arXiv:1511.02258v1 [cs.LG] 6 Nov 2015

EFFICIENT MULTISCALE GAUSSIAN PROCESS REGRESSION USING HIERARCHICAL CLUSTERING*

ZE JIA ZHANG † KARTHIK DURAISAMY ‡ NAIL A. GUMEROV §

Abstract. Standard Gaussian Process (GP) regression, a powerful machine learning tool, is computationally expensive when it is applied to large datasets, and potentially inaccurate when data points are sparsely distributed in a high-dimensional feature space. To address these challenges, a new multiscale, sparsified GP algorithm is formulated, with the goal of application to large scientific computing datasets. In this approach, the data is partitioned into clusters and the cluster centers are used to define a reduced training set, resulting in an improvement over standard GPs in terms of training and evaluation costs. Further, a hierarchical technique is used to adaptively map the local covariance representation to the underlying sparsity of the feature space, leading to improved prediction accuracy when the data distribution is highly non-uniform. A theoretical investigation of the computational complexity of the algorithm is presented. The efficacy of this method is then demonstrated on simple analytical functions and on data from a direct numerical simulation of turbulent combustion.

Key words. Gaussian process regression, multiscale methods, sparsity

AMS subject classifications. 62J, 62G

1. Introduction.

The rapid growth in computing power has resulted in the generation of massive amounts of highly-resolved datasets in many fields of science and engineering. Against this backdrop, machine learning (ML) is becoming a widely-used tool in the identification of patterns and functional relationships that has resulted in improved understanding of underlying physical phenomena [13], characterization and improvement of models [25, 11, 14], control of complex systems [5], etc. In this work, we are interested in further developing Gaussian Process (GP) regression, which is a popular supervised learning technique. While GPs have been demonstrated to provide accurate predictions of the mean and variance of functional outputs, application to large-scale datasets in high-dimensional feature spaces remains a hurdle. This is because of the high computational costs and memory requirements during the training stage and the expense of computing the mean and variance during the prediction stage. To accurately represent high-dimensional function spaces, a large number of training data points must be used.

The computational complexity of GPs at the training stage is related to the kernel inversion and the computation of the log-marginal likelihood. For example, an algorithm based on the Cholesky decomposition is one well-known approach [24]. The computational complexity of Cholesky decomposition for a problem with N training points is $O\left(N^3\right)$. For high-dimensional problems with large N, this can be prohibitive. Even if this cost can be reduced using, for instance, iterative methods [7], the computation of the predictive mean and variance at M test points will be $O\left(NM\right)$, and

*THIS IS A PREPRINT VERSION OF THE PAPER SUBMITTED TO THE SIAM JOURNAL ON SCIENTIFIC COMPUTING IN OCTOBER 2015.

†Department of Aerospace engineering, Univ of Michigan, Ann Arbor, MI 48109.

‡Department of Aerospace engineering, Univ of Michigan, Ann Arbor, MI 48109.

§Department of Aerospace engineering, Univ of Michigan, Ann Arbor, MI 48109, and Institute for Advanced Computer Studies, Univ of Maryland, College Park, MD 20742.

$O\left(N^2M\right)$, respectively. The evaluation (or testing) time could be of great significance if GP evaluations are made frequently during a computational process. Such a situation can occur in a scientific computing setting [3, 14] where GP evaluations may be performed every time-step or iteration of the solver. Reducing both the training and evaluation costs while preserving the prediction accuracy is the goal of the current work.

Much effort has been devoted to the construction of algorithms of reduced complexity. Among these is the family of methods of sparse Gaussian process regression. These methods seek to strategically reduce the size of the training set or find appropriate sparse representations of the correlation matrix through the use of induced variables. The covariance matrix, $\mathbf{\Phi} \in \mathbb{R}^{N\times N}$, contains the pairwise covariances between the N training points. Descriptions for several methods of this family are provided by Quiñonero-Candela *et al.* [15], who define induced variables $\mathbf{U} \in \mathbb{R}^{d\times P}$, $\mathbf{V} \in \mathbb{R}^P$ such that the prediction of a function $f(\mathbf{q})$ is given by

$$p(f|\mathbf{U}) \sim \mathcal{N}(\underline{\phi}_u^T \mathbf{\Phi}_u^{-1}\mathbf{V}, 1+\sigma^2 - \underline{\phi}_u^T \mathbf{\Phi}_u^{-1}\underline{\phi}_u), \tag{1}$$

where $\underline{\phi}_u \in \mathbb{R}^P$ is a vector whose elements are $\phi(\mathbf{u}_i, \mathbf{q})$, and $\mathbf{\Phi}_u(m,n) = \phi(\mathbf{u}_m, \mathbf{u}_n)$. P is the number of induced variables used in this representation, and d is the dimensionality of the inputs; i.e. $\mathbf{u}$, $\mathbf{q} \in \mathbb{R}^d$. $\mathbf{q}$ is a single test input vector, and $\mathbf{u}_m$ is the m'th column of $\mathbf{U}$. ϕ is the covariance function. From this, it is evident that if $P < N$, the matrix inversion will be less costly to perform.

The process of determining the induced variables that can optimally represent the training set can introduce additional computational costs. The most basic method is to randomly select a fraction of data points and define them as the new training set, i.e. choosing $(\mathbf{U}, \mathbf{V})$ from $(\mathbf{Q}, \mathbf{y})$, where $\mathbf{Q} \in \mathbb{R}^{d\times N}$ and $\mathbf{y} \in \mathbb{R}^N$ are the original training inputs and outputs. Seeger *et al.* [18], Smola *et al.* [19], and others have developed selection methods that provide better test results compared to random selection. These methods often depend on optimizing modified likelihoods based on $(\mathbf{U}, \mathbf{V})$, or on approximating the true log marginal likelihood [21]. Methods that introduce new induced variables instead of using a subset of the training points can involve solving optimization problems, again requiring the user to consider the additional computational burden.

In addition to choosing the inducing variables, one may also change the form of the covariance function from ϕ to ϕ', where

$$\phi'(\mathbf{q}_m, \mathbf{q}_n) = \phi(\mathbf{q}_m, \mathbf{U})\mathbf{\Phi}_u^{-1}\phi(\mathbf{q}_n, \mathbf{U})^T. \tag{2}$$

In contrast to the case where only a subset of the training data is used, all training points are considered in this approach and the matrices are only of rank P. There are many variations on this type of manipulation of the covariance. Alternatively, one can approximate $\mathbf{\Phi}$ directly by obtaining random samples of the kernel, as is described by Rahimi *et al.* [16]. However, altering the covariance matrix can cause undesirable behaviors when computing the test variance, since some matrices no longer mathematically correspond to a Gaussian process.

Other methods of accelerating GP regression focus on dividing the training set into smaller sets of more manageable size. Snelson and Ghahramani [20] employ a local GP

consisting of points in the neighborhood of the test point in addition to a simplified global estimate to improve test results. Urtasun and Darrell [23] use the local GP directly, without the induced variables. However, this requires the test points to be either assigned to a pre-computed neighborhood of training points [20], or to be determined on-line [23]. Both of these approaches can be quite expensive depending on training set size, though massive parallelization may mitigate the cost in some aspects [2] [6].

In this work, the data is partitioned into clusters based on k-center algorithms and the cluster centers are used to define a reduced training set. This leads to an improvement over standard GPs in terms of training and testing costs. This technique also adapts the covariance function to the sparsity of a given neighborhood of training points. The new technique will be referred to as Multiscale GP and abbreviated as MGP. Similar to the work of Snelson and Ghahramani [20], this flexibility may allow MGP to produce more accurate outputs by means of a more informed perspective on the input space. The next section of this paper recapitulates the main aspects of standard GP regression. Section 3 introduces the philosophy behind the proposed approach and presents the algorithm. Section 4 analyses the complexity of the MGP algorithm for both testing and training. In section 5, quantitative demonstrations are presented on a simple analytical problems as well as on data derived from numerical simulations of turbulent combustion. Following this, key contributions are summarized.

2. Preliminaries of GP regression. In this section, a brief review of standard GP regression is presented. While a detailed review can be found in the excellent book by Rasmussen [17], the description and notations below provide the necessary context for the development of the multiscale algorithm proposed in this paper.

We consider the finite dimensional weight-space view and the infinite dimensional function-space view as in Rasmussen [17]. Given a training set $\mathcal{D}$ of N observations, $\mathcal{D} = \{(\mathbf{q}_n, y_n) \mid n = 1, ..., N\}$, where $\mathbf{q}_n \in \mathbb{R}^d$ is an input vector and y is a scalar output, the goal is to obtain the predictive mean, m_*, and variance, v_*, at an arbitrary test point $\mathbf{q}_* \in \mathbb{R}^d$. Training inputs are collected in the design matrix $Q \in \mathbb{R}^{d\times N}$, and the outputs form the vector $\mathbf{y} \in \mathbb{R}^N$. Furthermore, it is assumed that

$$y = f(\mathbf{q}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2), \tag{3}$$

where ϵ is Gaussian noise with zero mean and variance σ^2.

2.1. Weight-space view. The standard linear regression model presents $f(\mathbf{q})$ in Eq. (3) as

$$f(\mathbf{q}) = \mathbf{q}^T\mathbf{w}, \tag{4}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the vector of weights. If the prior distribution of the weights is a multivariate Gaussian with zero mean and covariance matrix Σ_p, then the posterior distribution is also a multivariate Gaussian with mean $\overline{\mathbf{w}}$ and covariance matrix Σ:

$$\overline{\mathbf{w}} = \frac{1}{\sigma^2}\Sigma Q\mathbf{y}, \quad \Sigma^{-1} = \frac{1}{\sigma^2}QQ^T + \Sigma_p^{-1}. \tag{5}$$

For low dimensionality d of the inputs, linear regression cannot satisfy a variety of dependencies encountered in practice. Therefore, the input is mapped into a high-dimensional feature space $\mathbb{R}^D$, $D > d$, and the linear regression model is applied in

this space. To avoid any confusion with the original feature space $\mathbb{R}^d$, this space will be referred to as the "extended feature space" and its dimensionality can be referred to as the "size" of the system. Denoting the mapping function $\phi(\mathbf{q})$, $\phi : \mathbb{R}^d \to \mathbb{R}^D$ we have

$$f(\mathbf{q}) = \phi(\mathbf{q})^T \mathbf{w} = \sum_{j=1}^{D} w_j \phi_j(\mathbf{q}), \quad \mathbf{w} = (w_1, ..., w_D) \in \mathbb{R}^D. \tag{6}$$

The covariance matrices Σ_p and Σ have sizes $D \times D$ instead of $d \times d$. The design matrix Q should be replaced by matrix $\Phi \in \mathbb{R}^{D \times N}$, whose columns are $\phi(\mathbf{q}_n)$ instead of $\mathbf{q}_n$, $n = 1, ..., N$. The predictive mean and the covariance matrix can then be computed as

$$\overline{\mathbf{w}} = \frac{1}{\sigma^2} \Sigma \Phi \mathbf{y}, \quad \Sigma^{-1} = \frac{1}{\sigma^2} \Phi \Phi^T + \Sigma_p^{-1}. \tag{7}$$

The Bayesian formalism provides Gaussian distributions for the training outputs and for the predictive mean m_* and variance v_*.

$$\begin{aligned} \mathbf{y} \sim \mathcal{N}(0, \Sigma_l), \quad y_* \sim \mathcal{N}(m_*, v_*), \\ \Sigma_l^{-1} = \frac{1}{\sigma^2} \left(I - \frac{1}{\sigma^2} \Phi^T \Sigma \Phi \right), \\ m_* = \phi(\mathbf{q}_*)^T \overline{\mathbf{w}}, \quad v_* = \phi(\mathbf{q}_*)^T \Sigma \phi(\mathbf{q}_*) + \sigma^2, \end{aligned} \tag{8}$$

where I is the identity matrix. These expressions allow for the computation of the log-marginal likelihood (LML),

$$\log p(\mathbf{y}|Q) = -\frac{1}{2} \mathbf{y}^T \Sigma_l^{-1} \mathbf{y} - \frac{1}{2} \log |\Sigma_l| - \frac{N}{2} \log(2\pi). \tag{9}$$

Maximization of this function leads to optimal values of hyper-parameters such as σ^2 and other variables that define $\phi(\mathbf{q})$. For example, this can be the scale, h, if Gaussian functions centered at some points $\mathbf{q}'_j$ (which need not necessarily come from the training set) are selected as a basis:

$$\phi_j(\mathbf{q}) = g(\mathbf{q}, \mathbf{q}_j, h) = \exp\left(-\frac{\left\| \mathbf{q} - \mathbf{q}'_j \right\|^2}{h^2} \right), \quad j = 1, ..., D, \quad \phi = (\phi_1, ..., \phi_D)^T. \tag{10}$$

2.2. Function-space view. A Gaussian process is completely specified by the mean function $m(\mathbf{q})$ and the covariance, or kernel, function $k(\mathbf{q}, \mathbf{q}')$. The kernel function is, by definition, symmetric with respect to the arguments. The regression model described above provides

$$\begin{aligned} m(\mathbf{q}) = \mathbb{E}[f(\mathbf{q})] = \phi(\mathbf{q})^T \mathbb{E}[\mathbf{w}] = 0, \\ k(\mathbf{q}, \mathbf{q}') = \mathbb{E}[f(\mathbf{q}) f(\mathbf{q}')] = \phi(\mathbf{q})^T \mathbb{E}\left[\mathbf{w}\mathbf{w}^T\right] \phi(\mathbf{q}') = \phi(\mathbf{q})^T \mathbf{\Sigma}_p \phi(\mathbf{q}'). \end{aligned} \tag{11}$$

For finite training sets, a $N \times N$ matrix of covariances, $K_{mn} = k(\mathbf{q}_m, \mathbf{q}_n)$ can be introduced. In the case of Eq. (11), we have

$$K = \Phi^T \mathbf{\Sigma}_p \Phi. \tag{12}$$

The function-space view provides a solution of the problem in the form

$$\begin{aligned} (13) \qquad & \mathbf{y} \sim \mathcal{N}(0, K + \sigma^2 I), \quad y_* \sim \mathcal{N}(m_*, v_*), \\ & m_* = \mathbf{k}_*^T \left(K + \sigma^2 I\right)^{-1} \mathbf{y}, \quad v_* = k\left(\mathbf{q}_*, \mathbf{q}_*\right) - \mathbf{k}_*^T \left(K + \sigma^2 I\right)^{-1} \mathbf{k}_* + \sigma^2, \end{aligned}$$

where $\mathbf{k}_*$ is a column vector with entries $k\left(\mathbf{q}_n, \mathbf{q}_*\right)$, $n = 1, ..., N$. The marginal likelihood can be computed using Eq. (9), where the matrix $K + \sigma^2 I$ is used in place of Σ_l.

Note that, in the kernel approach, the predictive mean can be written in a form similar to that in the weight-space approach (Eqns. 6, 8)

$$(14) \quad m_* = f\left(\mathbf{q}_*\right) = \mathbf{k}_*^T \alpha = \sum_{n=1}^{N} \alpha_n k\left(\mathbf{q}_*, \mathbf{q}_n\right), \quad \alpha = \left(\alpha_1, ..., \alpha_N\right)^T = \left(K + \sigma^2 I\right)^{-1} \mathbf{y}.$$

The distinction between the weight- and function-space approaches has been documented [17], but some important points are noted below:

- The matrix Φ can be non-symmetric, while the covariance matrix K should be symmetric and positive semi-definite.
- The number of basis functions in Eq. (14) is N, while the number of basis functions in Eq. (6) is D.
- The relationship between K and Φ is given by Eq. (12), and the case $K = \Phi$ and $\alpha = \overline{\mathbf{w}}$ is realized only if $D = N$, $\mathbf{q}'_n = \mathbf{q}_n$, Φ is symmetric and positive definite, and the covariance matrix for the weights $\mathbf{\Sigma}_p = \Phi^{-1}$. If this is not the case, forms (8) and (14) present expansions over different bases. For example, for Gaussian functions (10), the function-space view with $\Sigma_p = \sigma_p^2 I$ provides, according to Eq. (12), the following entries for matrix $K = \sigma_p^2 \Phi^T \Phi$:

$$(15) \quad K_{mn} = \sigma_p^2 \sum_{j=1}^{D} \exp\left(-\frac{\left\|\mathbf{q}_n - \mathbf{q}'_j\right\|^2 + \left\|\mathbf{q}_m - \mathbf{q}'_j\right\|^2}{h^2}\right), \quad m, n = 1, ..., N.$$

Taking the limit of this at $\sigma_p^2 \sim 1/D$ and $D \to \infty$, we obtain a well-known result:

$$\begin{aligned} (16) \quad k\left(\mathbf{q}_m, \mathbf{q}_n\right) = K_{mn} &\sim \int_{\mathbb{R}^d} \exp\left(-\frac{\left\|\mathbf{q}_n - \mathbf{q}'\right\|^2 + \left\|\mathbf{q}_m - \mathbf{q}'\right\|^2}{h^2}\right) d\mathbf{q}' \\ &\sim \exp\left(-\frac{\left\|\mathbf{q}_m - \mathbf{q}_n\right\|^2}{2h^2}\right), \quad m, n = 1, ..., N, \end{aligned}$$

which is different from Eq. (10) because the Gaussians in kernel $k\left(\mathbf{q}, \mathbf{q}'\right)$ have larger scales, $2^{1/2}h$, than the functions $\phi_j\left(\mathbf{q}\right)$.

While the weight-space and the kernel approaches have similarities, the complexity of these algorithms can be different in practice and the optimization of the hyperparameters may also be different.

3. Multiscale sparse GP regression. As stated earlier, the goal of the proposed GP-based regression process is to decrease the complexity of both training and testing and to make the prediction more robust for datasets that have a highly irregular distribution of features.

We begin with a generalization of basis functions in Eq. (10),

(17)
$$\phi_j(\mathbf{q}) = g\left(\mathbf{q}, \mathbf{q}'_j, h_j\right) = \exp\left(-\frac{\left\|\mathbf{q}-\mathbf{q}'_j\right\|^2}{h_j^2}\right), \quad j=1,...,D, \quad \phi = (\phi_1,...,\phi_D)^T,$$

where each function is characterized not only by its center $\mathbf{q}_j$, but now also by a scale h_j. The need for such a generalization may arise from the underlying physics (e.g. particle density estimation) or from the substantial non-uniformity of the input data distribution, which could, for instance, demand smaller scales in denser regions.

We note that the matrix $\Phi\Phi^T$ (which is required for the computation of covariance matrix Σ in Eq. (7) and the kernel matrix K in Eq. (12))is given by

(18)
$$\begin{aligned}\left(\Phi\Phi^T\right)_{ij} &= \sum_{n=1}^{N} \phi_i(\mathbf{q}_n)\,\phi_j(\mathbf{q}_n) \\ &= \sum_{n=1}^{N} \exp\left(-\frac{\|\mathbf{q}_n-\mathbf{q}'_i\|^2}{h_i^2} - \frac{\left\|\mathbf{q}_n-\mathbf{q}'_j\right\|^2}{h_j^2}\right), \quad i,j=1,...,D,\end{aligned}$$

(19)
$$\begin{aligned}k(\mathbf{q}_m,\mathbf{q}_n) = K_{mn} &= \sigma_p^2 \sum_{j=1}^{D} \phi_j(\mathbf{q}_m)\,\phi_j(\mathbf{q}_n) \\ &= \sigma_p^2 \sum_{j=1}^{D} \exp\left(-\frac{\left\|\mathbf{q}_m-\mathbf{q}'_j\right\|^2 + \left\|\mathbf{q}_n-\mathbf{q}'_j\right\|^2}{h_j^2}\right), \quad m,n=1,...,N.\end{aligned}$$

3.1. Sparse representations. While N training points may be available and maximum information can be obtained when the size of the extended feature space $D=N$, we will search for subsets of these points leading to a lower size $D<N$. The size of the extended feature space is related to the accuracy of the low-rank approximation of matrix Φ built on the entire training set (*i.e.* for the case $D=N$).

Assume for a moment that we have a single scale, $h_1=...=h_N=h$. If h is chosen to be smaller than the minimum distance between the training points, the matrix Φ will have a low condition number and a low marginal likelihood. On the other end, if h is substantially larger than the maximum distance between the training points the problem will be ill-posed and the marginal likelihood will be also low. The optima should lie between these extremes. If the solution is not sensitive to the removal of a few training points, a good low-rank approximation can be sought.

The fact that the $N\times N$ matrix Φ can be well-approximated with a matrix of rank $r<N$ means that $N-r$ rows or columns of this matrix can be expressed as a linear combination of the other rows or columns with relatively small error ϵ'. Assuming that r locations (or centers) of the radial basis functions (or Gaussians) are given and

denoting such centers by $\mathbf{q}'_j$, $j = 1, ..., r$, we have

$$\phi_n(\mathbf{q}) = g(\mathbf{q}, \mathbf{q}_n, h_n) = \sum_{j=1}^{r} \eta_{nj} \phi_j(\mathbf{q}) + \epsilon'_n, \quad \mathbf{q}'_j \in \mathbb{Q}' \subset \mathbb{Q}, \quad n = 1, ..., N, \tag{20}$$

where η_{nj} are coefficients to be determined. Hence, output (Eqn. 3), where f is expanded as Eqn. 6 with $D = N$, can be written as

$$\begin{aligned} y(\mathbf{q}) &= \sum_{n=1}^{N} w_n \phi_n(\mathbf{q}) + \epsilon = \sum_{j=1}^{r} w'_j \phi_j(\mathbf{q}) + \epsilon + \epsilon', \\ w'_j &= \sum_{n=1}^{N} w_n \eta_{nj}, \quad \epsilon' = \sum_{n=1}^{N} w_n \epsilon'_n. \end{aligned} \tag{21}$$

This shows that if ϵ' is lower or comparable with noise ϵ, then a reduced basis can be used and one can simply set the size of the extended feature space to be the rank of the low-rank approximation, $D = r$, and coefficients w'_j can be determined by solving a $D \times D$ system instead of an $N \times N$ system.

3.2. Representative training subsets. Let us consider now the problem of determination of the centers of the basis functions and scales h_j. If each data-point is assigned a different scale, then the optimization problem will become unwieldy, as D hyperparameters will have to be determined. Some compromises are made, and we instead use a set of scales $h_1, ..., h_S$. In the limit of $S = 1$, we have a single scale model, while at $S = D$ we prescribe a scale to each training point. To reduce the number of scales while providing a broad spectrum of scales, we propose the use of hierarchical scales, e.g. $h_s = h_1 \beta^{s-1}$, $s = 1, ..., S$, where h_1 is of the order of the size of the computational domain and $\beta < 1$ is some dimensionless parameter controlling the scale reduction.

While there exist several randomization-based strategies to obtain a low-rank representation by choosing D representative input points [15], we propose a more regular, structured approach. For a given scale h, we can specify some distance $a = \gamma h$, where $\gamma < 1$ is chosen such that

- The distance from any input point to a basis point is less than a. Such a construction provides an approximately uniform coverage of the entire training set with d-dimensional balls of radius a.
- The number of such balls is - in some sense - optimal. Note that smaller γ results in smaller errors ϵ'_n and ϵ' and larger r in Eqs. (20) and (21). If $\gamma < h/a_{\min}$, where $a_{\min}$ is the minimal distance between the points in the training set, then $\epsilon' = 0$ and $r = D = N$, which corresponds to a full rank representation.

The problem of constructing an optimal set as described above is computationally NP-hard. However, approximate solutions can be obtained with a modification of the well-known k-means algorithm [8]. In computational geometry problems, the k-means algorithm partitions a domain in d-dimensional space into a prescribed number, k, of clusters. In the present problem, the number of clusters is unknown, but the distance parameter a is prescribed. Thus, the number of centers depends on the input point

distribution and can be determined after the algorithm is executed. Since only the cluster centers are required for the present problem, the following algorithm is used

Algorithm #1: Determination of cluster centers and k

Input: set of training points $\mathbb{Q} = \{\mathbf{q}_1, ..., \mathbf{q}_N\}$, max cluster radius a.

1. Define $\mathbb{R}_0 = \mathbb{Q}$ and set $k = 0$;
2. Do steps 3 to 6 while $\mathbb{R}_k \neq \emptyset$;
3. $k = k + 1$;
4. Assign $\mathbf{q}'_k = \mathbf{q}_j \in \mathbb{R}_{k-1}$ (where $\mathbf{q}_j$ is a random point from set $\mathbb{R}_{k-1}$)
5. Find all points $\mathbf{q}_{ki} \in \mathbb{R}_{k-1}$, such that $|\mathbf{q}_{ki} - \mathbf{q}'_k| \leqslant a$.
6. Define $\mathbb{Q}_k = \{\mathbf{q}_{ki}\}$ and $\mathbb{R}_k = \mathbb{R}_{k-1} \backslash \mathbb{Q}_k$.

Output: set of cluster centers $\mathbb{Q}' = \{\mathbf{q}'_1, ..., \mathbf{q}'_k\}$, number of clusters k.

The construction of training subsets in the case of multiple scales requires further modification of the algorithm. Starting with the coarsest scale h_1, k_1 centers can be determined using Algorithm 1 with distance parameter $a_1 = \gamma h_1$. In our approach, we select the bases such that each input point can serve as a center of only one basis function; therefore, the k_1 cluster centers of scale h_1 should be removed from the initial training set to proceed further. Next, we partition the remaining set using Algorithm 1 with distance parameter $a_2 = \gamma h_2$ and determine k_2 cluster centers. After removal of these points we repeat the process until scale h_S is reached at which we determine k_S cluster centers and stop the process. This algorithm is described below.

Algorithm #2: Determination of cluster centers in multiple scales

Input: Set of training points $\mathbb{Q} = \{\mathbf{q}_1, ..., \mathbf{q}_N\}$, set of scales $\mathbb{H} = \{h_1, ..., h_S\}$, parameter γ.

1. Define $\mathbb{R} = \mathbb{Q}$;
2. Do steps 3 to 4 for $s = 1, ..., S$;
3. Execute Algorithm #1 with input $\mathbb{R}$ and max cluster radius $a = \gamma h_s$ and get cluster centers $\mathbb{Q}'_s$ and number of clusters k_s;
4. Set $\mathbb{R} = \mathbb{R} \backslash \mathbb{Q}'_s$.

Output: set of cluster centers $\mathbb{Q}' = \cup \mathbb{Q}'_s$, $\mathbb{Q}'_s = \left\{\mathbf{q}_1^{(s)\prime}, ..., \mathbf{q}_{k_s}^{(s)\prime}\right\}$, number of clusters for each scale k_s, $s = 1, ..., S$.

Figure 1 illustrates the clustering of a 2-dimensional dataset. In principle, one can use the process even further until all the points in the training set become the cluster

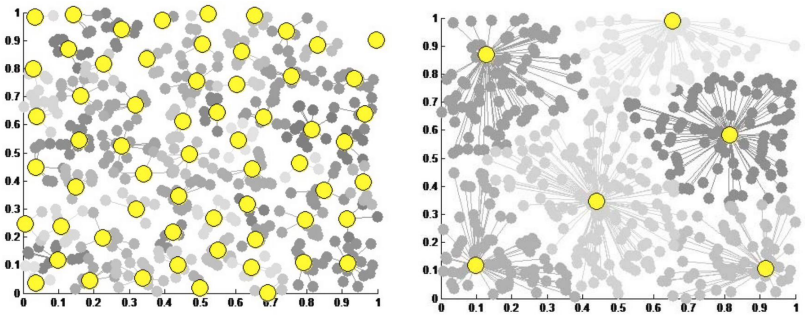


Fig. 1: The separation of 500 random points distributed inside a unit square into clusters. The cluster centers are shown in yellow; points belonging to different clusters are in different shades of gray. Left: small h_n. Right: large h_n.

centers at some scale (*i.e.* $D = N$). However, as mentioned above, the reduction of the basis, $D < N$, is important to reduce the computational complexity of the overall problem.

4. Complexity. The asymptotic complexity of algorithm #1 is $O(Nk)$ and that of algorithm #2 is $O(ND)$. While fast neighbor search methods [1] can accelerate these algorithms, the overall computational cost of the regression is still expected to be dominated by the training and evaluation costs. It is thus instructive to discuss the complexities of different algorithms available for training and testing, especially when $D \ll N$.

4.1. Training. The goal of the training stage is to determine quantities necessary for computations in the testing stage. At this point, two types of computations should be recognized: i) pre-computations, when the set of hyperparameters is given, and ii) the optimization problem to determine the hyperparameters.

To accomplish the first task, we consider two equivalent forms for computing the predictive mean and variance using Eqs. (8) and (13). The difference is that the expressions in Eq. (8) are given in terms of $D \times D$ matrices and $D \times 1$ vectors, while Eq. (13) involves $N \times N$ and $N \times 1$ arrays. The methods can be referenced as "method D" and "method N," respectively, and involve three common steps:

- The initial step is to form the matrix Φ, and then either the $D \times D$ matrix Σ^{-1} in Eq. (7) or the $N \times N$ matrix $K + \sigma^2 I$ in Eq. (12). According to Eqs. (18) and (19), the costs of this step are $O\left(ND^2\right)$ and $O\left(N^2 D\right)$, respectively. Note that these costs are much larger than the $O(ND)$ required to determine the basis function centers using the k-means type algorithm described in Section 3.2.

- The next step involves the determination of the weights $\overline{\mathbf{w}}$ from Eq. (7) or α from Eq. (14). This requires solving either a $D \times D$ or an $N \times N$ system. If solutions are obtained through direct methods, the costs are $O\left(D^3 + ND\right)$ (the cost of the solution plus the cost of the matrix-vector multiplication $\mathbf{\Phi y}$) and $O\left(N^3\right)$, respectively. Note that the former is computationally less intensive than forming the system matrix Σ^{-1}, while the latter is more expensive

than obtaining $K + \sigma^2 I$.

- Following this, the inverses Σ and $\left(K + \sigma^2 I\right)^{-1}$ should be determined[1]. The Cholesky decomposition is typically used in this situation [17];

$$\Sigma^{-1} = L_D L_D^T, \quad K + \sigma^2 I = L_N L_N^T, \tag{22}$$

  where L_D and L_N are lower triangular matrices. The decompositions can be performed with costs $O\left(D^3\right)$ and $O\left(N^3\right)$, respectively. Note that the Cholesky decompositions can also be used to solve the systems for $\overline{\mathbf{w}}$ and α, in which case the complexity of solution becomes $O\left(D^2 + ND\right)$ and $O\left(N^2\right)$, respectively.

The second task requires the computation of the log marginal likelihood. Eq. (9) is appropriate for the case when matrix $K + \sigma^2 I = \Sigma_l$ is computed and decomposed as in Eq. (22). In this case, we have

$$\log p\left(\mathbf{y}|Q\right) = -\frac{1}{2}\mathbf{y}^T\alpha - \log|L_N| - \frac{N}{2}\log\left(2\pi\right), \quad \log|L_N| = \sum_{n=1}^{N}\log L_{N,nn} \tag{23}$$

The cost to compute the first term is $O(N)$. The cost to compute the second term is also $O(N)$, since this is a sum of logarithms of the diagonal elements of L_N, denoted as $L_{N,nn}$. However, this form is not consistent with the idea of using $D \times D$ matrices, since the complexity of the Cholesky decomposition of $K + \sigma^2 I$ is $O(N^3)$ and thus dominates the overall computational expense. This problem can be resolved with the aid of the following lemma.

LEMMA 4.1. *Determinants of matrices Σ and Σ_l, defined by Eqs. (7) and (8), are related as*

$$\frac{1}{\sigma^{2N}}\frac{|\Sigma|}{|\Sigma_p|} = \frac{1}{|\Sigma_l|}. \tag{24}$$

Proof. According to the definition of Σ_l (8), we have in the right hand side of Eq. (24)

$$\frac{1}{|\Sigma_l|} = \left|\Sigma_l^{-1}\right| = \frac{1}{\sigma^{2N}}\left|I - \frac{1}{\sigma^2}\Phi^T\Sigma\Phi\right|. \tag{25}$$

Note now the Sylvester determinant theorem, which states that

$$|I_N + AB| = |I_D + BA|, \tag{26}$$

where A is an $N \times D$ matrix and B is $D \times N$, while I_N and I_D are the $N \times N$ and $D \times D$ identity matrices. Thus, we have in our case

$$\left|I - \frac{1}{\sigma^2}\Phi^T\Sigma\Phi\right| = \left|I_N - \frac{1}{\sigma^2}\Phi^T\left(\Sigma\Phi\right)\right| = \left|I_D - \frac{1}{\sigma^2}\left(\Sigma\Phi\right)\Phi^T\right| = \left|I - \frac{1}{\sigma^2}\Sigma\Phi\Phi^T\right|. \tag{27}$$

[1] Alternatively, efficient decompositions enabling solutions with multiple right hand sides can be used.

From Eq. (7) we have

$$I = \Sigma\Sigma^{-1} = \Sigma\left(\frac{1}{\sigma^2}\Phi\Phi^T + \Sigma_p^{-1}\right), \tag{28}$$

$$\left|I - \frac{1}{\sigma^2}\Sigma\Phi\Phi^T\right| = \left|\Sigma\Sigma_p^{-1}\right| = \frac{|\Sigma|}{|\Sigma_p|}.$$

Combining results (25), (27), and (28), one can see that Eq. (24) holds. □

Now, replacing $\Sigma_l^{-1}\mathbf{y}$ in the first term in the right-hand side of Eq. (9) with expressions for Σ_l^{-1} and $\overline{\mathbf{w}}$ from Eqs. (7) and (8) and using Eq. (24) in the second term, we obtain

$$\log p\left(\mathbf{y}|X\right) = -\frac{1}{2\sigma^2}\left(\mathbf{y} - \Phi^T\overline{\mathbf{w}}\right)^T\mathbf{y} - \frac{1}{2}\log\left|\Sigma^{-1}\right| - \frac{1}{2}\log|\Sigma_p| - \frac{N}{2}\log\left(2\pi\sigma^2\right). \tag{29}$$

In the case of $\Sigma_p = \sigma_p^2 I$ and using the Cholesky decomposition of Σ^{-1}, we obtain

$$\log p\left(\mathbf{y}|X\right) = -\frac{1}{2\sigma^2}\left(\mathbf{y} - \Phi^T\overline{\mathbf{w}}\right)^T\mathbf{y} - \sum_{j=1}^{D}\log L_{D,jj} - \frac{N}{2}\log\left(2\pi\sigma_p^2\sigma^2\right). \tag{30}$$

Here, the cost of computing the first term on the right hand side is $O\left(ND\right)$, and the cost of the second term is $O\left(D\right)$.

Multi-dimensional optimization is usually an iterative process. Assuming that the number of iterations for both methods of computation is N_{iter}, we can write the costs of the training steps marked by the respective superscripts as follows

$$C_{train}^{(D)} = O\left(N_{iter}D\left(N + D^2\right)\right), \quad C_{train}^{(N)} = O\left(N_{iter}N^3\right). \tag{31}$$

This shows that

$$C_{train}^{(N)}/C_{train}^{(D)} = \begin{cases} O\left(\left(N/D\right)^2\right), D^2 \ll N \\ O\left(\left(N/D\right)^3\right), D^2 \gtrsim N \end{cases}, \tag{32}$$

i.e. "method D" is asymptotically speaking much faster than "method N."Particularly, in the case of substantial oversampling, D does not depend on N, and at large N "method D" scales linearly with N whereas "method N" scales as N^3.

4.2. Testing. The cost of testing for methods D and N can be estimated from Eq. (8) and Eq. (13) as the costs of computing the predictive mean and variance. Taking into account the Cholesky decomposition (22), these costs can be written as

$$C_{mean}^{(D)} = O\left(MD\right), \quad C_{mean}^{(N)} = O\left(MND\right),$$
$$C_{var}^{(D)} = O\left(MD^2\right), \quad C_{var}^{(N)} = O\left(MN^2\right), \tag{33}$$

where M is the number of test points. Here, the relatively high cost $C_{mean}^{(N)}$ is determined by the $O\left(ND\right)$ cost of computing the vector $\mathbf{k}_*$, a vector of length N where each component is computed as a sum of D terms according to Eq. (19). In contrast, cost $C_{mean}^{(D)}$ is very low, as it is determined by the cost of the dot product of vectors of length D per evaluation point. The costs for computing variances are much higher than the costs for computing means, and in both methods they are determined by the cost of solving triangular systems of size $D \times D$ and $N \times N$ *per* evaluation point. Again, it is seen that method D is much faster than method N.

5. Numerical Results. In this section, a set of simple analytical examples are first formulated to critically evaluate the accuracy and effectiveness of MGP and to contrast its performance with conventional GP regression. Following this, data from a large turbulent combustion simulation is used to assess the viability of the approach in scientific computing problems.

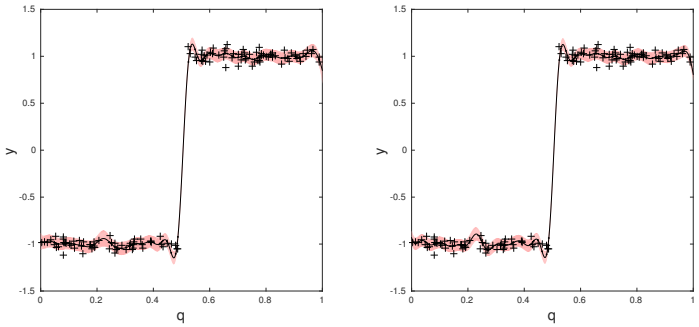


Fig. 2: Comparison of the output of the standard kernel GP using a Gaussian kernel (left) and MGP with $S = 1$ (right) for a step function. The crosses are training points ($N = 128$), while the continuous lines and shaded areas show the predictive mean and the variance.

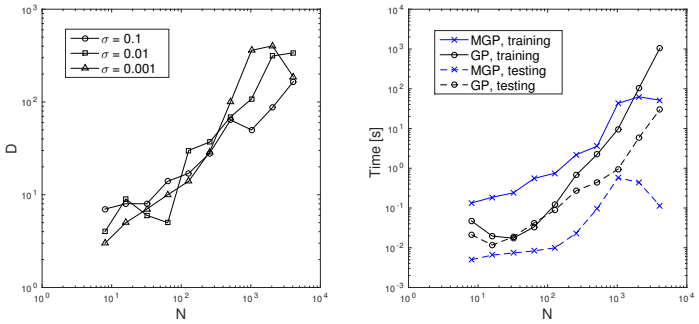


Fig. 3: Comparison of the performance of the standard GP and MGP for the step function in Figure 2. The plot on the left shows the size of the extended feature space for different input noise σ. The wall clock time is obtained using MATLAB on a typical desktop PC.

5.1. Analytical examples. The first numerical example we present is a 1-D step function ($d = 1$). We compare the standard kernel GP with a Gaussian kernel and the present MGP algorithm restricted to a single scale ($S = 1$). The training set of size N was randomly selected from 10000 points distributed in a uniform grid. Points not used for training were used as test points. The initial data was generated by adding normally distributed noise with standard deviation σ to the step function. Optimal hyperparameters were found using the Lagarias algorithm for the Nelder-Mead method (as implemented by MATLAB's `fminsearch` function) [10]. For the standard GP, σ and h were optimized, while for the MGP algorithm, the optimal γ was determined in addition to σ and h. In both cases, the optimal σ was close to the actual

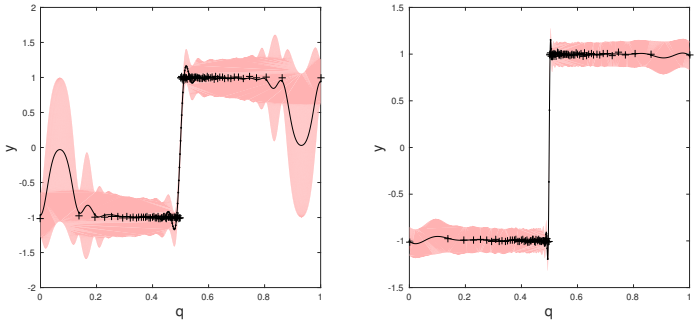


Fig. 4: Comparison of the standard GP (left) and MGP (right) for a step function. In contrast to Figure 2, the training point distribution is non-uniform with an exponential density increase near the jump. The number of training points $N = 101$. Gaussian noise with $\sigma = 0.03$ was added. The crosses represent training points, while the continuous lines and shaded areas show the predictive mean and the variance. The dots on the continuous line have abscissas of the training points.

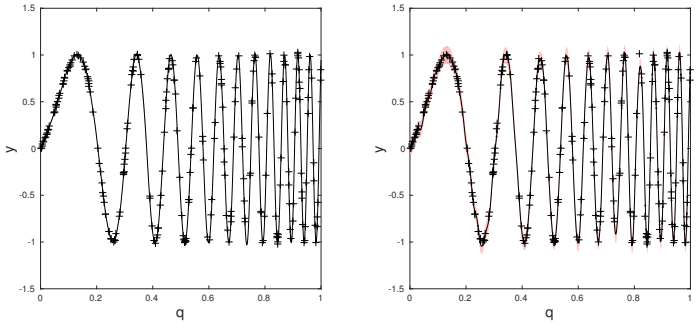


Fig. 5: Comparison of performance of the standard GP (left) and MGP with $S = 1$ (right) for a sine function, $y = \sin(2\pi q(4q + 1)^{1.5})$ with Gaussian noise of standard deviation $\sigma = 0.01$. Crosses represent training points ($N = 128$); continuous lines and shaded regions show the predictive mean and the variance.

σ used for data generation. The ratio between the optimal h for the kernel and the finite-dimensional ("weight-space") approaches was found to be approximately 1.4, which is consistent with the difference of $\sqrt{2}$ predicted by the theory (see discussion after Eq. (16)).

The plots in Figure 2 show that there is no substantial difference in the mean and the variance computed using both methods, while the sparse algorithm required only $D = 38$ functions compared to the $N = 128$ required for the standard method. Figure 3 shows the relationship between the extended feature space (D) with respect to the size of the training set (N). Ideally, this should be a linear dependence, since the step function is scale-independent. Due to random noise and the possibility that the optimization may converge to local minima of the objective function, however, the relationship is not exactly linear. Since D is nevertheless several times smaller than N, the wall-clock time for the present algorithm is shorter than that for the standard

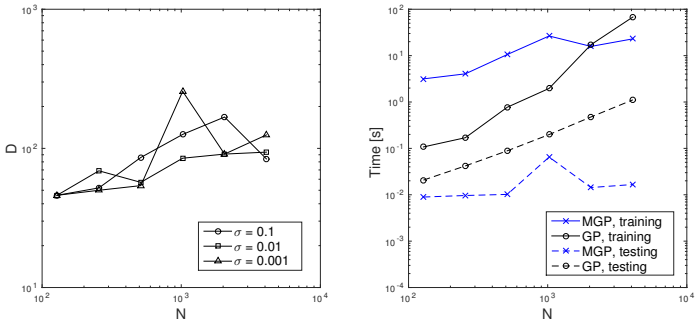


Fig. 6: Comparison of the performance of the standard GP and MGP for the sine function in Figure 5. The wall-clock time is obtained using MATLAB on a typical desktop PC.

GP in both testing and training, per optimizer iteration. The total training time for the present algorithm can be larger than that of the standard algorithm due to the overhead associated with the larger number of hyperparameters and the resultant increase in the number of optimization iterations required. However, we observed this only for relatively small values of N. For larger N, such as $N = 4096$, the present algorithm was approximately 5, 10, and 20 times faster than the standard algorithm for $\sigma = 0.1$, 0.01, and 0.001, respectively.

Figure 4 illustrates a case where the training points are distributed non-uniformly. Such situations frequently appear in practical problems, where regions of high functional gradients are sampled with higher density to provide a good representation of the function. For example, adaptive meshes to capture phenomena such as shockwaves and boundary layers in fluid flow fall in this category. In the case illustrated here, the training points were distributed with exponential density near the jump ($q = 0.5 \pm h_t \ln z$, $\exp(-0.5/h_t) \leqslant z \leqslant 1$, where z are distributed uniformly at the nodes of a regular grid. $h_t = 0.1$ was used). For MGP, the number of scales was $S = 6$ and the other hyperparameters were optimized using the same routine as before. Due to multiple extrema in the objective function, it is rather difficult to optimize the number of scales, S. In practice, one should start from several initial guesses or fix some parameters such as S. We used several values of S and observed almost no differences for $5 \leq S \leq 10$, while results for $S = 1$ and $S = 2$ were substantially different from the cases where $S > 2$.

It is seen that the MGP provides a much better fit of the step function in this case than the standard method. This is achieved due to its broad spectrum of scales. In the present example, we obtained the following optimal parameters for scales distributed as a geometric progression, $h_s = h_1\beta^{s-1}$: $h_{\max} = h_1 = 0.1233$, $h_{\min} = h_6 = 0.0032$, and $\beta = 0.4806$. Other optimal parameters were $\gamma = 0.258$ and $\sigma = 0.127$. For the standard GP, the optimal scale was $h = 0.0333$. Figure 4 shows that with only a single intermediate scale, it is impossible to approximate the function between training points with a large spacing, whereas MGP provides a much better approximation. Moreover, since $h_{\min} < h$, we also have a better approximation of the jump, i.e. of small-scale behavior. This is clearly visible in the figure; the jump for the standard GP is stretched over about 10 intervals between sampling points, while the jump for

MGP only extends over 3 intervals. Note that in the present example, we obtained $D = N = 101$, so the wall-clock time for testing is not faster than the standard GP. However, this case illustrates that multiple scales can provide good results for substantially non-uniform distributions where one scale description is not sufficient.

As final analytical example, we explore a sine wave with varying frequency, depicted in Figure 5. As before, N training points are randomly chosen from a uniform distribution, and $M = 10000 - N$ test points are used. For MGP, $S = 1$ was used. Compared to the previous examples, this is an intermediate case in terms of scale dependence. One noteworthy result from this dataset is that D is almost constant with respect to N, as seen in Figure 6. This shows that when the function is relatively smooth, the optimization process is not limited to producing a linear relationship between D and N. Another result is that the output variance of the multiscale method is visibly higher than that of the standard method. For the previous cases, the variances have been either been close to equal, or the standard method would produce the higher variance. This could be due to the fact that h and D are inherently related for the current method, whereas h is unrestricted for the standard GP. Since $D < N$, the multiscale h for $S = 1$ is typically greater than the standard method's h. According to Eqs. (13) and (16), this would result in greater variance.

5.2. Data from Turbulent Combustion. Combustion in the presence of turbulent flow involves an enormous disparity in time and length scales, meaning that direct numerical simulations (DNS) are not possible in practical problems. Large eddy simulations (LES), in which the effect of scales smaller than the mesh resolution (*i.e.* subgrid scales) are modeled, is often a pragmatic choice. A key difficulty in LES of combustion is the modeling of the subgrid scale fluxes in the species transport equations [12, 22]. These terms arise as a result of the low-pass filtering - represented by the operator $\bar{(\cdot)}$ - of the governing equations, and are of the form

$$f_k = \overline{\rho u_k C} - \frac{\overline{\rho u_k}\,\overline{\rho C}}{\bar{\rho}} \tag{34}$$

where ρ, u, C represent density, velocity and species mass fraction, respectively. Subgrid-scale closures based on concepts from non-reacting flows, such as the equation below,[2]

$$f_k = -\frac{\overline{\rho} C_s^2 \Delta^2 \sqrt{2\tilde{S}_{ij}\tilde{S}_{ij}}}{Sc} \frac{\partial \tilde{c}}{\partial x_k} \tag{35}$$

are found to be inadequate for turbulent combustion. Modeling of the scalar fluxes thus continues to be an active area of research, and many analytical models are being evaluated by the community. Reference [4] provides a concise summary of such developments in the area of premixed turbulent flames.

In this work, we intend to apply GP and MGP to identify the model relationships from data. The simulation data is obtained from Ref. [9], in which DNS of a propagating turbulent flame is performed. The configuration involves a premixed hydrogen-air flame propagating in the x-direction. The flame forms a thin front between the burnt

[2] C_s, Sc are typically constants, and $S_{ij} = \frac{1}{2}\left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right]$ is the strain-rate tensor. The superscript $\tilde{(\cdot)}$ denotes Favre-filtering and is defined by $\tilde{q} = \frac{\overline{\rho q}}{\bar{\rho}}$. Δ is the filter size.

Table 1: Numerical results for learning F_1 in the flame. Error is defined as $||\mathbf{y} - f(\mathbf{Q}_*)||/||\mathbf{y}||$, where $\mathbf{y}$ is a vector of the exact outputs F_1, and $f(\mathbf{Q}_*)$ is a vector of the GP or MGP outputs at the test points $\mathbf{Q}_*$. Time is test time in seconds, obtained in MATLAB.

	GP		MGP	
	Error	Time	Error	Time
F_1	0.1439	0.94	0.1566	9.9×10^{-2}

and unburnt gases. Specifically, we attempt to learn the normalized flux in the x-direction, F_1, as a function of seven variables:

$$F_1 = \frac{f_1}{\overline{\rho}\Delta^2} = f(\nabla\tilde{u}, \nabla\tilde{c}, \tilde{S}) \tag{36}$$

A total of 3 million training points were generated from the dataset by performing low-pass filtering. Some care was taken in choosing training points. Since the flame is thin relative to the size of the domain, the majority of the data points were found to lie outside the region where F_1 is nonzero. To mitigate this disadvantageous distribution, 80 percent of the training points were randomly chosen from the data with $f_1 > 0.05$, and 20 percent were chosen from data with $f_1 \leq 0.05$, i.e. outside the flame. About 2500 training points were used in total. For testing, a single $x - y$ plane of around 6500 points was set aside.

The predictive results on this plane are shown in Table 1, and Figure 8 shows the ML output versus the true DNS values. From these plots, it is seen that MGP has achieved a ten-fold increase in evaluation speed for a corresponding two percent increase in error. In the scatter plot, MGP appears to perform better than GP for low values of F_1 and worse for high values, but the overall difference is small. Figure 7 is a side-by-side comparison of contours of F_1 from DNS and from GP and MGP. It is especially evident in the contour plot that GP and MGP are both able to capture features of the flame, whereas analytical models described in Ref. [4] were not as accurate[3]. Figure 9 plots F_1 along several locations perpendicular (x) and parallel (y) to the flame in the test plane.

[3]These results are not shown

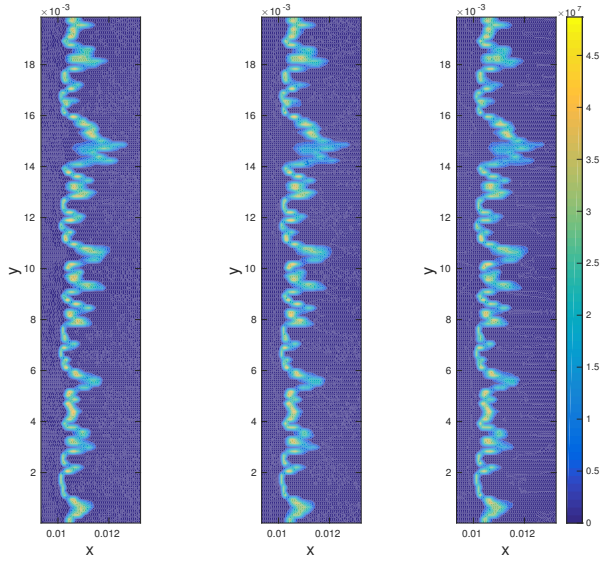


Fig. 7: Contours of F_1 on the test $x - y$ plane. Left: DNS values (exact). Center: GP output. Right: MGP output.

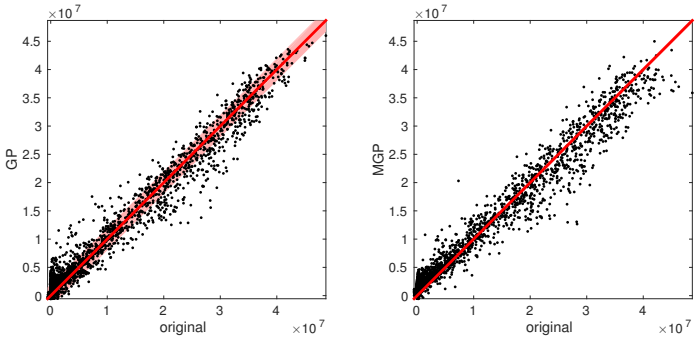
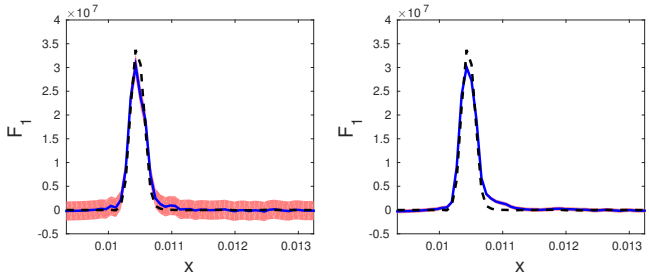
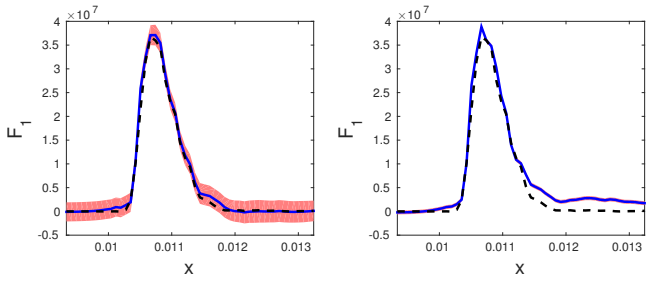


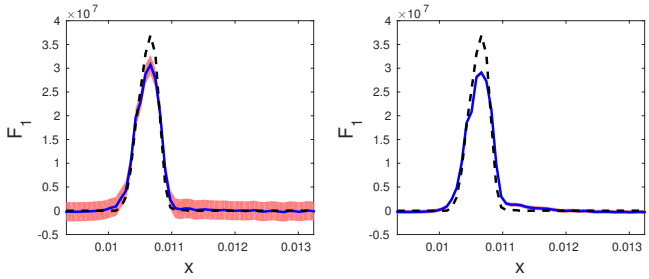
Fig. 8: Original test output versus learned result for F_1. Red regions are $\pm\sigma$ from the diagonal, i.e. the optimized estimate of the input noise.



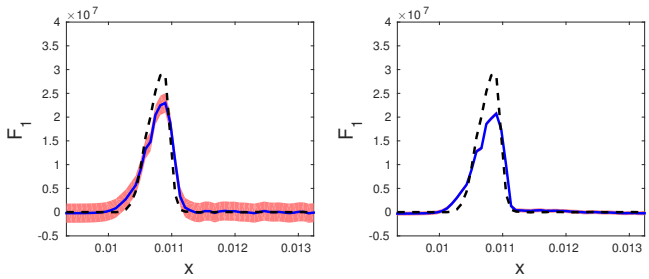
(a) $y = 2 \times 10^{-5}$



(b) $y = 8 \times 10^{-4}$



(c) $y = 5 \times 10^{-3}$



(d) $y = 1 \times 10^{-2}$

Fig. 9: F_1 as a function of x for four different constant values of y on the test $x - y$ plane. Dashed black lines are from DNS. GP results on the left, MGP results on the right. Red regions are one standard deviation for the output.

6. Conclusion. In this paper, a new Gaussian process (GP) regression technique was presented. The method, referred to as MGP, introduces multiple scales among the Gaussian basis functions and employs hierarchical clustering to select centers for these sparse basis functions. These modifications reduce the computational complexity of GP regression and also achieve better accuracy than standard GP regression when training points are non-uniformly distributed in the d-dimensional feature space. We illustrated these improvements through analytical examples and in a turbulent combustion datasets.

Based on the results presented in this work, MGP is a potentially attractive method for use in many scientific computing applications in which datasets may be large, and sparsely distributed in a high-dimensional feature space. The MGP can be especially useful when predictive evaluations are performed frequently. However, further developments and more detailed application studies are warranted. It was observed that the optimization process is more likely to terminate at local minima compared to conventional GPs. An immediate area of investigation could explore more efficient and robust techniques for optimization of the hyperparameters. Since an appealing feature of MGP is the reduced complexity when working with large datasets, efficient parallelization strategies should be explored.

The software and all the examples in this paper are openly available at `http://umich.edu/~caslab/#software`.

Acknowledgments. This work was performed under the NASA LEARN grant titled "A Framework for Turbulence Modeling Using Big Data."

REFERENCES

[1] D. Cheng, A. Gersho, B. Ramamurthi, and Y. Shoham, *Fast search algorithms for vector quantization and pattern matching*, in Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84., vol. 9, IEEE, 1984, pp. 372–375.

[2] A. Choudhury, P. B. Nair, and A. J. Keane, *A data parallel approach for large-scale gaussian process modeling.*, in SDM, SIAM, 2002, pp. 95–111.

[3] K. Duraisamy, Z. Zhang, and A. Singh, *New approaches in turbulence and transition modeling using data-driven techniques*, (2014).

[4] Y. Gao, M. Klein, and N. Chakraborty, *Assessment of sub-grid scalar flux modelling in premixed flames for large eddy simulations: A-priori direct numerical simulation analysis*, European Journal of Mechanics - B/Fluids, 52 (2015), pp. 97 – 108.

[5] N. Gautier, J. Aider, T. Duriez, B. R. Noack, M. Segond, and M. Abel, *Closed-loop separation control using machine learning*, Journal of Fluid Mechanics, 770 (2015), pp. 442–457.

[6] R. B. Gramacy, J. Niemi, and R. M. Weiss, *Massively parallel approximate gaussian process regression*, SIAM/ASA Journal on Uncertainty Quantification, 2 (2014), pp. 564–584.

[7] N. A. Gumerov and R. Duraiswami, *Fast radial basis function interpolation via preconditioned krylov iteration*, SIAM Journal on Scientific Computing, 29 (2007), pp. 1876–1899.

[8] J. A. Hartigan and M. A. Wong, *Algorithm AS 136: A k-means clustering algorithm*, Applied statistics, (1979), pp. 100–108.

[9] M. Hassanaly, V. Raman, H. Koo, and M. B. Colkett, *Influence of fuel stratification on turbulent flame propagation*, (2014).

[10] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, *Convergence properties of the nelder–mead simplex method in low dimensions*, SIAM Journal on optimization, 9 (1998), pp. 112–147.

[11] J. Ling and J. Templeton, *Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty*, Physics of Fluids, 27 (2015).

[12] A. N. Lipatnikov and J. Chomiak, *Effects of premixed flames on turbulence and turbulent scalar transport*, Progress in Energy and Combustion Science, 36 (2010), pp. 1–102.

[13] E. Mjolsness and D. DeCoste, *Machine learning for science: state of the art and future prospects*, Science, 293 (2001), pp. 2051–2055.
[14] E.J. Parish and K. Duraisamy, *A paradigm for data-driven predictive modeling using field inversion and machine learning*, Journal of Computational Physics, Accepted for publication, (2015).
[15] J. Quiñonero-Candela and C. E. Rasmussen, *A unifying view of sparse approximate gaussian process regression*, The Journal of Machine Learning Research, 6 (2005), pp. 1939–1959.
[16] A. Rahimi and B. Recht, *Random features for large-scale kernel machines*, in Advances in neural information processing systems, 2007, pp. 1177–1184.
[17] C. E. Rasmussen, *Gaussian processes for machine learning*, (2006).
[18] M. Seeger, C. Williams, and N. Lawrence, *Fast forward selection to speed up sparse gaussian process regression*, in Artificial Intelligence and Statistics 9, no. EPFL-CONF-161318, 2003.
[19] A. J. Smola and P. Bartlett, *Sparse greedy gaussian process regression*, in Advances in Neural Information Processing Systems 13, MIT Press, 2001, pp. 619–625.
[20] E. Snelson and Z. Ghahramani, *Local and global sparse gaussian process approximations*, in International Conference on Artificial Intelligence and Statistics, 2007, pp. 524–531.
[21] M. K. Titsias, *Variational learning of inducing variables in sparse gaussian processes*, in International Conference on Artificial Intelligence and Statistics, 2009, pp. 567–574.
[22] S. Tullis and R. S. Cant, *Scalar transport modeling in large eddy simulation of turbulent premixed flames*, Proceedings of the Combustion Institute, 29 (2002), pp. 2097–2104.
[23] R. Urtasun and T. Darrell, *Sparse probabilistic regression for activity-independent human pose inference*, in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.
[24] J.H. Wilkinson, *The algebraic eigenvalue problem*, vol. 87, Clarendon Press Oxford, 1965.
[25] T. Yairi, Y. Kawahara, R. Fujimaki, Y. Sato, and K. Machida, *Telemetry-mining: a machine learning approach to anomaly detection and fault diagnosis for space systems*, in Space Mission Challenges for Information Technology, 2006. SMC-IT 2006. Second IEEE International Conference on, IEEE, 1992, pp. 8–pp.