Gaussian random process. They also have a natural mechanism to model noisy data (ie, to avoid interpolation) and have been widely used in a variety of applications such as providing insights into the problem of interest[1-3] (eg, determining the response sensitivities to inputs), quantifying the prediction uncertainty,[4] optimizing expensive functions,[5-9] and enabling tractable and efficient Bayesian calibration and bias correction.[10,11]

A common rule of thumb for fitting a GP model to a well-behaved response surface is to have at least $10D$ observation points,[12] where D is the dimensionality of the inputs denoted by $\boldsymbol{x} = [x_1, \ldots, x_D]$. However, as computational resources

have become more available, it is now far less expensive to collect data and far more desirable to create large training data sets and fit more accurate metamodels.[13] It is well known in the literature that, as the size of the training data set increases, the process of fitting a GP model becomes hampered by numerical issues[14] and the fitting costs increase prohibitively (we elaborate on these issues in Section 2). Methods such as fixed rank kriging,[15] local kriging,[16] and domain partitioning[16-18] exist for addressing these numerical issues when fitting GP models to large data sets. However, the predictions may not depend on all available observations. In contrast, this paper focuses on the problem of fitting a *global* GP model such that the future predictions depend on the entire training data set to ensure that both long- and short-range correlations are considered. The proposed approach improves the predictive power and decreases the numerical instabilities and computational expense that arise when the data set is large and/or the dimensionality is high.

We note that, for the purposes of this work, large data set refers to the order of up to thousands of observations and high dimensionality refers to more than 20 dimensions. A classic problem that can be very high dimensional and require large training data sets is building structure–property relations in materials[19-22] where the inputs (or, in machine learning parlance, the predictors) correspond to statistical descriptors that characterize the material microstructure.[23] Because the properties of such materials are generally highly nonlinear (when viewed as a function of microstructural descriptors), a large training data set is required to understand how the properties can be linked to the microstructural descriptors.

To fit a GP model, the most common approach is to use maximum likelihood estimation (MLE)[24-26] for estimating the model parameters. Although many optimization methods such as genetic algorithms (GAs),[27] pattern searches,[28,29] and particle swarm optimization (PSO)[30] have been used in the MLE process, gradient-based optimization techniques are commonly preferred due to their ease of implementation and computational efficiency. These methods, however, do not guarantee *global optimality*, and so it is essential to perform the optimization procedure numerous times with different initial guesses. Additionally, the profile of the objective function in the MLE process is known to have flat regions (ie, the gradient is small), which can render the convergence slow. These 2 issues greatly increase the cost of fitting an accurate GP model with gradient-based optimization methods (although the overall cost is still less than that achieved with, for example, GA or PSO[31]).

Inversion of the correlation matrix is an integral part of the GP modeling, and the computational bottleneck in fitting a GP model to large data sets lies in repeatedly constructing and inverting the correlation matrix (see Section 2). When 2 or more observations are nearby (which is often inevitable in large data sets), the correlation matrix becomes nearly singular, and *numerical issues* arise while calculating its inverse**.** The most common approach to address this is to add a nugget or jitter δ to the diagonal elements of the correlation matrix.[1,32-34] While this improves the conditioning of the correlation matrix,[35] the resulting metamodel no longer perfectly interpolates the observations (perfect interpolation is desirable for noiseless data). In other words, inclusion of a nugget, if it is chosen too large, may misrepresent the system that the GP model is emulating because the nugget is generally employed to deal with noisy data. While some work has been done to determine how the nugget alleviates ill-conditioning,[35,36] little attention has been given to fully understanding its impact on the log-likelihood function and capitalizing on it to, simultaneously, improve the model fitting and prediction performance.

To address the aforementioned issues, we propose 2 mechanisms to improve the accuracy and efficiency in fitting the hyperparameters of a GP model. Specifically, we propose (i) a novel method to leverage the smoothing effect of the nugget parameter on the profile of the log-likelihood function to efficiently estimate the optimal hyperparameters by tracking their evolution as the nugget parameter is varied and (ii) a new criterion for determining the optimal nugget parameter for a given data set.

We emphasize that our contribution is not developing an optimization algorithm. Rather, in our method, the profile of the objective function (ie, the log-likelihood function) is adaptively changed to help the utilized optimization algorithm (eg, steepest descent or Levenberg-Marquardt[37,38]) find the global optimum. We also address the issues related to singularity of the correlation matrix, fitting costs, and inclusion of a nugget. As for the nugget parameter, in particular, its inclusion in the model and the choice of its final value are thoroughly examined. In the literature, there are major disagreements on when to include the nugget and what its value should be, especially in the case of noiseless data sets. This is partly because the inclusion of a nugget represents noise in observing the response values and avoids perfect interpolation. In spite of this, Gramacy and Lee[34] argue that a nugget should always be included to combat numerical problems and help eliminate potential model biases that are inherently introduced in simulation works, even for noiseless data. Multiple works practice the inclusion of nugget solely based on the condition number κ of the correlation matrix.[33,39,40] From a new perspective, our studies indicate that controlling the minimum eigenvalue of the correlation matrix while minimizing the cross-validation error provides a robust and computationally sound method to decide when the nugget should be used and what value should be assigned to it.

The approach outlined in this paper is implemented in the **R** package **GPM**, which is publicly available through CRAN. Our approach can fit accurate and global predictive metamodels to large data sets for problems with high dimensionality. To validate this claim, we test the performance of the package against a set of benchmark problems spanning a wide range of data set size and problem dimensionality. As previously mentioned, one such application is learning (ie, predicting) the properties of new materials. While neural networks have been previously used to computationally predict material properties,[23] our approach requires far less data than typical neural network applications because neural networks inherently require more data to ensure accuracy. Creating metamodels to predict the material properties of a microstructure would greatly improve the computational expense of multiscale modeling, which is currently one of the major obstacles in computational materials science.[19]

The remainder of this paper is organized as follows. Section 2 presents a general overview of GP modeling. Section 3.1 presents our method for leveraging the nugget to more effectively estimate GP model hyperparameters. Details on the choice of the final value of the nugget and the implementation of the proposed approach in **GPM** are included in Sections 3.2 and 3.3, respectively. A performance comparison between **GPM** and the popular **R** package **GPfit**[31] for 7 noiseless problems of various dimensionality and data set sizes is provided in Section 4. Section 5 illustrates the use of **GPM** on a data set used to model the constitutive law of a hyperelastic composite, and Section 6 concludes this paper.

2 | GP METHODOLOGY

A GP model is a special case of a kriging model originally developed by mining engineer D. G. Krige. The GP predictor is the best linear unbiased predictor[41] in that it minimizes the mean squared error (MSE) of prediction among all unbiased linear predictors.

The standard formulation of a GP model is

$$Y(\boldsymbol{x}) = f(\boldsymbol{x}) + Z(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x} = [x_1, \ldots, x_D]$ denotes the D-dimensional input vector, $f(\boldsymbol{x})$ is a linear combination of some known set of basis functions (such as polynomial or exponential), and $Z(\boldsymbol{x})$ is a zero-mean stationary stochastic process. Taking $f(\boldsymbol{x}) = \beta$, the global prior mean of the process, we arrive at the ordinary GP formulation

$$Y(\boldsymbol{x}) = \beta + Z(\boldsymbol{x}), \tag{2}$$

which is commonly regarded as producing accurate predictions as long as one avoids extrapolation.[42] As previously mentioned, $Z(\boldsymbol{x})$ is assumed to be stationary with a covariance function of the form

$$cov\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \sigma^2 R\left(\boldsymbol{x}, \boldsymbol{x}'\right), \tag{3}$$

where $R(\cdot)$ is the spatial correlation function (SCF) and σ^2 is the prior variance. While the **GPM** package can handle multiple-response data sets, in the following discussion we focus on a single-response model and refer the readers to the work of Kennedy and O'Hagan.[43] Many different SCFs have been widely used in the literature, such as the Gaussian, Matern,[44] lifted Brownian,[45] and power exponential,[46] to name a few. We direct the readers to the end of the section for further details on these SCFs. Hereafter, we focus on the Gaussian correlation function, a special case of the power exponential correlation. The discussions can readily be extended to any other correlation function.

The Gaussian correlation function for D-dimensional data is given by

$$R\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \exp\left\{-\left(\boldsymbol{x} - \boldsymbol{x}'\right)^T \boldsymbol{\Theta}\left(\boldsymbol{x} - \boldsymbol{x}'\right)\right\}, \tag{4}$$

where $\boldsymbol{\Theta} = diag(\boldsymbol{\theta})$ and $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_D]$ are the scale (or roughness) parameters. Given a set of n observations, the elements of the correlation matrix $\boldsymbol{R}$ are $\boldsymbol{R}_{ij} = R(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $i, j \in [1, 2, \ldots, n]$. The most common method to estimate the optimal hyperparameters (ie, $\hat{\boldsymbol{\theta}}$, $\hat{\beta}$, and $\hat{\sigma}^2$) is through minimization of the negative logarithm of the likelihood function (for derivations, see the works of Mardia and Marshall[24] and Currin et al[25])

$$\hat{\boldsymbol{\psi}} = \left[\hat{\boldsymbol{\theta}}, \hat{\beta}, \hat{\sigma}^2\right] = \underset{\boldsymbol{\theta}, \beta, \sigma^2}{\operatorname{argmin}} \frac{1}{2} \ln \mid \boldsymbol{R} \mid + \frac{n}{2} \ln\left(\sigma^2\right) + \frac{(\boldsymbol{y} - \mathbf{1}\beta)^T \boldsymbol{R}^{-1} (\boldsymbol{y} - \mathbf{1}\beta)}{2\sigma^2} = \underset{\boldsymbol{\theta}, \beta, \sigma^2}{\operatorname{argmin}} \mathcal{L}, \tag{5}$$

where $\boldsymbol{y} = [y_1, \ldots, y_n]^T$ is the vector of observed responses and $\mathbf{1}$ is an $n \times 1$ vector. With the log-likelihood formulation, the closed-form solutions for β and σ^2 can be found as a function of $\boldsymbol{\theta}$ (this procedure is called profiling). Thereafter, the entire optimization procedure is expressed solely in terms of the scale parameters $\boldsymbol{\theta}$.

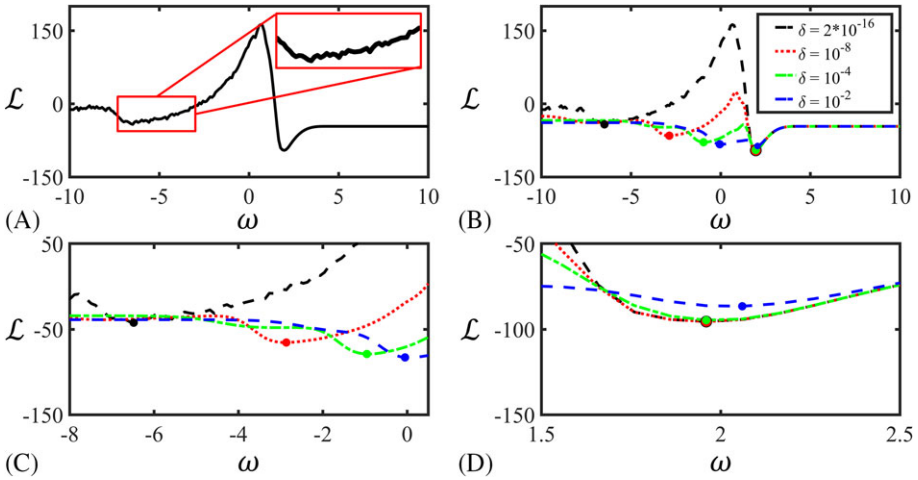

FIGURE 1 The evolution of the profile of the log-likelihood function $\mathcal{L}$ as the nugget is varied. A, The profile of $\boldsymbol{\mathcal{L}}$ for $\delta = 2 \times 10^{-16}$ showing roughness and many local minima; B, The profile of $\boldsymbol{\mathcal{L}}$ for 4 different values of δ; C and D, Magnified views from panel B where the evolution of the 2 local optima is tracked as the nugget is decreased from 10^{-2} to 10^{-16} [Colour figure can be viewed at wileyonlinelibrary.com]

The core challenge of fitting a GP model lies in the complex nature of $\mathcal{L}$ for large data sets. The function typically has many local optima and large relatively flat regions.[41,47] As aforementioned, gradient-based searches have been shown to generally outperform other optimization techniques, but to ensure global optimality, it is necessary to perform optimization many times, each with a different initial guess for $\boldsymbol{\theta}$.

These issues result in an expensive computational search to estimate the $\boldsymbol{\theta}$'s. MacDonald et al[31] improved the performance of gradient-based optimization by reparameterizing the scale parameter in the correlation function as

$$\theta_i = 10^{\omega_i} \tag{6}$$

and then optimizing $\mathcal{L}$ in Equation 5 with respect to $\boldsymbol{\omega}$. Reparameterization facilitates optimization of the objective function, but the resulting profile can still have many local optima and large flat regions (see Figure 1A). As the dimension of the problem grows, the search space for the optimal hyperparameters grows accordingly, further increasing the computational expense of the optimization procedure. This leads us to the primary research question of this work: *Is there a way to further improve the optimization of $\mathcal{L}$, ie, to reduce the number of optimization iterations while improving the ability to find the global optimum*?

Another challenge in the optimization problem involves the inversion of $\boldsymbol{R}$ in Equation 5. As the size of the training data set grows, this inversion becomes susceptible to numerical issues (arising from near singularity) and prohibitively expensive. Such issues not only hinder the optimization process but also adversely affect the predictive power of the fitted model. As noted before, currently, the most common approach to address the singularity issue is to add a nugget or jitter, δ,[1,32-34] to the correlation matrix $\boldsymbol{R}$. Mathematically, one would replace the ill-conditioned $\boldsymbol{R}$ with $\boldsymbol{R}_\delta$ where

$$\boldsymbol{R}_\delta = \boldsymbol{R} + \boldsymbol{I}\delta, \tag{7}$$

with $\boldsymbol{I}$ being the identity matrix of size $n \times n$. The value of δ is chosen to be a very small number (eg, 10^{-6}) based on past experience, determined by constraining the condition number of $\boldsymbol{R}$[31,40] or even estimated via the MLE method.[34] For noiseless data, if we decrease the value of the nugget added to the model, less smoothing will occur and the GP predictor will more closely interpolate the data (barring numerical inaccuracies, which are exacerbated when the nugget is reduced), resulting in a more accurate model. In the case of noisy data, δ should be chosen to be directly proportional to the noise variance. In light of this, the second research question of this work revolves around *whether a better criterion exists to determine (i) the inclusion of the nugget and (ii) the best nugget value*.

As previously mentioned, the choice of SCF varies greatly in the literature, and different SCFs have been used for different types of problems. Four of the most commonly used SCFs are shown in Table 1 along with their mathematical representations. The simplified versions of the power exponential and lifted Brownian SCFs (the Gaussian and lifted Brownian with $\gamma = 1$, respectively) are particularly useful in that they model smooth or analytic functions. Regardless of the choice of SCF used in a GP model, the MLE method can be used to estimate all the parameters of the correlation function.

TABLE 1 The 4 spatial correlation functions (SCFs) implemented in the GPM package. In the simplified version of lifted Brownian SCF, $\gamma = 1$, and hence, the resulting GP model is infinitely differentiable like the Gaussian SCF. For the details on the features of these SCFs and their parameters, see the work of Plumlee and Apley.[45] Bold lowercase and uppercase letters denote vectors and matrices, respectively

SCF	Mathematical Representation $R(\boldsymbol{x}, \boldsymbol{x}')$	Hyperparameters
Power Exponential	$\exp\left\{-\sum_{i=1}^{D}\theta_i\left(x_i - x_i'\right)^p\right\}$	$\boldsymbol{\theta}, \sigma^2, \beta, p$
Gaussian	$\exp\left\{-\sum_{i=1}^{D}\theta_i\left(x_i - x_i'\right)^2\right\}$	$\boldsymbol{\theta}, \sigma^2, \beta$
Lifted Brownian	$\psi(\boldsymbol{x}) - \psi(\boldsymbol{x}') - \psi(\boldsymbol{x} - \boldsymbol{x}') - 1$ with $\psi(\boldsymbol{x}) = (1 + (\boldsymbol{x}^T\boldsymbol{A}\,\boldsymbol{x})^{\gamma})^{\zeta}$	$\gamma, \zeta, \boldsymbol{A}$
Lifted Brownian with $\gamma = 1$	$\psi(\boldsymbol{x}) - \psi(\boldsymbol{x}') - \psi(\boldsymbol{x} - \boldsymbol{x}') - 1$ with $\psi(\boldsymbol{x}) = (1 + \boldsymbol{x}^T\boldsymbol{A}\,\boldsymbol{x})^{\zeta}$	$\zeta, \boldsymbol{A}$

3 | OUR APPROACH FOR IMPROVED GP FITTING

In this section, we first introduce our proposed nugget-leveraging method for improving the optimization process for the reparameterized scale parameter $\boldsymbol{\omega}$. Then, we elaborate on our approach for defining the ill-conditioning of the correlation matrix during the optimization procedure and how to determine the best value of the nugget by examining the cross-validation error. This section is concluded with a discussion on the implementation of the approach in our **GPM** package.

3.1 | Leveraging the effect of nugget on the log-likelihood function profile

When fitting a GP model using the MLE approach (see Equation 5), the goal is to find the value of $\boldsymbol{\theta}$ that minimizes $\mathcal{L}$ while searching over $(0, \infty)^D$, where D is the dimensionality of the input space. A common strategy (eg, MacDonald et al[31]) is to reparameterize by working with the log of the scale parameters to reshape the profile of $\mathcal{L}$. With this reparameterization, the hyperparameters are no longer bounded, ie, $\boldsymbol{\omega} \in (-\infty, \infty)^D$. In practice, however, the models are typically fit on a hypercube no larger than $\boldsymbol{\omega} \in [-10, 10]^D$.

To illustrate how the nugget affects the objective function, we use a simple $1D$ example in which the response function is

$$f(x) = \log(x + 3) + \sin\left(2x^2\right) + x^2, x \in [-2, 3]. \tag{8}$$

The corresponding $\mathcal{L}$ as a function of ω is plotted in Figure 1A, where 20 equally spaced observations of x in $[-2, 3]$ and a very small nugget are used. As illustrated, the left tail of $\mathcal{L}$ has multiple local optima, and the right tail is very flat. These issues, as aforementioned, make it difficult for a gradient-based optimization technique to ensure global optimality. Our studies indicate that these issues can be addressed by investigating how the nugget affects the profile of $\mathcal{L}$. In Figure 1B, the profile of $\mathcal{L}$ is plotted for various δ's. It is evident that as δ increases, $\mathcal{L}$ is smoothed more and the noisy local optima are eliminated. Hence, with larger nuggets, the optimal $\boldsymbol{\omega}$ can be estimated much faster because fewer initial points are needed in the gradient-based optimization. Although the optimal $\boldsymbol{\omega}$ is not in the exact same location for each curve in Figure 1B as δ is varied, the solutions are within the same neighborhood for small changes in δ. In other words, it is possible to determine the global optima for the no-nugget model (ie, when $\delta \to 0$) by first finding the local optima on the curve with large δ and then tracking its evolution as δ is decreased in a controlled manner. (Here, it is assumed that a model with a small or zero nugget would provide the highest predictive power. In Section 3.2, we elaborate on how to choose the best nugget value.) This is demonstrated in Figures 1C and 1D, where the 2 local optima of $\mathcal{L}$ with $\delta = 10^{-2}$ are tracked as δ is decreased incrementally.

To further illustrate these findings, we examine a more complicated example in $2D$. Given the 6-hump camelback function[42,48] defined in $x_1 \in [-2, 2], x_2 \in [-1, 1]$

$$f(x) = \left(4 - 2x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(-4 + 4x_2^2\right)x_2^2, \tag{9}$$

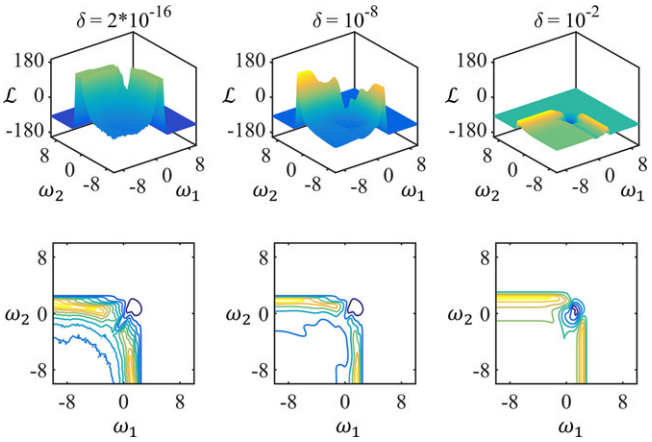

FIGURE 2 The evolution of $\mathcal{L}$ as the nugget is varied for the 2*D* example in Equation 9: as the surfaces and the corresponding contour plots indicate, $\mathcal{L}$ is smoothed as δ is increased [Colour figure can be viewed at wileyonlinelibrary.com]

we are again interested on how the profile of $\mathcal{L}$ changes as a function of δ. Figure 2 demonstrates the evolution of $\mathcal{L}$ for multiple choices of δ for 30 observations (generated with a space-filling algorithm). From the surface plots on the top row, we can clearly see the smoothing effect on the log-likelihood function as the applied δ increases. The contour plots in the bottom row illustrate that the location of the optimal solution does not change significantly across the profiles.

To investigate why the objective function is flattened and smoothed as δ increases, we re-examine how $\mathcal{L}$ is calculated. From Equation 5, we can see that $\mathcal{L}$ is composed of the sum of 3 separate terms, with the third being a constant when considering the closed-form solution for $\hat{\beta}$ and $\hat{\sigma}^2$, resulting in only 2 terms. The first is a function of the determinant of $\boldsymbol{R}$, and the second is a function of the model variance. We can view the addition of a nugget to the correlation matrix as the addition of noise to the observations. As this noise increases, the error in observations will increase, resulting in an $\mathcal{L}$ whose value is less dependent on the scale parameters (ie, $\boldsymbol{\omega}$'s) but more dependent on the noise variance, which is, in our case, the nugget. Mathematically, the determinant of a matrix is equivalent to the product of its eigenvalues; hence, by replacing $\boldsymbol{R}$ with $\boldsymbol{R}_\delta$, the eigenvalues of $\boldsymbol{R}$ will have less impact on $\mathcal{L}$ as δ increases.

Prior to describing the method in detail, we first briefly introduce our approach to detect numerical issues and apply the nugget accordingly. We will elaborate on how to choose the final value for the nugget in Section 3.2. We select the nugget δ to ensure that the minimum eigenvalue of $\boldsymbol{R}_\delta = \boldsymbol{R} + \boldsymbol{I}\delta$ exceeds a specified value ε at *any* evaluation of the correlation matrix during the optimization process. That is, we select δ as

$$\delta = \begin{cases} \varepsilon - \lambda_{min} & \lambda_{min} < \varepsilon \\ 0 & \lambda_{min} \geq \varepsilon, \end{cases} \tag{10}$$

where λ_{min} is the smallest eigenvalue of $\boldsymbol{R}$. This selection is designed to reduce or eliminate the numerical issues while controlling the smoothness of the profile of $\mathcal{L}$. Thus, we control δ indirectly via controlling ε directly. We highlight that the interpretations drawn from the discussions regarding Figures 1 and 2 (ie, that the convergence of the optimal hyperparameter can be tracked as δ is decreased) still hold if one varies ε directly instead of varying δ.

As demonstrated in Figure 3, we propose the following iterative optimization procedure to leverage the smoothing effect of nugget on $\mathcal{L}$ to estimate the hyperparameters and, simultaneously, the best nugget value. In our approach, we first conduct the optimization with an overly large ε to have the profile of $\mathcal{L}$ greatly smoothed. Next, ε is incrementally decreased and the optimization is repeated iteratively using the previous local optima as the initial guess(es) for the current iteration until a desired value of ε is achieved.

The algorithm consists of 2 main loops. The outer loop is over p candidate values of ε stored in the vector $\boldsymbol{\varepsilon} = [\varepsilon_1, \ldots, \varepsilon_p]$ such that $\varepsilon_i > \varepsilon_{i+1} > 0$ for $i = 1, \ldots, p-1$ (eg, $\boldsymbol{\varepsilon} = [10^{-1}, 10^{-2}, \ldots, 10^{-12}]$). In this loop, the goal is to first find the *distinct* local optima of $\mathcal{L}$ for ε_1 and then track their evolution as ε is gradually decreased to ε_p. As we employ a gradient-based optimization method (other methods can also be used) for each iteration over i, $\mathcal{L}$ is minimized m_i times starting from m_i initial guesses (corresponding to the inner loop: see the next paragraph for details) to ensure global optimality. Because we intend to record the evolution of the optimization solutions as ε (and, hence, δ and the profile of $\mathcal{L}$) is varied, $\boldsymbol{R}$ is replaced with $\boldsymbol{R}_{\delta_i}$ (see Equations 7 and 10) in the inner loop (ie, during the m_i minimizations) at the ith iteration.

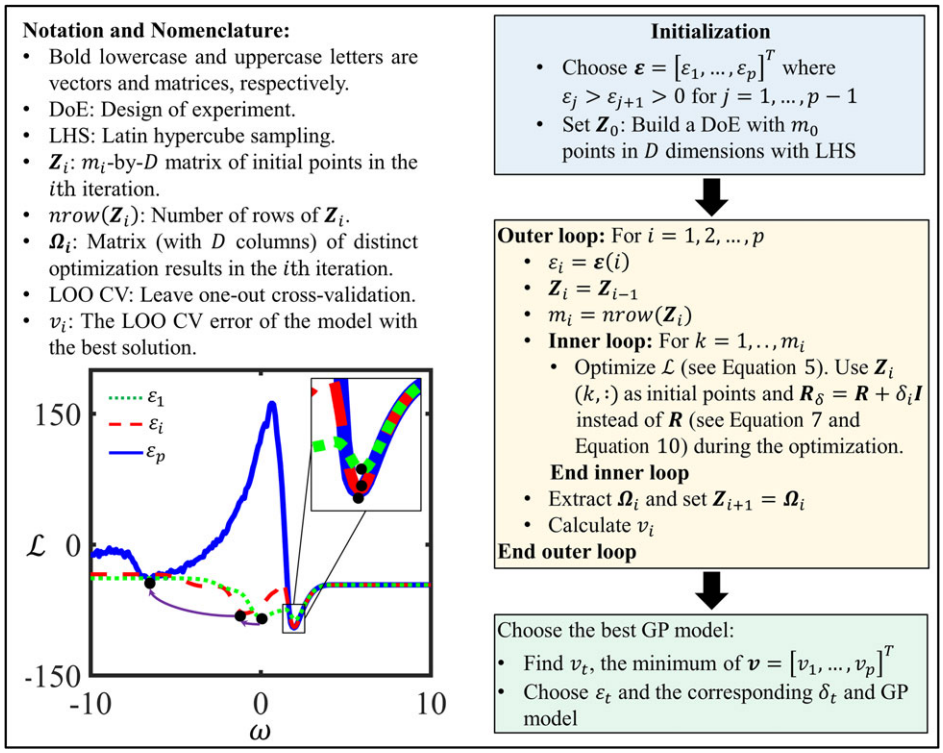

FIGURE 3 Flowchart of our iterative approach for fitting a GP model: the 2 black dots on the green dashed curve in the figure mark the local optima of $\mathcal{L}$ when $\varepsilon = \varepsilon_0$. Tracking the evolution of these points on the profile of $\mathcal{L}$ as ε is decreased constitutes the essence of our iterative optimization approach (for clarity, only 3 curves are shown). See the text for more details [Colour figure can be viewed at wileyonlinelibrary.com]

For $i = 1$ (corresponding to ε_1), the $m_0 = m_1$ initial points used in the inner loop are generated via a space-filling design of experiment (DoE) method (eg, Latin hypercube sampling [LHS]) and stored in the $\boldsymbol{Z}_0 = \boldsymbol{Z}_1$ matrix (which stores each initial point in a row vector). It must be noted that, provided that ε_1 is large enough (which will most likely* result in using a large nugget), m_0 does *not* need to be large. This is because, as explained earlier in this section, an overly large ε_1 will render the profile of $\mathcal{L}$ very smooth with only a few local optima. For the next iterations (ie, $i > 1$), the optimization processes in the inner loop at iteration i are initialized from the *distinct* optimization solutions at iteration $i-1$ (collectively denoted by $\boldsymbol{\Omega}$ in the flowchart). For instance, if $m_0 = m_1 = 10$ and the 10 optimizations result in 3 distinct solutions, then $m_2 = 3$. We emphasize that, for $i > 1$, the optimization processes are quite fast because from one iteration to the next (i) the number of distinct solutions generally decreases, ie, $m_i \leq m_{i-1}$, and (ii) the location of local optima (see the green and red profiles in the figure within the flowchart) changes in a small neighborhood, and hence, the optimizations of the likelihood function converge fast.

At the end of the ith iteration, the best solution is selected from the set of local optima stored in $\boldsymbol{\Omega}_i$. Then, the leave-one-out cross-validation (LOO CV) error of the corresponding GP model, v_i, is calculated via a closed-form solution (which only uses the training data set and does not require any model fitting procedure[49]). The output of the algorithm is the GP model whose LOO CV error is the least. We elaborate on this criterion in Section 3.2.

3.2 | Final value of the nugget

In GP modeling, the nugget is used to circumvent numerical issues (associated with the inversion of the correlation matrix) and/or represent legitimate noise if the observations are noisy. In the former case, perhaps the most common implementation is to check the condition number κ of $\boldsymbol{R}$ (defined as the ratio of $\boldsymbol{R}$'s maximum eigenvalue λ_{max} to $\boldsymbol{R}$'s minimum eigenvalue λ_{min}). In particular, if $\kappa > \kappa_{max}$ for some predetermined value κ_{max} (eg, 10^{16}), the matrix is determined to be ill-conditioned and, subsequently, a nugget is used to replace $\boldsymbol{R}$ with $\boldsymbol{R}_\delta = \boldsymbol{R} + \boldsymbol{I}\delta$ for some appropriately chosen δ.

*We say *most likely* because in some cases (eg, for small data sets in large dimensions), λ_{min} might be quite large itself, resulting in a zero nugget (see Equation 10).

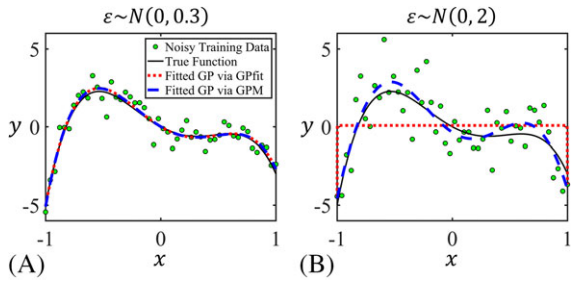

FIGURE 4 Fitting GP models to noisy data sets with the GPM and GPfit packages: training data sets are with (panel A) small and (panel B) large noise variance. Both packages are run with default settings (the Gaussian correlation function is used in **GPfit**) [Colour figure can be viewed at wileyonlinelibrary.com]

Although correlation matrices are theoretically positive semidefinite, $\lambda_{min} < 0$ sometimes occurs due to the numerical errors during optimization (eg, in the inner loop in Figure 3). Hence, as aforementioned, we use the criterion $\lambda_{min} < \varepsilon$ to designate ill-conditioning of $\boldsymbol{R}$. Following this, we use Equation 10 to select the nugget value and ensure that the minimum eigenvalue of $\boldsymbol{R}_\delta = \boldsymbol{R} + \boldsymbol{I}\delta$ exceeds the specified ε at any evaluation of the correlation matrix during the optimization of likelihood function. We note that, with this approach, the value of δ is directly a function of both ε and $\boldsymbol{\omega}$ since λ_{min} is determined by $\boldsymbol{R}$, which depends strongly on $\boldsymbol{\omega}$.

Provided that the right value is chosen for ε, the above procedure can be employed to use the nugget and address, simultaneously, numerical issues and noise. In our approach, the outer loop of the algorithm summarized in Figure 3 helps us to estimate this ε by searching over the elements of $\boldsymbol{\varepsilon} = [\varepsilon_1, \ldots, \varepsilon_p]$. That is, the final value of the nugget is chosen from the candidate values to minimize the LOO CV error of the GP model on the training data set.

In Section 4, seven noiseless problems of different dimensionalities and training data set sizes are examined to test the capability of our approach in addressing numerical issues. To illustrate how our approach may be used to handle noisy data sets, we consider the $y = -10x^4 + 5x^3 + 6x^2 - 4 + \varepsilon$ function, where $\tilde{\varepsilon}N(0, \sigma)$ is the added zero-mean noise with variance σ^2. We generated 2 training data sets of size 50 using evenly spaced points in the $x \in [-1, 1]$ region with 2 values for σ, ie, one small and one large (0.3 and 2, respectively). Then, we used both our **GPM** package and the **GPfit** package to fit GP models to these 2 data sets. In our simulations, we used the default settings for both packages and, for consistent comparisons, chose the correlation function as Gaussian in **GPfit**. Similar to Section 4, we compare the performance of our approach against **GPfit** because it is widely used to fit GP models and has been shown to outperform other GP-fitting packages.[31]

The resulting models, the training data sets, and the true function are plotted in Figure 4. It is evident that with small noise (Figure 4A), both packages can fit quite accurate models. However, for large amounts of noise (Figure 4B), only **GPM** can adapt and fit a GP model that captures the true behavior of the underlying function. We emphasize that the **GPfit** package can address noise and fit a sufficiently accurate GP model with the right input settings. However, generally, it is *not* known a priori if, and more importantly to what extent, a data set is noisy. Therefore, having an automated method to detect small to large amounts of noise in GP modeling is attractive.

3.3 | Implementation details of the GPM package

We recommend setting $\boldsymbol{\varepsilon} = [10^{-2}, 10^{-3}, \ldots, 10^{-12}]$ and $m_0 = 5$. The 5 different initial optimization points of the inner for loop, $\boldsymbol{Z}_1$, for the first iteration (ie, $i = 1$ in Figure 3) are generated via LHS over the $[-10, 5]^D$ space, where D is the dimension of $\boldsymbol{x}$. We have found these settings to provide an accurate and efficient set of solutions for the local optima of $\mathcal{L}$ with large ε_1. The solutions are then reduced to a set of distinct solutions (denoted by $\boldsymbol{\Omega}_1$), ε is decremented to 10^{-3}, and optimization is performed once more over $[-10, 5]^D$ but this time via $\boldsymbol{Z}_2 = \boldsymbol{\Omega}_1$ as the initial guesses of the optimization. This process is then repeated with ε decreasing by 1 on the $\log_{10}$ scale (ie 10^{-3}, 10^{-4}, 10^{-5} …) until either (i) $\varepsilon = 10^{-12}$ or (ii) the LOO CV increases from one optimization iteration to the next. The first termination criterion of $\varepsilon = 10^{-12}$ is chosen to ensure that there are no numerical issues when inverting the correlation matrix. LOO CV has frequently been used to characterize the model quality without a validation data set and is convenient due to its analytical closed-form representation[50] for GP modeling. For our purposes, if the LOO CV increases, it can be assumed that the performance of the metamodel has deteriorated and the optimal value of ε has been found.

Our approach is more efficient than the typical optimization approach that requires many initial guesses and is implemented in packages such as **GPfit** (which searches over $\boldsymbol{\omega} \in [-10, 10]^D$). As the dimension D or the size of the data set increases, the objective function becomes much more complex and finding the global solution becomes extremely expensive. Additionally, by requiring fewer optimization iterations, the inverse of the correlation matrix must be calculated far fewer times, which greatly improves the computational efficiency when working with large training data sets.

It should be noted that all the aforementioned default values in the **GPM** package can be adjusted by the user. The package contains extensive documentation for this purpose, and we refer the reader to that for more details. We have developed the **GPM** package both with experts (who can tune all the default values) and with novice users (who only need to model the mapping function between a set of inputs and a set of outputs) in mind. For instance, the user can set the type of the correlation function, the search space, the number of optimization iterations, and so on. Once fitted, the package can also be used to directly predict the gradient of a function, which is very attractive when, for example, GP modeling is used for optimizing an expensive function. Finally, we highlight that we have provided a visualization function in **GPM** that allows the user to easily plot any D-dimensional GP model prediction along with the associated uncertainty bounds (for $D > 2$, the user can choose 2 inputs for plotting and set fixed values for the rest of the inputs).

4 | GPM COMPARISON WITH GPFIT

To illustrate how our approach improves the efficiency and performance of GP models, we compare the performance of **GPM** against that of the popular **R** package **GPfit**[31] for a variety of analytical problems. **GPfit** was chosen as the benchmark because it is viewed as the state-of-the-art package for GP modeling. The first major advantage of **GPM** is computational cost improvement, ie, the efficiency of the algorithm in Figure 3 in estimating the optimal hyperparameters. The second claim is that our approach can fit more accurate models by finding the global optimum solution of $\mathcal{L}$ in Equation 2 (rather than a local solution) and choosing a better value for the nugget. To test these claims, we will compare the fit time and MSE of a validation (also known as test) set for both packages. In all cases, the default settings are used for both packages in the fitting process. In addition, care was taken to ensure that all problems were run on the same computer without overloading either the CPU or memory to ensure that all comparisons are fair.

In the comparison, we present 7 analytical examples (see the Appendix for details) of various sizes in terms of training set and dimensionality D. For each example, training sets of size $10D$, $20D$, and $40D$ were generated. For statistical purposes, each problem was performed with 20 unique training sets except the $40D$ case in Example 7 for which only 10 training sets were used due to computational costs of **GPfit.** In total, 390 GP models were fitted with each package. The mean fit time and MSE over the 20 replicates are reported for each problem. To calculate the MSE, a single validation set of size 10 000 was generated for each example problem independent of the training sets. All the training and validation data sets were generated via Sobol sequence[51] due to its ability to efficiently generate very large space-filling data sets.

The MSE and fit time results for both packages across all examples and training set sizes are provided in Table 2. We summarize the findings as follows. (i) Except for Examples 1 and 2, **GPM** always outperforms **GPfit** in terms of computational costs, often by a large margin (compare columns 5 and 6 in Table 2). In the cases where **GPfit** outperformed **GPM**, the difference was small. (ii) Excluding Example 1 with DoE size $10D$, the MSE achieved by **GPM** is no larger than that of **GPfit** (compare columns 3 and 4 in Table 2). Additionally, the raw MSE values in this case (example 1, $10D$) were too high for the metamodels to be considered accurate and only 3 of the 20 replicates had a lower MSE when fitted with **GPfit** compared to **GPM**. (iii) The performance improvement for **GPM** in terms of time generally grows as either the dimensionality or the DoE size increases, whereas performance improvement in terms of accuracy generally grows as the DoE size increases.

Regarding (ii), we note that, for simple problems (such as Examples 1 and 2), the fitting costs are negligible, and hence, improving the model accuracy (ie, reducing the MSE) while spending a few more seconds for fitting seems reasonable. In particular, adaptively controlling the ϵ parameter may result in solving the optimization more times in total than if the ϵ parameter had been set to the final value from the start with a greater number of initial guesses. Thus, the fitting cost for small problems might increase a few seconds but significant computational savings are achieved for large problems. For instance, in Example 7 with $40D$, **GPM** drastically improved the fit time from over 18 hours to under 3 hours.

To enable a better comparison, we visualize the results reported in Table 2 in Figure 5A, which shows the ratio of the mean fit times achieved by **GPM** to those achieved by **GPfit** for all the examples and training data set sizes. Since ratios smaller than 1 indicate lower computational costs, **GPM** is faster than **GPfit**. The same argument holds for Figure 5B where the ratio of the MSEs of the 2 packages are plotted.

TABLE 2 Comparison of GPM and GPfit based on mean squared error (MSE) and mean fit time: the reported data, except for Example 7 with 40*D* samples, are averaged over 20 replicates. For Example 7 with 40*D* samples, 10 replicates were used due to computational costs of **GPfit**

Example Number (Dimension)	Training Set Size	Mean GPM MSE	Mean GPfit MSE	Mean GPM Fit Time, seconds	Mean GPfit Fit Time, seconds
1 (2*D*)	10*D*	3.01E-01	2.86E-01	4.38E-01	6.74E-01
	20*D*	7.24E-02	8.31E-02	1.26E + 00	1.24E + 00
	40*D*	8.24E-06	1.00E-03	9.52E + 00	7.33E + 00
2 (3*D*)	10*D*	6.83E-08	6.96E-08	9.60E-01	1.82E + 00
	20*D*	2.03E-09	5.70E-09	4.40E + 00	4.44E + 00
	40*D*	5.02E-11	9.68E-10	2.04E + 01	1.87E + 01
3 (6*D*)	10*D*	3.03E-02	3.03E-02	7.29E + 00	3.34E + 01
	20*D*	1.34E-02	1.34E-02	1.87E + 01	1.19E + 02
	40*D*	6.22E-03	6.22E-03	7.26E + 01	4.53E + 02
4 (7*D*)	10*D*	4.36E-09	8.15E-08	2.10E + 01	6.74E + 01
	20*D*	4.50E-10	1.04E-08	8.34E + 01	2.86E + 02
	40*D*	1.17E-10	2.70E-09	5.14E + 02	1.61E + 03
5 (8*D*)	10*D*	7.19E-02	1.06E-01	3.06E + 01	1.51E + 02
	20*D*	7.81E-03	1.87E-02	1.78E + 02	6.89E + 02
	40*D*	2.38E-03	7.89E-03	8.18E + 02	3.98E + 03
6 (10*D*)	10*D*	1.90E-03	1.96E-03	6.43E + 01	6.95E + 02
	20*D*	1.31E-03	1.30E-03	2.35E + 02	3.46E + 03
	40*D*	9.18E-04	9.52E-04	1.23E + 03	1.75E + 04
7 (15*D*)	10*D*	2.17E-03	2.33E-03	2.17E + 02	3.51E + 03
	20*D*	2.80E-04	8.82E-04	4.53E + 02	1.42E + 04
	40*D*	8.43E-05	5.03E-04	2.54E + 03	6.78E + 04

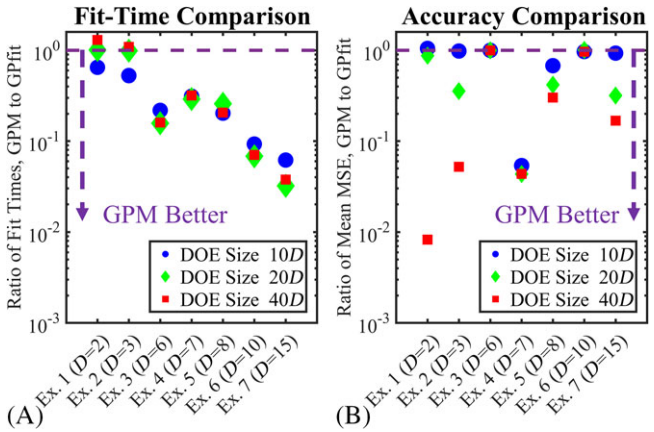

FIGURE 5 Comparison of GPM and GPfit based on mean squared error (MSE) and mean fit time for the analytical problems: the ratio of the (panel A) mean fit time and (panel B) MSE of **GPM** to **GPfit**. Any number below the dashed horizontal line indicates the superior performance of our **GPM** package. The data in panel A summarize those corresponding to columns 5 and 6 in Table 2. Similarly, panel B summarizes the data in columns 3 and 4 in Table 2 [Colour figure can be viewed at wileyonlinelibrary.com]

The adaptive control scheme for the ϵ parameter allows the fitting process to adapt to the specific training data set at hand. In general, the optimal ϵ parameter tends to decrease as the data set size increases (as long as the training points are not so close to result in numerical issues). This intuitively makes sense because, as the size of the training set increases, there is more knowledge of the underlying function and the fitted model should need less smoothing. In the case of data sets that are prone to numerical issues (regardless of the dimensionality of the problem or the size of the training data set), the required nugget can also be estimated via our adaptive approach. For example, when examining Example 1 with a training data set of size 20*D*, the typical optimal value for ϵ over the 20 fitted models via **GPM** was found to be 10^{-4}. However, for some repetitions, the optimal value was 10^{-2}, in which case at least two data points were very close to one

another, and thus, a larger constraint was needed to handle the numerical issues associated with the ill-conditioning of the correlation matrix.

In summary, we conclude from these results that **GPM** is able to consistently fit GP models considerably faster and with comparable performance for smaller DoEs and greatly improved performance for large DoEs when compared to the popular GP package **GPfit**. The time savings and better prediction performance are more noticeable as the problem dimensionality or the DoE size increases.

5 | GPM ENGINEERING APPLICATION

We now further test our approach on an engineering problem. As highlighted by the Materials Genome Initiative, material design and synthesis is one of the fastest growing disciplines with the current grand challenge of designing materials with targeted properties.[52-57] One such area of interest is to predict the properties of new materials computationally to expedite the discovery of superior materials. Here, we examine the case of learning the constitutive law of a 2*D* hyperelastic composite representative volume element, for which no closed-form constitutive law exists. Without a constitutive law, a boundary value problem must be solved for each integration (also known as Gauss or material) point of a meshed structure in the finite element method (FEM).[58] This approach can be prohibitively expensive and hence restricts the use of FEM in materials design. In our composite material system, the matrix and inclusions are modeled via the Neo-Hookean and Arruda-Boyce hyperelastic laws, respectively, and the goal is to learn the constitutive law of the composite at the microscale as a function of the applied macroscopic strains (ie, the boundary conditions). While we do not know the exact nature of the constitutive law, we do know that it is highly nonlinear and the material response can vary greatly depending on whether the material is in a state of tension or compression. In order to accurately capture the nonlinear phenomena across a wide range of boundary conditions, a large training set is needed.

In this case, the inputs for our data set are 3 boundary conditions, representing the 2 axial strains and shear strain, to a predetermined 2*D* composite microstructure. The output is a single scalar value produced by the FEM simulation corresponding to the homogenized potential density. The FEM simulation also provides the gradients of the response with respect to the 3 boundary conditions. The gradients correspond to the stress in the microstructure, which is needed to perform FE^2. The motivation behind this work is to replace the need for nested FEM simulations with a fast and efficient surrogate model to predict the material behavior (ie, gradient of the homogenized response) at the microscale.[20] Our data set consists of 1000 observations of a single microstructure with a unique set of boundary conditions for each data point. Based on our experience, a response error of less than 1% and a gradient error of less than 5% are necessary to confidently use the metamodel.

The 1000 data points were determined by the Sobol sequence.[51] Two hundred of these points were separated to serve as an independent validation data set. The remaining 800 data points were used for training. **GPM** was then used to fit models using 100 to 800 data points in increments of 100 to study the convergence of the prediction error as the size of the training data set increases. Following the algorithm in Figure 3, each data set was fitted using all the 4 correlation functions available in **GPM** (power exponential, Gaussian, lifted Brownian, and lifted Brownian with $\gamma = 1$). Since we are assuming that the response is smooth and differentiable, we would expect the exponent p and γ for the power exponential and lifted Brownian correlations (see Table 1) to be equal to 2 and 1, respectively.

To measure the prediction performance of the models, the (normalized) percent response error was then calculated on the validation data set. The percent response error is defined as

$$e_y = 100 \times \sqrt{\sum_{i=1}^{200} \frac{(y_i - \hat{y}_i)^2}{y_i^2}}\%, \tag{11}$$

where y_i and $\hat{y}_i$ are, respectively, the finite element analysis and predicted responses for the ith sample. Similarly, the percent gradient error was also calculated defined as

$$e_g = 100 \times \sqrt{\sum_{i=1}^{200} \frac{\|\nabla y_i - \nabla \hat{y}_i\|_2^2}{\|\nabla y_i\|_2^2}}\%, \tag{12}$$

where ∇y_i and $\nabla \hat{y}_i$ store the response gradients calculated via, respectively, the finite element analysis and the fitted metamodel. $||\cdot||_2$ denotes the L_2-norm of a vector.

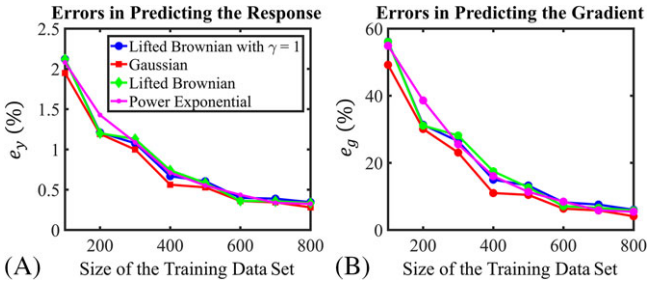

FIGURE 6 Convergence of prediction errors for the hyperelastic representative volume element example: prediction errors of (panel A) the response and (panel B) its gradients decrease as the size of the training data set increases. The 4 available correlation functions in the **GPM** package are used for training [Colour figure can be viewed at wileyonlinelibrary.com]

As suspected, the estimates of p and γ for the power exponential and lifted Brownian correlations, respectively, were 2 and 1 for all data sets. Figure 6 illustrates the percent response error and the percent gradient error as a function of the training set size and the correlation function used. A consistent improvement is seen in predicting both the response and its gradient as the training size increases across all correlation functions. In this problem, the Gaussian correlation provides the best results: with the largest training set of size 800, the model is able to achieve a response error of less than 0.5% and a gradient error of approximately 5%. From these results, we conclude that the improvements implemented in **GPM** allow users to fit larger and more accurate models that can have high-enough fidelity that the gradient of the model can accurately be predicted.

6 | CONCLUSION AND FUTURE WORK

In this work, we have presented a novel method to fit GP models. The approach, implemented in the R package **GPM**, is shown to be more computationally efficient and have better predictive accuracy than the popular **GPfit** package. Our approach enables more efficient GP modeling by leveraging the nugget parameter's effect on the log-likelihood function and employing cross-validation for estimating the best nugget value. We illustrated how **GPM** can automatically handle noisy data sets and, in the comparative study on analytical examples, demonstrated that the improved efficiency and accuracy are the most significant for models trained on large data sets and/or high dimensions. Additionally, models trained on relatively large data sets were accurate enough to predict the gradient of the response in the engineering application. This was demonstrated by improving the prediction of the stress of a hyperelastic material, which required approximating the gradient of the GP model. The applications for such an approach are wide reaching in sciences and engineering fields as data become more readily available.

ACKNOWLEDGEMENT

The authors appreciate the anonymous reviewers for their insightful comments. The research presented in this paper was supported by the National Science Foundation under grant 1537641 and the Air Force Office of Scientific Research through award FA9550-12-1-0458.

ORCID

Ramin Bostanabad https://orcid.org/0000-0001-7785-4117
Tucker Kearney http://orcid.org/0000-0002-0303-4762
Wei Chen http://orcid.org/0000-0002-4653-7124

REFERENCES

1. Sacks J, Welch WJ, Mitchell TJ, Wynn HP. Design and analysis of computer experiments. *Stat Sci*. 1989;4(4)409-423.
2. Hassaninia I, Bostanabad R, Chen W, Mohseni H. Characterization of the optical properties of turbid media by supervised learning of scattering patterns. *Sci Rep*. 2017;7(1):15259.

3. Tao S, Shintani K, Bostanabad R, et al. Enhanced Gaussian process metamodeling and collaborative optimization for vehicle suspension design optimization. Paper presented at: ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference; 2017; Cleveland, OH.
4. Jin R, Chen W, Simpson TW. Comparative studies of metamodelling techniques under multiple modelling criteria. *Struct Multidiscip Optim*. 2001;23(1):1-13.
5. Mera NS. Efficient optimization processes using kriging approximation models in electrical impedance tomography. *Int J Numer Methods Eng*. 2007;69(1):202-220.
6. Qomi MJA, Noshadravan A, Sobstyl JM, et al. Data analytics for simplifying thermal efficiency planning in cities. *J R Soc Interface*. 2016;13(117). https://doi.org/10.1098/rsif.2015.0971
7. Simpson TW, Mistree F, Korte J, Mauery T. Comparison of response surface and kriging models for multidisciplinary design optimization. 1998.
8. Simpson TW, Mauery TM, Korte J, Mistree F. Kriging models for global approximation in simulation-based multidisciplinary design optimization. *AIAA J*. 2001;39(12):2233-2241.
9. Wang GG, Shan S. Review of metamodeling techniques in support of engineering design optimization. *J Mech Des*. 2007;129(4):370-380.
10. Kennedy MC, O'Hagan A. Bayesian calibration of computer models. *J R Stat Soc: Ser B (Stat Methodol)*. 2001;63(3):425-464.
11. Farhang-Mehr A, Azarm S. Bayesian meta-modelling of engineering design simulations: a sequential approach with adaptation to irregularities in the response behaviour. *Int J Numer Methods Eng*. 2005;62(15):2104-2126.
12. Loeppky JL, Sacks J, Welch WJ. Choosing the sample size of a computer experiment: a practical guide. *Technometrics*. 2012;51(4):366-376. https://doi.org/10.1198/TECH.2009.08040
13. Shan S, Wang GG. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Struct Multidiscip Optim*. 2010;41(2):219-241.
14. Davis GJ, Morris MD. Six factors which affect the condition number of matrices associated with kriging. *Math Geol*. 1997;29(5):669-683.
15. Cressie N, Johannesson G. Fixed rank kriging for very large spatial data sets. *J R Stat Soc: Ser B (Stat Methodol)*. 2008;70(1):209-226.
16. Gramacy RB, Apley DW. Local Gaussian process approximation for large computer experiments. *J Comput Graph Stat*. 2015;24(2):561-578.
17. Gramacy RB, Lee HK. Bayesian treed Gaussian process models with an application to computer modeling. *J Am Stat Assoc*. 2012;103(483):1119-1130.
18. Kim H-M, Mallick BK, Holmes C. Analyzing nonstationary spatial data using piecewise Gaussian processes. *J Am Stat Assoc*. 2005;100(470):653-668.
19. Geers M, Yvonnet J. Multiscale modeling of microstructure-property relations. *MRS Bull*. 2016;41(08):610-616.
20. Bessa MA, Bostanabad R, Liu Z, et al. A framework for data-driven analysis of materials under uncertainty: countering the curse of dimensionality. *Comput Methods Appl Mech Eng*. 2017;320:633-667.
21. Xu HY, Li Y, Brinson C, Chen W. A descriptor-based design methodology for developing heterogeneous microstructural materials system. *J Mech Des*. 2014;136(5). https://doi.org/10.1115/1.4026649
22. Xu HY, Liu R, Choudhary A, Chen W. A machine learning-based design representation method for designing heterogeneous microstructures. *J Mech Des*. 2015;137(5). https://doi.org/10.1115/1.4029768
23. Le B, Yvonnet J, He QC. Computational homogenization of nonlinear elastic materials using neural networks. *Int J Numer Methods Eng*. 2015;104(12):1061-1084.
24. Mardia KV, Marshall R. Maximum likelihood estimation of models for residual covariance in spatial regression. *Biometrika*. 1984;71(1):135-146.
25. Currin C, Mitchell T, Morris M, Ylvisaker D. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *J Am Stat Assoc*. 1991;86(416):953-963.
26. Martin JD, Simpson TW. Use of kriging models to approximate deterministic computer models. *AIAA J*. 2005;43(4):853-863.
27. Toal DJ, Bressloff NW, Keane AJ. Kriging hyperparameter tuning strategies. *AIAA J*. 2008;46(5):1240-1252.
28. Zhao L, Choi K, Lee I. Metamodeling method using dynamic kriging for design optimization. *AIAA J*. 2011;49(9):2034-2046.
29. Audet C, Dennis JE Jr. Analysis of generalized pattern searches. *SIAM J Optim*. 2002;13(3):889-903.
30. Toal DJ, Bressloaf NW, Keane AJ, Holden CME. The development of a hybridized particle swarm for kriging hyperparameter tuning. *Eng Optim*. 2011;43(6):675-699.
31. MacDonald B, Ranjan P, Chipman H. GPfit: an R package for fitting a Gaussian process model to deterministic simulator outputs. *J Stat Softw*. 2015;64(12):1-23.
32. Cressie N. *Statistics for Spatial Data: Wiley Series in Probability and Statistics*, vol. 15. New York, NY: Wiley-Interscience; 1993:105-209.
33. Booker AJ, Dennis JE Jr, Frank PD, Serafini DB, Torczon V, Trosset MW. A rigorous framework for optimization of expensive functions by surrogates. *Struct Optim*. 1999;17(1):1-13.
34. Gramacy RB, Lee HK. Cases for the nugget in modeling computer experiments. *Stat Comput*. 2012;22(3):713-722.
35. Andrianakis I, Challenor PG. The effect of the nugget on Gaussian process emulators of computer models. *Comput Stat Data Anal*. 2012;56(12):4215-4228.
36. Pepelyshev A. The role of the nugget term in the Gaussian process method. Paper presented at: MODA 9—Advances in Model-Oriented Design and Analysis; 2010; Bertinoro, Italy.
37. Marquardt DW. An algorithm for least-squares estimation of nonlinear parameters. *J Soc Ind Appl Math*. 1963;11(2):431-441.

38. Levenberg K. A method for the solution of certain non-linear problems in least squares. *Q Appl Math*. 1944;2(2):164-168.
39. Ababou R, Bagtzoglou AC, Wood EF. On the condition number of covariance matrices in kriging, estimation, and simulation of random fields. *Math Geol*. 1994;26(1):99-133.
40. Ranjan P, Haynes R, Karsten R. A computationally stable approach to Gaussian process interpolation of deterministic computer simulation data. *Technometrics*. 2011;53(4):366-378.
41. Stein ML. *Interpolation of Spatial Data: Some Theory for Kriging*. New York, NY: Springer Science & Business Media; 2012.
42. Martin JD, Simpson TW. A study on the use of kriging models to approximate deterministic computer models. Paper presented at: ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers; 2003; Chicago, IL.
43. Kennedy MC, O'Hagan A. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*. 2000;87(1):1-13.
44. Matérn B. *Spatial Variation*. Vol 36. New York, NY: Springer Science & Business Media; 2013.
45. Plumlee M, Apley DW. Lifted Brownian kriging models. *Technometrics*. 2017;59(2):165-177.
46. Subbotin MT. On the law of frequency of error. *Mat Sb*. 1923;31(2):296-301.
47. Warnes J, Ripley B. Problems with likelihood estimation of covariance functions of spatial Gaussian processes. *Biometrika*. 1987;74(3):640-642.
48. Sasena MJ. Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations [Doctoral dissertation]. General Motors; 2002.
49. Rasmussen CE, Williams CKI. *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press;2006.
50. Viana FA, Haftka RT, Steffen V Jr. Multiple surrogates: how cross-validation errors can help us to obtain the best predictor. *Struct Multidiscip Optim*. 2009;39(4):439-457.
51. Bratley P, Fox BL. Algorithm 659: implementing Sobol's quasirandom sequence generator. *ACM Trans Math Softw*. 1988;14(1):88-100.
52. Bostanabad R, Bui AT, Xie W, Apley DW, Chen W. Stochastic microstructure characterization and reconstruction via supervised learning. *Acta Mater*. 2016;103:89-102.
53. Bostanabad R, Chen W, Apley DW. Characterization and reconstruction of 3D stochastic microstructures via supervised learning. *J Microsc*. 2016;264(3):282-297.
54. Olson GB. Computational design of hierarchically structured materials. *Science*. 1997;277(5330):1237-1242.
55. Olson GB. Designing a new material world. *Science*. 2000;288(5468):993-998.
56. Rahimi-Aghdam S, Bažant ZP, Qomi MJA. Cement hydration from hours to centuries controlled by diffusion through barrier shells of C-S-H. *J Mech Phys Solids*. 2017;99:211-224.
57. Rahimi-Aghdam S, Bažant ZP, Caner FC. Diffusion-controlled and creep-mitigated ASR damage via microplane model. II: material degradation, drying, and verification. *J Eng Mech*. 2017;143(2).
58. Feyel F, Chaboche J-L. FE^2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre SiC/Ti composite materials. *Comput Methods Appl Mech Eng*. 2000;183(3-4):309-330.
59. Dixon L, Szegö G. The global optimization problem: an introduction. *Towards Global Optimization*. Vol 2. North Holland, The Netherlands: Elsevier Science; 1978:1-15.
60. Kenett R, Zacks S. *Modern Industrial Statistics–Design and Control of Quality and Reliability, Brooks*. Pacific Grove: Cole Publishing Co.; 1998.
61. Morris MD, Mitchell TJ, Ylvisaker D. Bayesian design and analysis of computer experiments: use of derivatives in surface prediction. *Technometrics*. 1993;35(3):243-255.

How to cite this article: Bostanabad R, Kearney T, Tao S, Apley DW, Chen W. Leveraging the nugget parameter for efficient Gaussian process modeling. *Int J Numer Methods Eng*. 2017;1–16. https://doi.org/10.1002/nme.5751

APPENDIX

The full description of the example problems used in Section 4 is provided below.

1. 2*D* 6-hump camelback function with multiple local minima and maxima[48]

$$f(x) = \left(4 - 2x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(-4 + 4x_2^2\right)x_2^2$$

$$x_1 \in [-2, 2], \quad x_2 \in [-1, 1]$$

2. 3*D* problem used to simulate the engineering mechanics of a beam bending problem[45]

$$f(x) = \frac{4}{10^9} * \frac{x_1^3}{x_2 * x_3^3}$$
$$x_1 \in [10, 20], x_2 \in [1, 2], x_3 \in [0.1, 0.2]$$

3. 6*D* Hartmann function frequently used to validate optimization routines[59]

$$f(x) = -\frac{1}{1.94}\left[2.58 + \sum_{i=1}^{4} \alpha_i exp\left(-\sum_{j=1}^{6} \boldsymbol{A}_{ij}\left(x_j - \boldsymbol{P}_{ij}\right)^2\right)\right]$$

where $x_i \in [0, 1]$ for $i = 1, 2, \ldots, 6$, $\alpha = [1.0, 1.2, 3.0, 3.2]$, and

$$\boldsymbol{A} = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$$

$$\mathbf{P} = 10^{-4}\begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8723 & 5743 & 1091 & 381 \end{pmatrix}$$

4. 7*D* function used to model the cycle time for a piston[60]

$$f(x) = 2\pi\sqrt{\frac{x_1}{x_4 + x_2^2\left(\frac{x_5 x_3 x_6}{V^2 x_7}\right)}}$$

where $V = \frac{x_2}{2x_4}\left(\sqrt{A^2 + 4x_4 x_5\left(\frac{x_6}{x_7}\right)}\right)$ and $A = x_5 x_2 + 19.62x_1 - \frac{x_4 x_3}{x_2}$

$$x_1 \in [30, 60], x_2 \in [0.005, 0.020], x_3 \in [0.002, 0.010], x_4 \in [1000, 5000],$$
$$x_5 \in [90\,000, 110\,000], x_6 \in [290,296], x_7 \in [340,360]$$

5. 8*D* function used to model the flow through a borehole[61]

$$f(x) = 2\pi x_3 (x_5 - x_6)\left(\log\left(\frac{x_2}{x_1}\right)\left[1 + \frac{2x_7 x_3}{\log\left(\frac{x_2}{x_1}\right)x_1^2 x_8 + \frac{x_3}{x_4}}\right]\right)^{-1}$$
$$x_1 \in [0.05, 0.15], x_2 \in [100, 50\,000], x_3 \in [63\,070, 115\,600], x_4 \in [63.1, 116], x_5$$
$$\in [990, 1110], x_6 \in [700,820], x_7 \in [1120, 1680], x_8 \in [9855, 12\,045]$$

6. 10*D* function used to model the local potential of a nonlinear elastic composite[23]

$$f(x) = \frac{9}{2}x_9\varepsilon_m^2 + \frac{x_8 x_{10}}{1 + x_7}\left[\frac{\varepsilon_{eq}}{x_{10}}\right]^{1+x_7}$$

where $\varepsilon_m = \frac{1}{3}Tr(\boldsymbol{\varepsilon}), \varepsilon_{eq} = \sqrt{\frac{2}{3}(\varepsilon_d : \varepsilon_d)}, \varepsilon_d = \boldsymbol{\varepsilon} - \varepsilon_m\mathbf{1}, \boldsymbol{\varepsilon} = \begin{pmatrix} x_1 & x_6 & x_5 \\ x_6 & x_2 & x_4 \\ x_5 & x_4 & x_3 \end{pmatrix}, x_i \in [-10^{-3}, 10^{-3}]$ for $i = 1, 2, \ldots, 6, x_7 \in [0.2,0.8]$, $x_8 \in [2, 8], x_9 \in [15, 25], x_{10} \in [0.8,1.2.]$

7. 15*D* function used previously to test the accuracy of neural networks[23]

$$f(x) = \sqrt{\sum_{i=1}^{15} x_i^2}$$

$$x_i \in [-1, 1] \text{ for } i = 1, 2, \ldots, 15$$