


# Sequential sampling for functional estimation via SIEVE

Alessia Benevento<sup>1</sup>  | Pouya Ahadi<sup>2</sup>  | Swati Gupta<sup>3</sup> | Massimo Pacella<sup>4</sup> | Kamran Paynabar<sup>2</sup> 

<sup>1</sup>Department of Economics, University of Salento, Lecce, Italy

<sup>2</sup>H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA

<sup>3</sup>Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

<sup>4</sup>Department of Engineering for Innovation, University of Salento, Lecce, Italy

## Correspondence

Kamran Paynabar, H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA.  
Email: [kamran.paynabar@isye.gatech.edu](mailto:kamran.paynabar@isye.gatech.edu)

## Funding information

Ministry of University and Research of Italy's; National Operational Program for Research and Innovation; National Science Foundation, Grant/Award Number: 2239824

## Abstract

Sequential sampling methods are often used to estimate functions describing models subjected to time-intensive simulations or expensive experiments. These methods provide guidelines for point selection in the domain to capture maximum information about the function. However, in most sequential sampling methods, determining a new point is a time-consuming process. In this paper, we propose a new method, named SIEVE, to sequentially select points of an initially unknown function based on the definition of proper intervals. In contrast with existing methods, SIEVE does not involve function estimation at each iteration. Therefore, it presents a greater computational efficiency for achieving a given accuracy in estimation. SIEVE brings in tools from computational geometry to subdivide regions of the domain efficiently. Further, we validate our proposed method through numerical simulations and two case studies on the calibration of internal combustion engines and the optimal exploration of an unknown environment by a mobile robot.

## KEYWORDS

adaptive sampling, Delaunay triangulation, Gaussian Process

## 1 | INTRODUCTION

Many present-day engineering problems involve the exploration of an unknown space for estimating a function, finding the optimum of the function in the space, and/or creating surrogate models that mimic the behavior of real-world systems. The surrogate models are typically used to analyze, monitor, and/or optimize complex physical or simulation systems.<sup>1–3</sup> For example, surrogate models using Gaussian Processes (GPs) are utilized to estimate the actual free-form surface obtained from a machining process during quality inspection.<sup>4–6</sup> The estimation of free-form surfaces can be used to test and optimize the dynamic accuracy of CNC machines.<sup>7,8</sup> In another example, in automotive manufacturing applications, to estimate the optimal control map during the calibration of an internal combustion engine, a GP surrogate model is used to model air mass flow (mg/stoke) as a function of engine rotational speed (rpm) and engine torque in Newton-meter.

Both the size of the training sample and the location of observations in the sampling space are important factors that impact the accuracy of a surrogate model. However, in practice, due to the high cost and time-consuming nature of

This is an open access article under the terms of the [Creative Commons Attribution-NonCommercial-NoDerivs](https://creativecommons.org/licenses/by-nc-nd/4.0/) License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2024 The Authors. *Quality and Reliability Engineering International* published by John Wiley & Sons Ltd.

experiments, the number of training observations is limited. For example, engine calibration is one of the most costly steps in the engine development process. The engine test bench is an expensive system to set up, and experimental measurements on the test bench are often time-intensive.<sup>9–11</sup> For example, in diesel engine performance optimization, in-cylinder pressure plays an important role, and performance optimization is challenging due to the costly and time taking physical tests.<sup>12</sup> Additionally, the high computational complexity of GP-based surrogate models (i.e.,  $O(n^3)$ , with  $n$  as the training sample size) is another limiting factor to large-scale experiments. These limitations prompt systematic sequential sampling methods for achieving an optimal result with limited sample size. Examples of such sampling methods in engineering applications include the air quality sensing,<sup>13</sup> the calibration of computer models,<sup>14,15</sup> aircraft aerodynamic design,<sup>16</sup> quality inspection<sup>7</sup> and fault detection in manufacturing,<sup>17</sup> and engine calibration.<sup>18</sup>

The selection of sampling points holds significant importance in functional estimation, and the utilization of space-filling design methods will be crucial.<sup>19</sup> In general, two types of strategies can be used to select the sampling points in the design space, namely *one-shot sampling* methods (also known as domain-based, model-free, and nonadaptive) and *sequential sampling* methods (also known as response-based). In the *one-shot sampling*, using an optimization criterion, for example, maximum distance,<sup>20,21</sup> a fixed number of sampling points are chosen a priori based on the available budget, and all the observed points are used to estimate the surrogate model. Two examples of one-shot sampling methods are (i) the Latin Hypercube designs and (ii) the simplex stochastic collocation method.<sup>22–24</sup> On the other hand, in the *sequential sampling* design, the new sampled points are selected one-by-one based on the information obtained from previous observations. At each step, the collected samples are fused to update the surrogate model, and its uncertainty is used as a measure for selecting the location of the subsequent observations. Examples of the sequential sampling methods include Refs. 25–27 that rely on the Expected Improvement criterion,<sup>28</sup> which again utilizes the expected improvement criterion for contour estimation,<sup>29</sup> which utilizes maximin criterion to perform sequential Latin Hypercube Design,<sup>30,31</sup> which is based on the Minimum Energy Design criterion, and online Kriging model-assisted methods that focus on sequential updating of an ad hoc objective function.<sup>32,33</sup>

As mentioned earlier, one-shot methods do not take advantage of information obtained by previous observations, which may result in inconsistent uncertainty of the surrogate model depending on the true underlying function to be estimated. Although the sequential sampling methods overcome this problem by iteratively updating the surrogate model based on the most recent observations, these methods could be quite slow, especially when the number of observations is large, as they require re-estimation of the surrogate model at each step. In some applications, such as the case studies in Section 6, computational efficiency is vital in the exploration process.

Our first case study is the engine calibration process discussed earlier, where the goal is to find optimal control parameters for each operating condition.<sup>34–36</sup> The control parameters of the actuators are often obtained through several trial-and-error experiments. Although the existing sequential sampling methods facilitate the calibration for one operating condition without requiring to collect data for all the operating conditions, their slow computation increases the engine idle time when running on the test bench. Consequently, costly resources such as labor, fuel, and time are wasted during the calibration process.

The second case study focuses on the optimal exploration of an unknown environment by a mobile robot. The problem of placing a mobile robot in a space such that it can optimally explore and monitor an environment and capture its essence finds applications in the autonomous environmental monitoring,<sup>37,38</sup> autonomous search-and-rescue missions,<sup>39,40</sup> and exploration of hazardous environment.<sup>41,42</sup> For example, it is vital to quickly and optimally cover a contaminated environment to estimate the spatial density of hazardous material. Initially, the environment is unknown, and the robot is guided in exploring it by a sequential sampling strategy. However, efficiency is vital in such applications because the computational power and robot battery are limited resources. Additionally, these missions are time-sensitive, and a quick exploration of the environment and estimation of the density is critical to the success of the mission.

To address the foregoing issues, in this paper, we propose a novel sequential sampling approach, named Sequential sampling algorithm with Interval-based Exploration and Value Estimation (SIEVE) that utilizes the information of all previous observations in the selection process, it does not require re-estimation of the surrogate model at each step. This makes the sampling approach efficient and capable of implementation in online settings. Unlike existing methods, SIEVE takes both the positions of the observed points in the domain (similarly to one-shot sampling) and the values of the function at those points (similarly to sequential sampling) into account using a two-part objective function: The first part is domain-based, and geometrically partitions the domain by using the positions of the sampled points, and the second part uses the observed function values to select the region with the highest benefit. SIEVE explores the most untouched areas of the domain and simultaneously uses the information of the observed points to choose the locations of the following samples. Moreover, it is based on observed function values rather than the estimated model of the function.

In other words, the difference between the SIEVE method and one-shot sampling methods is that once each sample is achieved, the SIEVE utilizes the observed function value and determines the next sampling point based on the whole information. On the other hand, the significant difference between sequential sampling approaches and the SIEVE is that after observing the function value for a sample, sequential sampling methods require fitting a new model, which will lead us to determine the next point to sample. However, in the SIEVE, we update the model once at the end of the sampling procedure, and hence the sampling is independent of any model estimation. This significantly reduces the computational complexity.

It is worth mentioning that if the experiment time compared to the sampling and computational time is high, the complexity of the approach for functional estimation is less crucial since the total time will be dependent on the experiments. On the other hand, when the experiment time is reasonably lower, the computational time, including sampling time and fitting the GP model, will be significant, and thus faster approaches will be favorable. For example, in the second case study, experiments by a mobile robot are done instantly, and hence sampling and computational time are significant. Because of the time-sensitive nature of these applications, faster approaches will outperform. We will later show that the SIEVE approach will provide good results in a short time, which implies that this approach is appropriate for those applications.

The remainder of the paper is organized as follows. In Section 2, we describe the problem setup. In Section 3, we formally present our Sequential sampling algorithm, (SIEVE), and we describe the ideas behind it. In Section 4, we provide technical details of the algorithm and its theoretical properties. Numerical studies and simulations are offered in Section 5, and in Section 6, we describe two real-world applications of our method. Section 7 concludes the paper and discuss the future work.

## 2 | PROBLEM SETTING

Suppose we are interested in the estimation of a  $d$ -dimensional smooth, continuous function  $f : \mathcal{D} \subset \mathbb{R}^d \rightarrow \mathbb{R}$  (for a bounded, convex domain  $\mathcal{D}$ ) by sequentially exploring its domain  $\mathcal{D}$ . Our goal is to propose a procedure to sequentially identify the best candidate point that improves the estimation at each iteration. At the end of the sampling procedure, the collected data are fused to estimate the function  $f$ . Therefore, the proposed sequential sampling procedure is independent of the estimation method. In the simulations and the case studies, we use the Empirical Bayes paradigm, that is, a data-based approach in which the function is modeled as a GP<sup>43,44</sup> (see Appendix A for details).

We denote the sequence of the input coordinates of the measured/observed response up to time  $t \in \mathbb{N}$  as  $X_t = \{\mathbf{x}_1, \dots, \mathbf{x}_t\}$ , where  $\mathbf{x}_i \in \mathcal{D}$  is the coordinate vector that defines the position of the sample observed at iteration  $i = 1, \dots, t$ . We also denote the observed response corresponding to the position  $\mathbf{x}_i$ , by  $y_i$ , given by  $y_i = f(\mathbf{x}_i) + \epsilon_i$ , where  $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$  are independent, and identically distributed (i.i.d.) noises. Thus, the unknown function  $f$  is measured through the noisy observations  $y_1, \dots, y_t$ .

For notational convenience, we define observed values  $\mathbf{y}_t := (y_1, \dots, y_t)^\top$ ,  $f_t := f(\mathbf{x}_t)$ , and true function values as  $\mathbf{f}_t := (f_1, \dots, f_t)^\top$ . Additionally,  $\hat{f}(\mathbf{r})$  is used to indicate the estimated value of the function at each point  $\mathbf{r} \in \mathcal{D}$ .

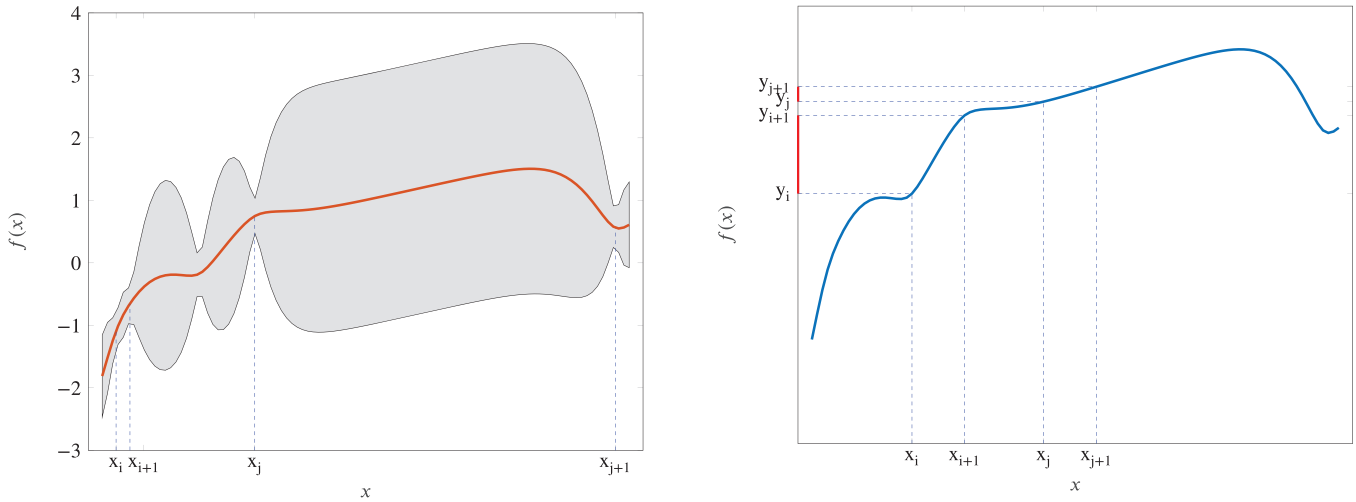
## 3 | PROPOSED SEQUENTIAL SAMPLING: SIEVE

As discussed earlier, existing sequential sampling methods require the estimation of the function at each iteration, which reduces their computational efficiency. In contrast, SIEVE exploits the available observations to obtain information about the unknown function behavior, without having to estimate an intermediate surrogate function.

The intuition behind our approach comes from two observations: First, as can be seen in Figure 1—left panel, the uncertainty of the surrogate model is likely small in the interval of two close observations in the domain, and it increases as observations get farther apart.<sup>43</sup> Second, a large gap between the responses of two neighbor observations, that is,  $|y_i - y_j|$ , may indicate a higher local variation, and hence, more observations are needed in that interval (see Figure 1—right panel).

We define the SIEVE sampling criterion by combining these two observations. Specifically, at each sampling iteration  $t$ , we search for an interval that maximizes the uncertainty proxy function given by

$$(i^*, j^*) = \operatorname{argmax}_{(i,j) \in \mathcal{P}_t} S_{i,j}^{(t)} := \operatorname{argmax}_{(i,j) \in \mathcal{P}_t} \{\|\mathbf{x}_i - \mathbf{x}_j\| + \lambda|y_i - y_j|\}, \quad (1)$$



**FIGURE 1** Visualization of the intuition. The left panel shows the estimation uncertainty in terms of variance. The right panel shows the amplitude of the local variation of the function (in red) being equal to the distance between the corresponding observed points.

where  $i, j \in \{1, \dots, t\}$ , and  $\mathcal{P}_t = \{(i, j) \mid \exists \text{ no observation between } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ up to } t \text{ observations}\}$  and  $(i^*, j^*)$  represent the end-points of the optimal interval in  $\mathcal{P}_t$ , and  $\lambda \geq 0$  is a parameter to scale the term  $|y_i - y_j|$  such that the two terms  $\|\mathbf{x}_i - \mathbf{x}_j\|$  and  $|y_i - y_j|$  will have the same magnitude for the optimization in Equation (1). Thus for the sake of optimization, we assume that the two terms have the same importance. As a guideline, we can use the following optimization to efficiently approximate a proper value for  $\lambda$ ,

$$\lambda = \max_{(i, j) \in \mathcal{P}_t} \frac{|y_i - y_j|}{\|\mathbf{x}_i - \mathbf{x}_j\|}. \quad (2)$$

To find the optimal solution for the foregoing problem, one crude way is to check all possible triples of points corresponding to iterations  $(i, j, k)$  to see if  $\mathbf{x}_k$  is co-linear with  $\mathbf{x}_i$  and  $\mathbf{x}_j$ . However, this approach is not scalable, especially for higher dimensions. For simplicity, we first describe the proposed algorithm for the one-dimensional case, and then we extend it to higher dimensions.

Suppose the location of observed points  $\mathbf{x}_1, \dots, \mathbf{x}_t \in \mathbb{R}$  are arranged in increasing order such that  $\mathbf{x}_{(i)} \leq \mathbf{x}_{(i+1)}$  for all  $i \in \{1, \dots, t-1\}$ , we only need to compute the scalars  $S_{(i), (i+1)}^{(t)}$ , (for  $i \in \{1, \dots, t-1\}$  in Equation 1), for neighboring intervals determined by  $\mathbf{x}_{(i)} \leq \mathbf{x}_{(i+1)}$ .  $S_{(i), (i+1)}$  denotes the uncertainty proxy function for ordered observations  $\mathbf{x}_{(i)}$  and  $\mathbf{x}_{(i+1)}$ . After finding the optimal interval  $(i^*, j^*)$ , the next sampled point is selected in the midpoint of this interval, that is,

$$\mathbf{x}_{t+1} = \frac{\mathbf{x}_{i^*} + \mathbf{x}_{j^*}}{2}. \quad (3)$$

Then, a realization of the function in  $\mathbf{x}_{t+1}$  is observed and recorded as  $y_{t+1}$ . In the  $d$ -dimensional ( $d \geq 2$ ) case, the *neighbor* points are determined by their *closeness* to one another, in the domain. The definition of closeness in two or more dimensions is given by leveraging on the Delaunay triangulation. The Delaunay triangulation or tessellation, commonly used in computational geometry,<sup>45</sup> is a partition of the domain into  $d$ -simplices whose vertices are the observed points. The circum-hypersphere of any resulting  $d$ -simplex contains no observed point. Such a partition can be made iteratively, that is, by adding one point at time resembling the iterative nature of SIEVE. It is known that in a Delaunay triangulation, the nearest neighbor graph is a subgraph of the Delaunay triangulation.<sup>46</sup> This implies that the closest neighbor to any observed point is on an edge in the Delaunay triangulation, which can help speed up the SIEVE. In fact, having pair of points whose separation distance is minimum allows to balance the contribution of  $\|\mathbf{x}_i - \mathbf{x}_j\|$  and  $|y_i - y_j|$  in Equation (1), that otherwise will shift in favor of  $\|\mathbf{x}_i - \mathbf{x}_j\|$  if it is too large. Figure 2 shows the difference between two possible triangulation of 17 points. The right one is the Delaunay triangulation. Figure 3 shows an example of the Delaunay triangulation of 10 points in a three-dimensional domain.

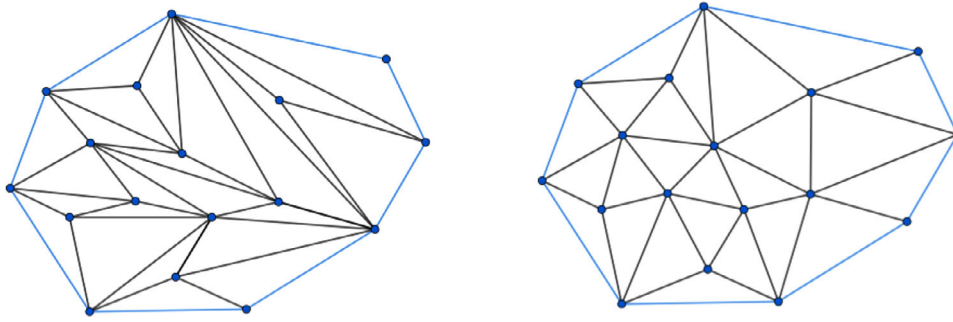


FIGURE 2 Arbitrary triangulation of 17 points (left) versus Delaunay triangulation (right).

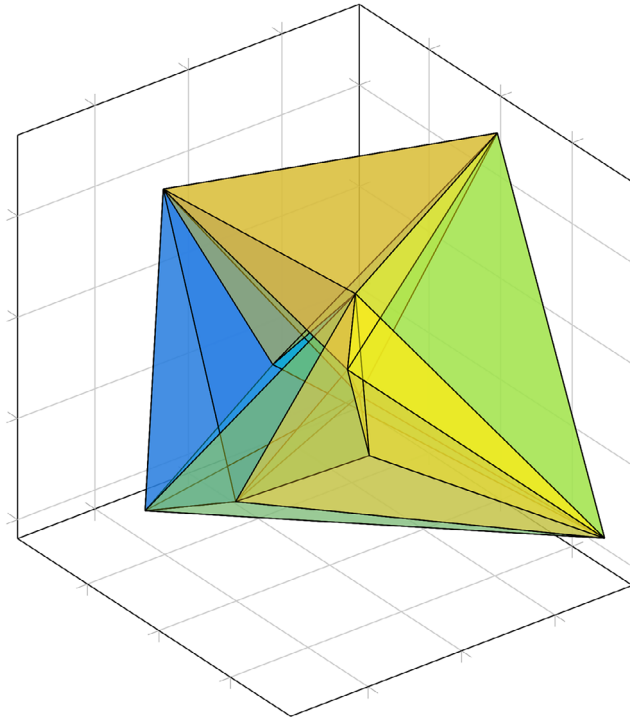


FIGURE 3 Triangulation of 10 points in a three-dimensional domain.

**ALGORITHM 1** Delaunay tessellation.

**Initialization:** Let  $\{p_1, \dots, p_k\}$  be a set of points forming a nondegenerate  $d$ -simplex.

**Addition of a new point:** Given a list of  $k + 1$  circumspheres and circumcenters and a new point  $p$  to be added:

- Flag the circumspheres that includes  $p$ ;
- remove the edges that belong to two or more flagged circumspheres;
- connect  $p$  with the points lying on the flagged circumspheres;
- update the list of circumspheres and circumcenters.

A pseudo-code of the Delaunay triangulation algorithm is given in Algorithm 1. To illustrate the algorithm, a two-dimensional step-by-step example is given in Figure 4. More details on the triangulation are given in Appendix B.

At each sampling iteration of the SIEVE, once the triangulation is completed, the quantity  $S_{i,j}$  is computed if the points  $\mathbf{x}_i$  and  $\mathbf{x}_j$  share a side of a  $d$ -simplex. By construction, if the middle point of  $(\mathbf{x}_i, \mathbf{x}_j)$ , that is,  $(\mathbf{x}_i + \mathbf{x}_j)/2$ , is in the set of the observed points, then  $\mathbf{x}_j$  and  $\mathbf{x}_i$  do not share any side of the simplices, and thus, they are not neighbors. This fact significantly reduces the number of pairs that should be checked to find the optimal interval. After the optimal interval in

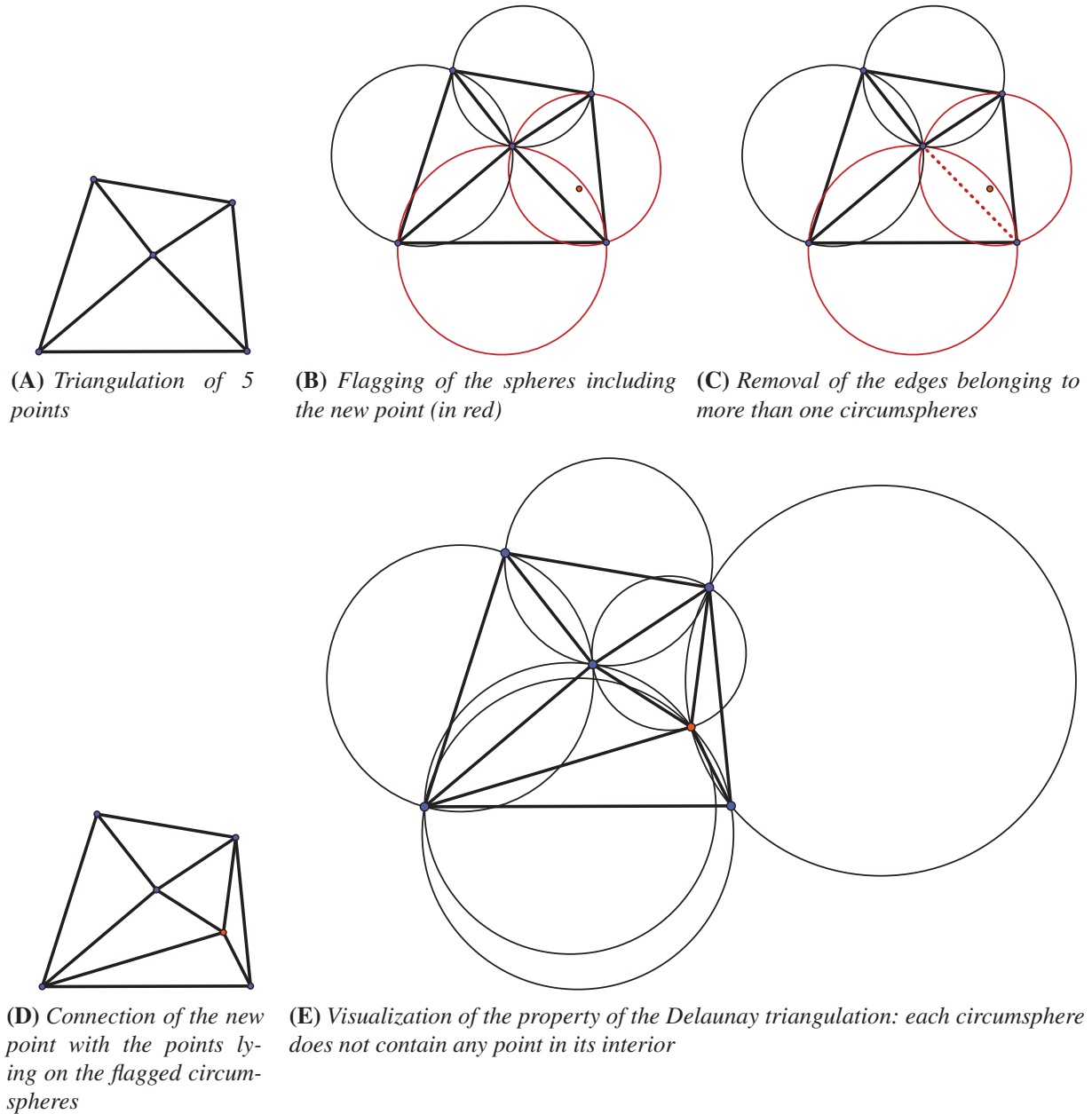


FIGURE 4 Example of addition of a new point in the triangulation process.

Equation (1) is found, the next point to be sampled is given by Equation (3) corresponding to the midpoint of the interval  $(i^*, j^*)$ .

*Remark 1.* The proposed sampling procedure, SIEVE, selects a new point inside the intervals bounded by the available observations. Therefore, to analyze the whole domain, initial points need to be placed on the boundary points of the domain.

This implies that the number of initial points, denoted by  $v$  should be at least  $d + 1$ .

The summary of all steps of the SIEVE procedure is formalized in Algorithm 2. The algorithm is run for  $N$  iterations, determined based on the available budget. Note that the function is only estimated at the end of the procedure with all the data at hand. The details describing the estimation of a function using GP are given in Appendix A.

**ALGORITHM 2** Sequential Sampling with Interval-based Exploration and Value Estimation (SIEVE).

**Initialization:** Let  $N$  be number of data points to be sampled. Select  $\mathbf{x}_1, \dots, \mathbf{x}_v$  in the vertices of the domain and acquire  $y_1, \dots, y_v$ ; run Algorithm 1 to triangulate the domain by using  $\mathbf{x}_1, \dots, \mathbf{x}_v$

**for**  $t=v+1, \dots, N$ : Given  $\mathbf{x}_1, \dots, \mathbf{x}_t$  and  $y_1, \dots, y_t$  and properly chosen value for  $\lambda$ :

**Sampling:** For each pair of indices  $(\mathbf{i}, \mathbf{j})$  corresponding to two neighbors  $\mathbf{x}_i$ , and  $\mathbf{x}_j$ , compute

$$(\mathbf{i}^*, \mathbf{j}^*) = \operatorname{argmax}_{\mathbf{i}, \mathbf{j} \in \mathcal{P}_t} \|\mathbf{x}_i - \mathbf{x}_j\| + \lambda |y_i - y_j|$$

and select

$$\mathbf{x}_{t+1} = \frac{\mathbf{x}_{i^*} + \mathbf{x}_{j^*}}{2}.$$

**Measurement:** observe the new data point

$$y_{t+1} = \mathbf{f}(\mathbf{x}_{t+1}) + \epsilon_{t+1}$$

**Triangulation:** Locally recompute the triangulation with  $y_{t+1}$  via Algorithm 1

**end**

**Estimation:** Compute an estimation  $\hat{\mathbf{f}}$  for the underlying function, using  $N$  observed data points.

## 4 | THEORETICAL AND COMPUTATIONAL ANALYSIS OF SIEVE

In this section, we provide some technical details related to SIEVE. Specifically, we analyze the computational complexity of the proposed procedure and study its convergence.

### 4.1 | Complexity analysis

Solving the optimization problem in Equation (1) is the main contributor to the complexity of the SIEVE, which requires the triangulation of a set of  $N$  points. The Delaunay triangulation is an iterative process whose total complexity is  $O(N^{(2d-1)/d})$ .<sup>47</sup> Computing the scalars  $S_{i,j}$  corresponding with the number of pairs of neighbor points, which is the same as the number of edges of the triangles, is linear in  $N$ .<sup>48</sup> Given the array of scalars  $S_{i,j}$ , finding the maximum value has linear complexity. Therefore, the overall complexity of the SIEVE algorithm is  $O(N^{(2d-1)/d})$ . Note that most of the existing sequential sampling methods require fitting a GP at each iteration, in addition to optimizing the sampling criterion. The complexity of only the GP part at iteration  $t$  is  $O(t^3)$ .<sup>43</sup> Hence, the overall GP complexity for  $N$  sequentially observed points is  $O(N^4)$ , which is significantly larger than the worst-case complexity (for the large values of  $d$ ) of the SIEVE,  $O(N^2)$ . Additionally, the complexity of optimizing the sampling criterion in most of the existing methods is worse than the linear time, the complexity of the optimization algorithm in SIEVE. As mentioned before, the experiment time plays a vital role in selecting a proper approach. When the experiment time is low, the complexity of sampling and fitting a GP will be dominant in the total time of the process. In this case, the SIEVE approach is advantageous since it requires only one function estimation, and thus, overall computation time will be low. The overall advantage of SIEVE in terms of time complexity is also demonstrated in the simulations in Section 5. For completeness, we note that the Delaunay triangulation for a two-dimensional domain can be computed much faster by using Ref. 49 Theorem 9.12 in Ref. 49 indicates that the Delaunay triangulation of a set of  $N$  points on a plane can be computed in the logarithmic expected time, that is,  $O(N \log N)$ .

### 4.2 | Convergence results and consistency

In this section, we show that the estimation error of the function  $\hat{\mathbf{f}}$  obtained by the samples  $\hat{\mathbf{f}}$  collected according to the SIEVE converges to zero as sample size,  $N$  increases. In other words, loosely speaking,  $\hat{\mathbf{f}}$  is a consistent estimator of  $\mathbf{f}$ . Clearly, the consistency of  $\hat{\mathbf{f}}$  depends on the posterior variance of the GP estimator. The behavior of the posterior variance has been investigated when the data are sampled randomly in the domain.<sup>50,51</sup> However, this is not trivial for the case when the samples are not i.i.d. or uniformly sampled. The result presented in this section shows the consistency of the posterior variance when the points are sampled via SIEVE. The result is valid under the following assumption.

**Assumption 1.** The unknown function  $f$  is  $L$ -Lipschitz and continuous, that is,  $\forall \mathbf{x}_1, \mathbf{x}_2 \in D, \exists L > 0$  s.t.  $|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|$ .

It is worth mentioning that the Lipschitz continuity holds for many functions, including convex functions with closed domains and polyhedral functions. Moreover, when the domain is a compact set, every continuously differentiable function is also Lipschitz continuous.

Using the Lipschitz continuity, one can write  $|y_i - y_j| = |f(\mathbf{x}_i) + \epsilon_i - f(\mathbf{x}_j) - \epsilon_j| \leq |f(\mathbf{x}_i) - f(\mathbf{x}_j)| + |\epsilon_i - \epsilon_j| \leq L\|\mathbf{x}_i - \mathbf{x}_j\| + |\epsilon_i - \epsilon_j|, \forall i, j \in \{1, \dots, t\}$ .

By the definition of the noise and the linearity of expectation, it can be seen that  $\mu(\|\epsilon_i - \epsilon_j\|) = 0$  and the bound can be approximated by  $\mathbb{E}(|y_i - y_j|) \leq L\mathbb{E}(\|\mathbf{x}_i - \mathbf{x}_j\|)$ . Moreover, Lipschitz continuity can be used to bound  $S_{i,j}^{(t)}$  for all  $t$  as follows.

**Lemma 1.** The scalars  $S_{i,j}^{(t)}$  can be upper bounded as

$$S_{i,j}^{(t)} \leq (\lambda L + 1)r(D),$$

where  $r(D)$  is the diameter of the domain  $D$ .

*Proof.*

$$\begin{aligned} S_{i,j}^{(t)} &\leq \max_{(i,j) \in \mathcal{P}_t} \|\mathbf{x}_i - \mathbf{x}_j\| + \lambda \max_{(i,j) \in \mathcal{P}_t} |y_i - y_j| \\ &\leq \max_{(i,j) \in \mathcal{P}_t} \|\mathbf{x}_i - \mathbf{x}_j\| + \lambda L \max_{(i,j) \in \mathcal{P}_t} \|\mathbf{x}_i - \mathbf{x}_j\| \\ &= (\lambda L + 1) \max_{(i,j) \in \mathcal{P}_t} \|\mathbf{x}_i - \mathbf{x}_j\| \\ &\leq (\lambda L + 1)r(D). \end{aligned}$$

□

This result ensures that the maximum computed in Equation (1) not only exists but is limited by a constant. The result is also used to prove the main theorem of this section, which states that if the points of the domain are sampled via SIEVE and then estimated through a GP, the posterior mean of the estimator  $\hat{f}$  goes to 0, that is, it converges to the true density function  $f$ .

**Theorem 1.** Suppose Assumption 1 holds for the underlying true function  $f$ . Consider a Gaussian Process with isotropic, decreasing covariance kernel  $K(\cdot)$ , an input training data set  $\mathbb{D}_T^x = \{\mathbf{x}_i\}_{i=1}^T$  collected via SIEVE and observation noise with variance  $\sigma^2$ . Then, for the posterior mean of the estimator  $\hat{f}$ , we show that  $\lim_{t \rightarrow \infty} \sigma_{t,\hat{f}}^2(\mathbf{r}) = 0$ ; for all  $\mathbf{r} \in D$ .

We include the proof of this theorem in Appendix C.

## 5 | EVALUATION AND COMPARISON USING SIMULATIONS

In this section, we evaluate the performance of the proposed SIEVE method using several simulation studies and compare it with three existing sequential sampling methods, namely, the sequential Expected Improvement<sup>52</sup> (see also Ref. 53) designated by “EI,” the sequential Minimum Energy<sup>30</sup> designated by “ME” design criterion and the Integrated Mean-Squared (prediction) Error for Sequential Design method<sup>54,55</sup> designated by “IMSE.” The Expected Improvement method seeks for an input value  $x$  that maximizes the expected value of the *improvement function*, where *improvement function* is a random variable which for any input  $x$ , measures how much the updated function will do better than the best value observed so far. On the other hand, the Minimum Energy design approach considers design points as particles with different charges



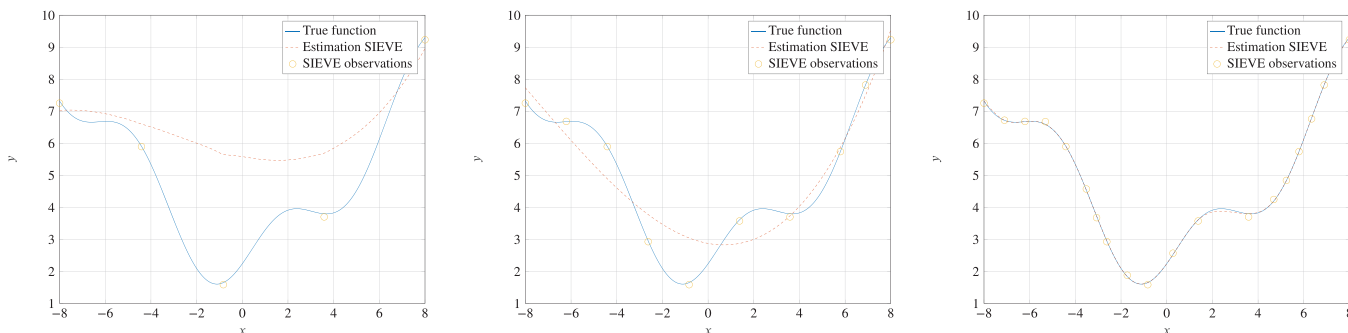


FIGURE 5 True function and estimated function after 5, 10, and 20 steps of SIEVE.

but all with the same sign. By choosing a proper charge function, this approach tends to find design points that minimize the total energy of the system of particles. The Integrated Mean-Squared Error for the Sequential Design method is based on the definition of the Integrated Mean-Squared Error, a metric used to assess the accuracy of a statistical prediction model. It might be minimized for choosing where to collect additional data, meaning to improve the accuracy of predictions across the entire input space. As in the other sequential method, the idea is to focus data collection in regions where the model is currently less accurate.

The accuracy of the estimation, measured by Relative Mean Square Error (RMSE), and the total elapsed time, are used as comparative measures. We perform the comparative study for functional estimation using sequential sampling. Additionally, to numerically study the scalability of the SIEVE to the function dimensions, we perform a sensitivity analysis for different dimensions and report the computation of time of the SIEVE for selecting the samples. For all of the following experiments, we tried possible values for  $\lambda$  and based on results, it turns out that  $\lambda = 1$  is the best value for all experiments. This means the parameter  $\lambda$  can be omitted from the formula (1). Moreover, we used *Matérn* covariance function with hyperparameters  $\log(1/4)$  and 0.

## 5.1 | Functional estimation

We begin with a simple one-dimensional example to illustrate how the proposed SIEVE works. Specifically, we consider  $f : [-8; 8] \subset \mathbb{R} \rightarrow \mathbb{R}$ ,  $f(x) = \sqrt{(x^2 + 5)} + \sin(x)$ .

We generate noisy observations from this function with the standard deviation of  $\sigma = 0.05$ . We consider three different sample budgets 5, 10, or 20 points collected by using the SIEVE, and use a GP model to estimate the function. To estimate the hyper-parameters of the GP model, we use a MATLAB package developed by Rasmussen and Williams.<sup>56</sup> The sampled points and the estimated function for each budget are shown in Figure 5. From this figure, we see the accuracy of the fitted function significantly increases as the budget increases. When  $N = 5$ , the locations of sampled points indicate that SIEVE first picks points whose corresponding responses have large differences while trying to cover the whole domain. A similar observations can be made for  $N = 10$  and  $N = 20$  cases. This is compatible with the intuition behind the sequential sampling optimization problem in Equation (1).

We also compare the SIEVE with EI and ME for  $N = 20$ . Figure 6 shows the estimation of the function along with the selected points for each method. As can be seen from the figure, the difference among the three methods is not visually identifiable. In order to quantify the accuracy of the methods, we compute the average RMSE at each iteration of sampling given by  $ARMSE(t) = \frac{1}{N} \sum_{i=1}^N \left( \sum_{i=1}^R (f_t(\mathbf{x}_i) - \hat{f}_t(\mathbf{x}_i))^2 / \sum_{i=1}^R f_t(\mathbf{x}_i)^2 \right)$ , where  $y_i$  is the true value of the function,  $\hat{f}_t(\mathbf{x}_i)$  is the predicted value of the function at point  $\mathbf{x}_i$  after  $t$  iterations, and  $R$  is the total number of points in a grid at which the function is evaluated. The ARSME values for different sampling budgets with  $R = 300$  are plotted in Figure 7. Figure 8 also shows the elapsed time for the selection of the points (in seconds) against the budget. As can be seen from the figures, although the performance of EI and SIEVE in term of accuracy is comparable, the SIEVE is about  $10^3$  faster than both the EI and ME for small sample sizes. This difference is more profound when the sample size increases. For example, for  $N = 100$ , SIEVE is  $10^5$  and  $10^{2.5}$  faster than EI and ME, respectively. This is because unlike the benchmarks, SIEVE's computational complexity for the one-dimensional case is linear in  $N$ , which is clear in Figure 8. Both the EI and SIEVE outperform the ME in terms of prediction accuracy.

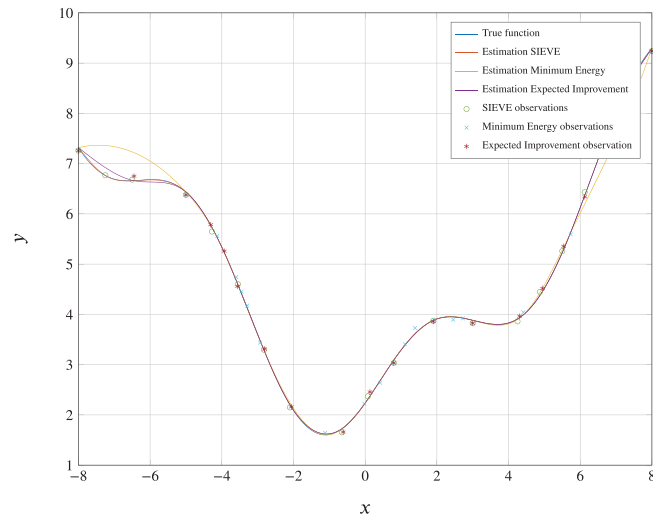


FIGURE 6 True versus estimated functions using 20 observations.

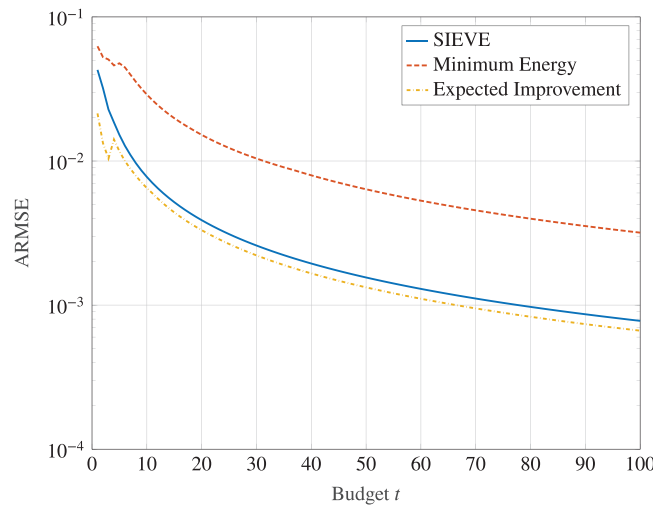


FIGURE 7 Performance comparison in terms of the ARMSE.

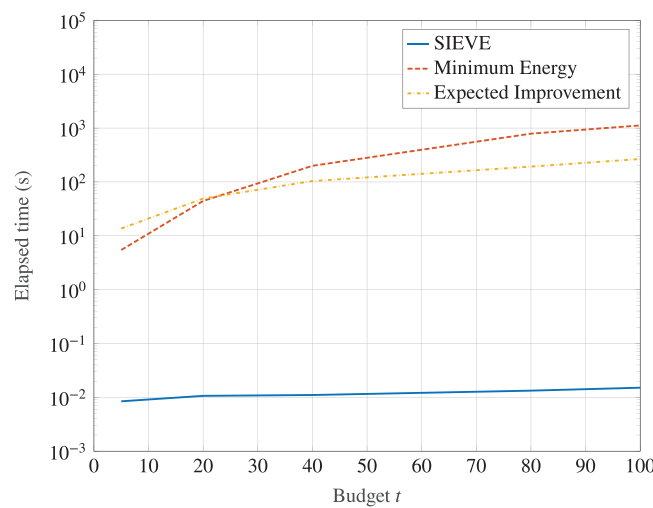
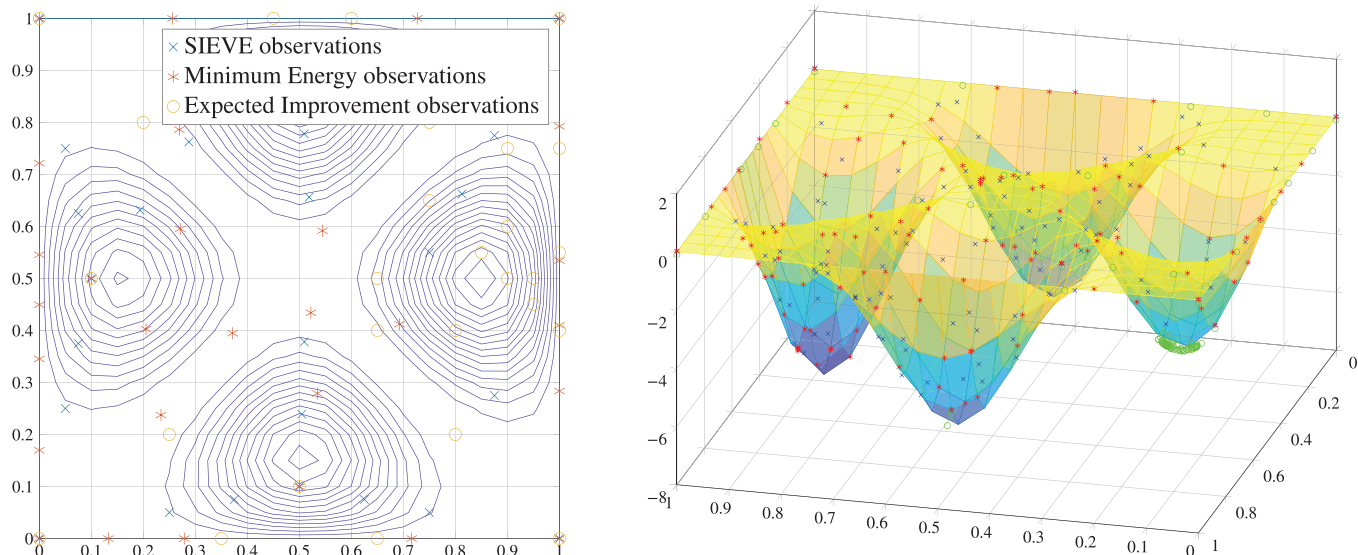
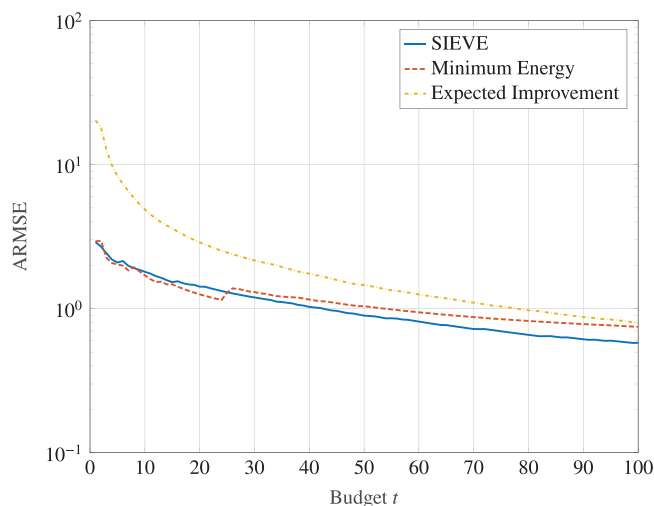


FIGURE 8 Comparison between the elapsed time of the three methods.



**FIGURE 9** Distribution of the observed points in the two-dimensional domain. Upper panel: 50 samples on contour lines. Lower panel: 100 samples the 3D plot.



**FIGURE 10** Performance comparison in terms of the ARMSE.

In the second example, we consider a two-dimensional function, which was also used in Ref. 57 to validate the Expected Improvement algorithm.  $f : [0;1]^2 \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ ,  $f(x_1, x_2) = -5(1 - (2x_1 - 1)^2)(1 - (2x_2 - 1)^2)(4 + 2x_1 - 1) \left( 0.05(2x_1 - 1)^2 - 0.05(2x_2 - 1)^2 \right)^2$ . Similar to the first example, we use the SIEVE as well as other benchmarks to sample and estimate the unknown function using different sample budgets ranging from 5 to 100. The observed responses are sampled according to Equation (3) where the standard deviation of the noise is  $\sigma^2 = 0.05$ . Figure 9 shows the distribution of points sampled by using the three methods for  $N = 100$ . As observed from the figure, the SIEVE has a smaller number of points collected on the flat regions of the function indicating that it uses the sample more effectively.

The ARMSE for  $R = 300$  and elapsed time curves are also shown in Figures 10 and 11, respectively. The SIEVE and ME show a similar accuracy, while they both outperform the EI. However, the SIEVE has substantially less computational time than the benchmarks for all sample budgets. Similar to the 1D example, as the sample size increases the elapse time gap between the SIEVE and the benchmarks increases.

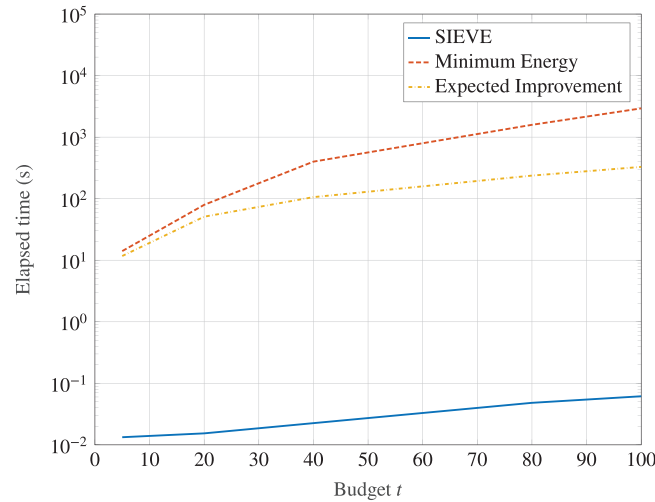


FIGURE 11 Comparison between the elapsed time of the three methods.

In the third example, we consider the Levy function that can be defined in  $d$  dimension as follows:

$$f(x) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1} (w_i - 1)^2 [1 + 10 \sin^2(\pi w_i + 1)] + (w_d - 1)^2 [1 + \sin^2(2\pi w_d)],$$

where  $w_i = 1 + \frac{x_i - 1}{4}$ ,  $\forall i \in \{1, \dots, d\}$ . For this example, we consider another benchmark, namely we consider the Integrated Mean-Squared (prediction) Error (IMSE) for Sequential Design method.<sup>54,55</sup> This approach seeks to find sample points to minimize the IMSE function defined as

$$IMSE = \int_D MSE(x) dx,$$

where  $MSE(x)$  represents the mean squared error function and  $D$  represents the domain. We compare the SIEVE as well as other benchmarks using different sample budgets ranging from 1 to 100. The observed responses are sampled according to Equation (3) where the standard deviation of the noise is  $\sigma^2 = 0.5$ . In order to compare the accuracy, we compute the root mean square error,  $RoMSE = \sqrt{(\sum_{i=1}^R [y_i - \hat{f}_R(\mathbf{x}_i)]^2)/R}$ . The RoMSE curves for  $R = 441$  are shown in Figure 12. The SIEVE, IMSE, and EI show a similar accuracy, while they all outperform the ME. However, the SIEVE has substantially less computational time than the benchmarks for all sample budgets. Similar to the 1D example, as the sample size increases the elapse time gap between the SIEVE and the benchmarks increases. The last example has been implemented in R by leveraging on the packages `deldir`, `DiceOptim`, `deepgp`, and `mined`.<sup>58–61</sup>

Moreover, to compare the SIEVE with a one-shot algorithm, we perform two additional simulation studies using the foregoing function and report their results in Appendix D. Specifically, we consider the one-shot sampling approaches including the Latin Hypercube, and a simplified version of SIEVE where we do not exploit the observed function values. As can be implied from the results, the SIEVE significantly outperforms both of these methods, which indicates the importance of exploiting the information of the observed samples at each step of sampling.

## 5.2 | Scalability analysis with respect to dimensions

One typical concern with the sequential sampling method is the scalability of the method as the problem dimensions increase. As we discussed in Section 4, the computational complexity of the SIEVE is  $O(N^{2-1/d})$ , which makes it quite scalable with respect to dimension. To verify this numerically, we run a sensitivity analysis where we compute the elapsed time for various dimensions using the SIEVE ranging from 1 to 45. We use a symmetric function on a square domain, given by,  $f : [0, 1]^d \rightarrow \mathbb{R}$ :  $f(\mathbf{x}) = 5 \cdot \exp \left\{ \sum_{i=1}^d (-x_i^2) \right\}$ . We used the vertices of the domain  $[0, 1]^d$  to initialize the algorithm,

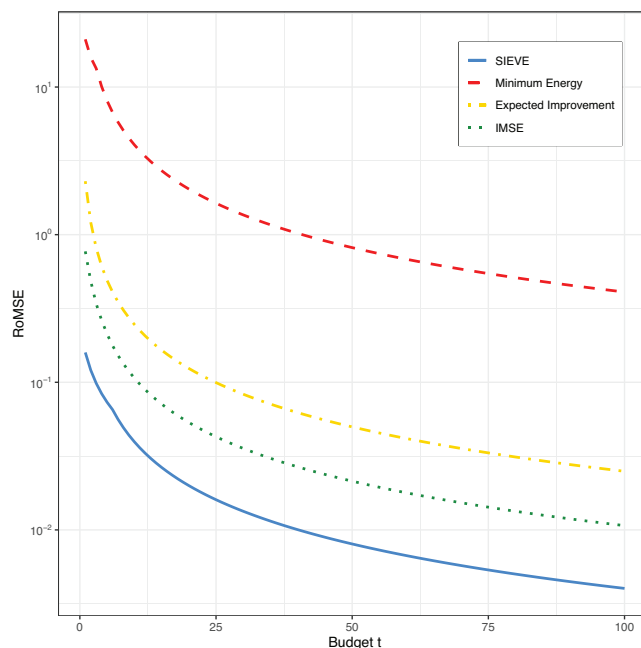


FIGURE 12 Performance comparison in terms of the RoMSE.

TABLE 1 Method performance for different domain sizes.

	$d = 3$	$d = 5$	$d = 7$
RoMSE after 150 iters	0.0049	0.0721	0.2779
Triangulation of 150 points	0.035065 s	0.140633 s	7.196230 s

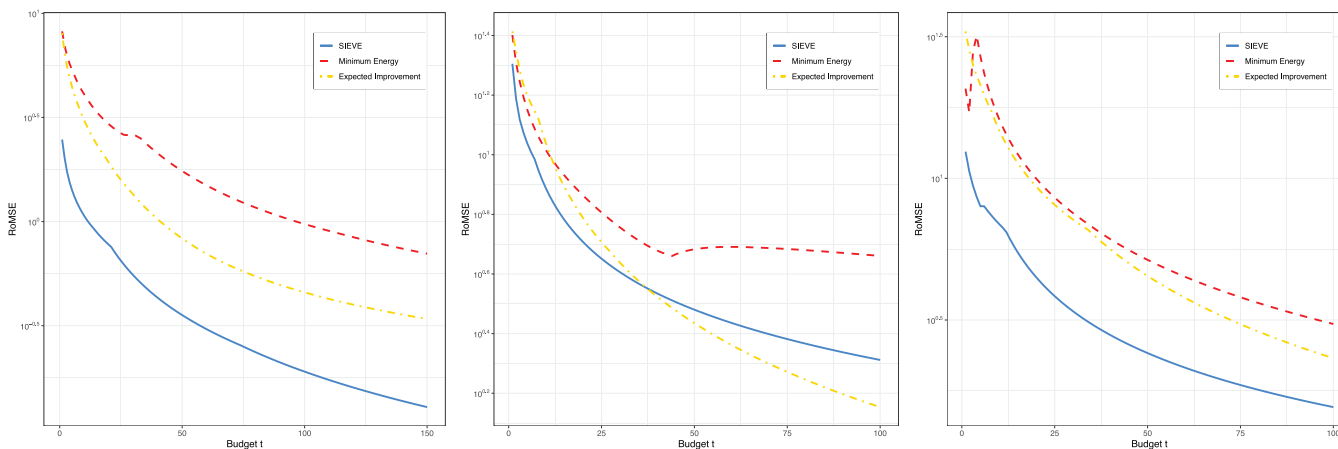


FIGURE 13 Comparison of methods for Levy function for  $d = 3$  (left figure),  $d = 5$  (middle figure), and  $d = 7$  (right figure) in terms of RoMSE.

and then we collected 150 points according to the SIEVE. In Table 1, we show the elapsed time as well as the RoMSE for  $d = 3, 5, 7$ . For example, in the seven-dimensional case, it takes the algorithm only about 7 s to select the location of all samples. As the results show, the proposed method is quite scalable to higher dimensions.

Lastly, we perform a comparison between SIEVE and the other benchmarks in three different dimensions: 3, 5, and 7 with application on the Levy function. The performances are shown in terms of RoMSE in Figure 13. It is not surprising that EI performs better under the same sample size. The real concern is that, while in greater dimensions the SIEVE requires little time to iteratively compute the new data, the other methods are extremely slower. In particular, SIEVE can compute the new points in order of seconds while the other two methods could require several minutes for each new point.

## 6 | CASE STUDY

In this section, we further validate our proposed method through two real case studies. The first case study focuses on an engineering application of SIEVE. Specifically, the engine calibration problem is discussed in which the proposed sequential sampling is used to find an optimal setting for diesel engines. In the second case study, we discuss how SIEVE can be used for exploring an unknown spatial domain using robots in hazardous environments.

### 6.1 | Engine calibration case study

In the last decade, the increase of legal requirements for exhaust emissions<sup>62,63</sup> has led to a continuous increase of complexity in internal combustion engine calibration, which is the process of experimentally finding the optimal set of control parameters in the engine. For optimal engine operations, the Engine Control Unit (ECU), an integrated computing device used on engines to provide control signals to the actuators, contains sets of look-up tables that store the control parameters for each actuator over the operating range of the engine. These look-up tables are shown by maps, each representing a two-dimensional function of the engine speed and torque (the *engine operating point*). By analyzing the signals acquired from the sensors, engineers can calibrate the actuator parameters stored in the ECU to optimize the engine performance (e.g., minimizing fuel consumption, emissions, vibrations, and maximizing hardware protection). These tasks are performed manually on the engine test bench. However, due to several sensors and actuators and hundreds or even thousands of operating points, a large number of resources (labor, fuel, and time) are required before an optimal parameter setting is determined. Therefore, a sequential sampling approach that reduces the number of tests is vital for engine calibration.<sup>9,10,64,65</sup>

We utilize the proposed SIEVE algorithm to calibrate a Diesel engine with a displacement of 2000cc and four cylinders. In particular, the calibration of two actuator maps is considered: (i) the air mass of the combustion process and (ii) the boost pressure. For each engine operating point, the calibration objective consists of obtaining the optimal air mass and boost pressure that results in the best  $\gamma$  signal (the air/fuel ratio). Thus, it regulates the lambda value to its stoichiometric value ( $\gamma = 1$ ) to achieve a maximum conversion efficiency of the catalytic converter, as required by legal requirements for exhaust emissions.

For both actuator signals, actual data were obtained from a test plan covering the whole engine operating range over a set of 256 engine operating points. The points have been obtained by combining values of engine speed and torque ranging in [1000, 4000] rpm and [10, 50] Nm, respectively (16 values of engine speed to 16 values of torque). For each operating point, the optimal air mass and boost pressure are recorded. Using these optimal values, we can create an emulator of each calibration map by fitting a GP with the 256 available data. We use the response surface generated by these emulators as the true underlying function. However, in reality, performing calibration over such a large number of engine operating conditions is not a cost-effective procedure. Furthermore, using such a test plan to cover the whole engine operating range may lead to unnecessary or redundant information for optimization purposes. Therefore, we use SIEVE to sequentially sample only 50 points. Figures 14 and 15 show the map emulator of air mass and boost pressure, respectively. In both figures, examples of the sampled points for iterations 10, 20, and 50 iterations are also illustrated.

We also compare the accuracy of the surrogate models estimated based on the points sampled by using the SIEVE, EI and by ME by repeating the experiment 10 times and computing the root mean square error,  $\text{RoMSE} =$

$\sqrt{(\sum_{i=1}^R [y_i - \hat{f}_{100}(\mathbf{x}_i)]^2)/R}$ . For this purpose, we compute the error over an equidistant grid of size  $57 \times 36$  operating points in the first experiment and  $30 \times 42$  operating points in the second experiment. The RoMSE boxplots of the air mass and boost pressure for all three methods are given in Figures 16 and 17, respectively. The RoMSE boxplots of both the air mass and boost pressure indicate both the SIEVE and EI with a similar accuracy outperforms the ME. For the boost pressure, however, the interquartile range of SIEVE is larger than that of the benchmarks.

As previously mentioned, the main advantage of the SIEVE is its computational efficiency. Tables 2 and 3 show the computational time of each method for both cases. As can be seen from the tables, SIEVE can complete the sequential sampling procedure in milliseconds, while it takes more than 500 s for EI and 4000 s for ME to determine the points' locations. To implement a point-by-point calibration strategy, it would be necessary to adopt a fast sequential sampling approach, minimizing the engine idle time when running on the test bench. From our results, we conclude that SIEVE can effectively be implemented as an algorithm to determine the ECU parameters with low test cost, less human effort, and via a point-by-point engine calibration strategy.

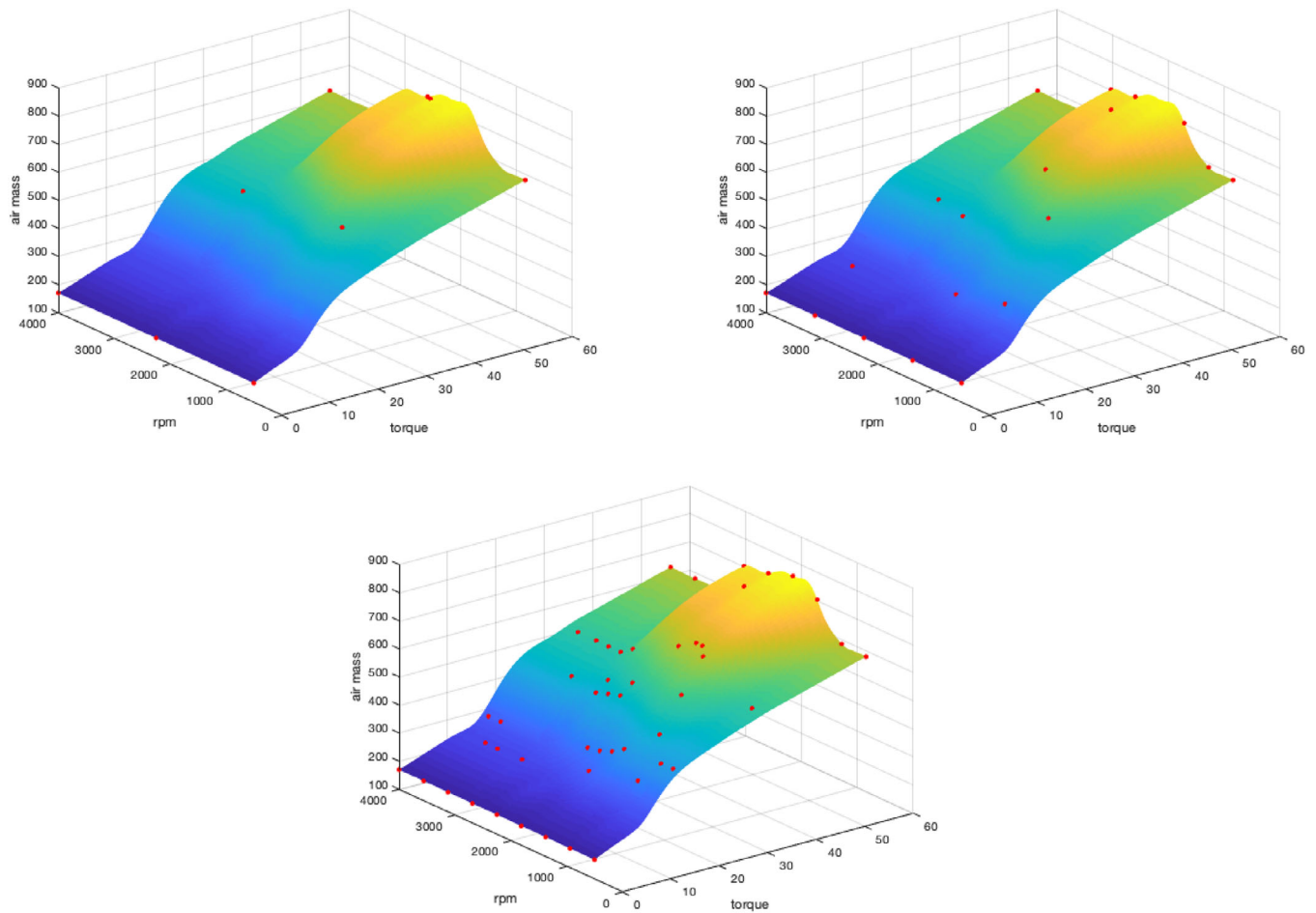


FIGURE 14 Air mass map: iterative selection of 10, 20, and 50 sampled points.

## 6.2 | Optimal exploration of an unknown environment

The problem of exploration and coverage of an unknown spatial field by a mobile robot has extensively been studied in many applications including autonomous environmental monitoring<sup>37,66</sup> and autonomous search-and-rescue missions.<sup>39,67</sup> In this context, the optimal coverage is referred to as placing/guiding a robot to a point in the domain  $D$  where it can best model an event whose importance is described by the density function of the field  $f$ .<sup>68,69</sup> For example, a robot may be asked to reconstruct the air quality map by detecting the pollution level in a certain area and selecting the best monitoring positions in this area.<sup>13</sup> In another scenario, a moving sensor can be used to reconstruct and also predict a spatio-temporal signal, finding application in climate change detection, weather forecasting, and water pollution mapping.<sup>70</sup> Typically, the solution of the problem is computed under the assumption that the density function is known.<sup>71,72</sup> Recently, however, Refs. 3, 73 studied the scenario where the density function is unknown and is estimated using a sequential sampling approach. The coverage problem can be formalized as an optimization problem where the objective function, the so-called locational cost, is given by<sup>68</sup>:

$$\ell(f, \mathbf{x}) = \int_{q \in D} \|q - \mathbf{x}\|^2 f(q) dq, \quad (4)$$

where  $\mathbf{x}$  represents the location of the robot. When the function  $f$  is known a local solution of the locational problem, denoted by  $c_m(f)$ , is the weighted center of mass of the domain, that is,

$$c_m(f) = \frac{1}{\int_D f(q) dq} \int_D q f(q) dq.$$

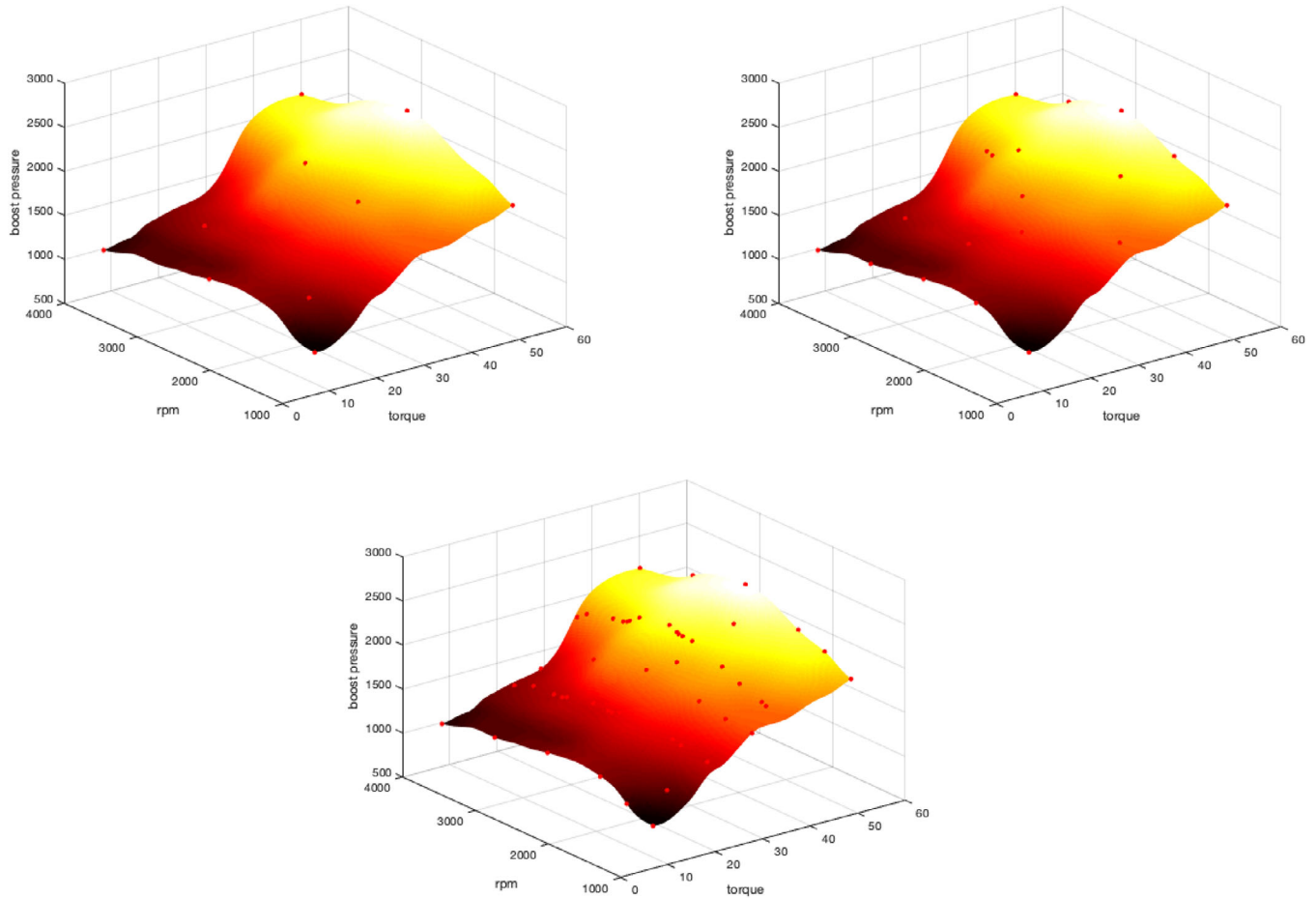


FIGURE 15 Boost pressure map: iterative selection of 10, 20, and 50 sampled points.

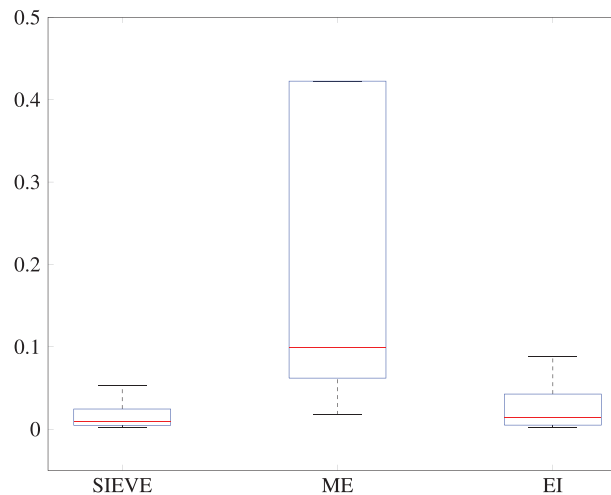


FIGURE 16 Box-plots of the RoMSE measures in the air mass case.

As in the search-and-rescue missions, time is of the essence, an agile sequential sampling procedure like the SIEVE would be vital in the success of the mission. Additionally, the robot has limited battery and computing capabilities. Therefore, reducing the time needed to compute the next position would be critical. In this case study, we use the SIEVE to guide a mobile robot to explore a domain and collect data for estimating the density function  $f$  and leading the robot to the optimal solution of the estimated coverage problem, that is,  $c_m(f)$ . To this end, we utilize the Robotarium



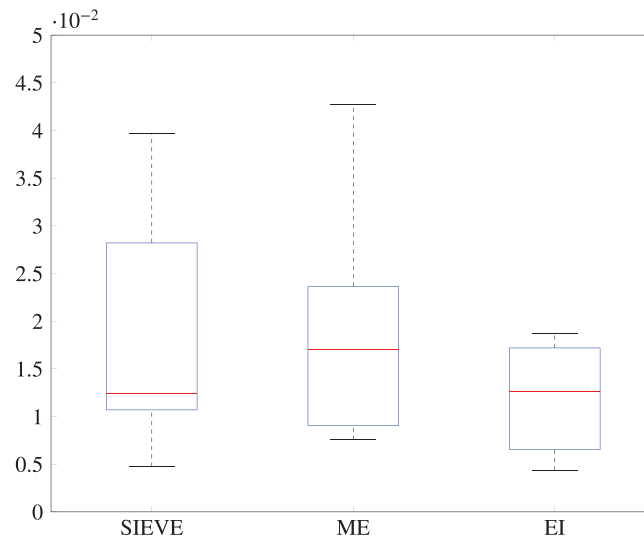


FIGURE 17 Box-plots of the RoMSE measures in the boost pressure case.

TABLE 2 Method performance when applied to the air mass map.

	Expected improvement	Minimum energy design	SIEVE
Mean of the elapsed time (s)	539.4888	4125.6	<b>0.1355</b>
Mean of the RMSE	0.0258	0.7131	<b>0.0168</b>
Median of the RMSE	0.0137	0.0992	<b>0.0095</b>
Bias of the RMSE	0.0377	0.3605	<b>0.0198</b>

The bold values represent the minimum of each row and highlight which method performed better in the simulations.

TABLE 3 Method performance when applied to the boost pressure map.

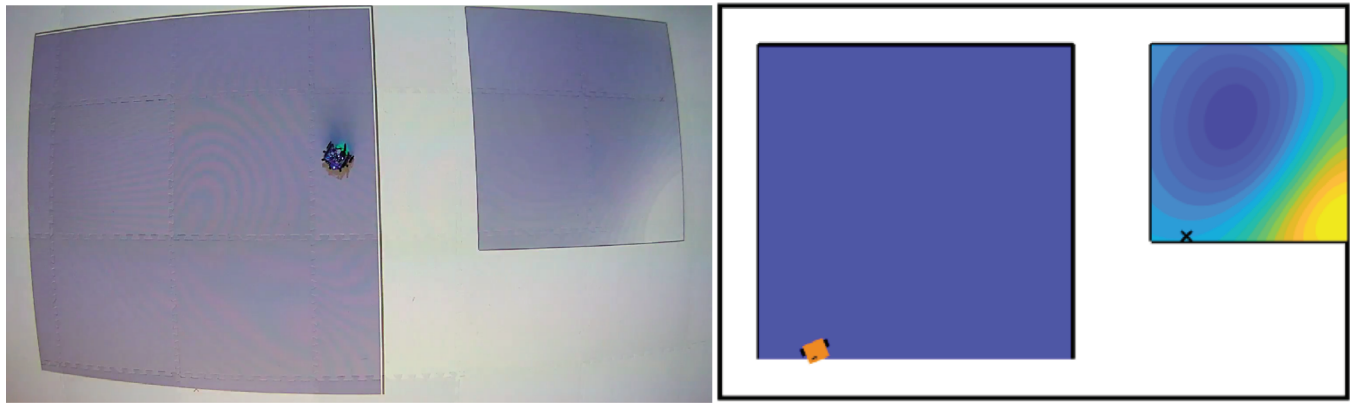
	Expected improvement	Minimum energy design	SIEVE
Mean of the elapsed time (s)	562.0488	59089	<b>0.1880</b>
Mean of the RMSE	<b>0.0119</b>	0.0190	0.0301
Median of the RMSE	0.0127	0.0170	<b>0.0124</b>
Bias of the RMSE	<b>0.0098</b>	0.0179	0.0897

The bold values represent the minimum of each row and highlight which method performed better in the simulations.

(<https://www.robotarium.gatech.edu>)<sup>74</sup> platform of Georgia Tech's Institute for Robotics and Intelligent Machines. We perform an experiment with a robot able to move according to a script created by using the SIEVE and previously uploaded on the platform. The density function used in the experiment is  $f : [-1, 1]^2 \rightarrow \mathbb{R}$ :

$$f(x) = 3(1 - x_1)^2 e^{-x_1^2 - (x_2 + 1)^2} - \frac{1}{3} e^{-(x_1 + 1)^2 - x_2^2} + \frac{7}{2} - 10 \left( \frac{x_1}{5} - x_1^3 - x_2^5 \right) e^{-(x_1^2 - x_2^2)}.$$

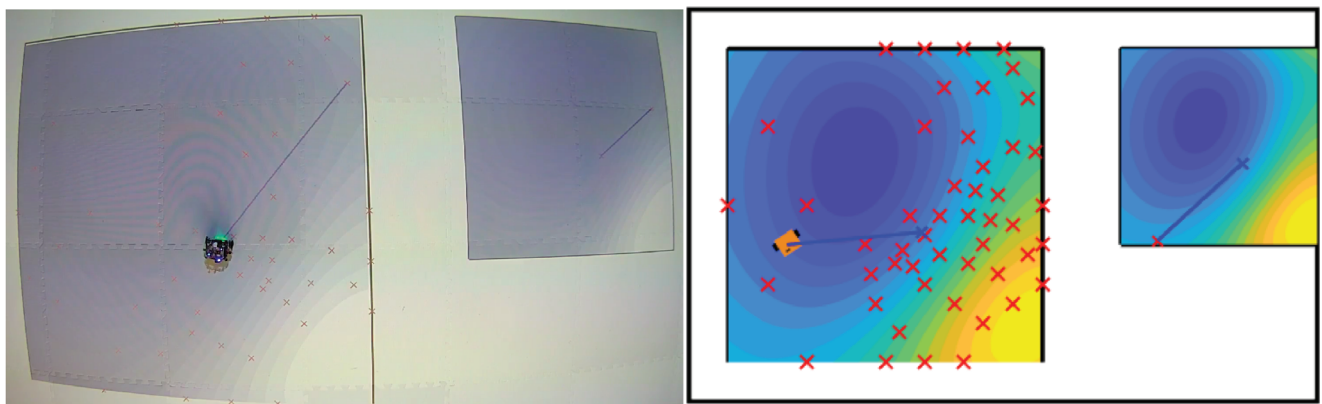
The robot starts from a set of initial positions on the vertices of a squared domain. Then, the robot is guided by the SIEVE to a new position and it senses a new value of the density function in the new position. When the number of sampled points reaches the prespecified budget, the density function is estimated and the robot moves to the weighted center of mass  $c_m(\hat{f})$  that minimizes the locational cost. A video shows that the complete experiment is available in the online appendix (<https://robotzoo-video.ecs.gatech.edu/owncloud/index.php/s/Oox8PQCe0QdHaUL/download>). For examples, three snapshots of the process at the initial stage, after collecting 10 points, and at the final stage, are shown in Figure 18. The left panels are the snapshots of the experiment performed in the Robotarium platform. The right panels represent the same experiment in simulations along with the contour lines of the estimated function  $\hat{f}$  (large square), and actual function  $f$  (small square). Notice that the final solution is almost the same as the solution of the optimization problem (4).



(A) Initial configuration: at the beginning, the robot is placed in an initial point  $x$ . The top-right squares show the contour lines of the true density function  $f$



(B) The robot explores the domain using the SIEVE. The positions explored by the robot are depicted in red. After exploring 10 points, an estimation of the density function is computed. The contour line of the estimated function is shown in the left square of the image



(C) After collecting 50 points, the density function is estimated, and the optimal coverage w.r.t. the estimated function is computed.

**FIGURE 18** The solution of the optimal coverage problem with initially unknown density function by a moving robot in the Robotarium (left) and simulation (right).

Finally, the distance between the optimal solution of Equation (4) and its estimate  $\|c_m(f) - c_m(\hat{f})\|$  is computed. This quantity as well as the RSME of the estimation of the function  $f$  after 50 iterations is also computed for the SIEVE as well as other benchmarks. To have a fair comparison, all methods start from the same set of initial points. The results are reported in Table 4. From the table, it is clear that the SIEVE has the best results both in terms of the accuracy of the estimated function and the accuracy of the optimal solution. The Expected Improvement has comparable results, while the Minimum Energy Design is the least accurate. The results show a similar behavior as seen in the previous simulations in terms of the time. The SIEVE is by far the fastest method with 0.91 s. Notice that the time reported in the table is the

TABLE 4 Optimal coverage.

	Expected improvement	Minimum energy design	SIEVE
$\ c_m(f) - c_m(\hat{f})\ $	$1.2516 \cdot 10^{-5}$	0.0031	<b><math>4.6403 \cdot 10^{-6}</math></b>
RSME	<b>0.0088</b>	0.0098	<b>0.0088</b>
Elapsed time	214 s	1981 s	<b>0.91 s</b>

The bold values represent the minimum of each row and highlight which method performed better in the simulations.

elapsed time in simulation. In the real experiment, one should add the time used by the robot to move from a point to the other, thus the overall time depends on the robot's speed.

## 7 | CONCLUSIONS

In many applications, estimating a function by exploring its domain should also consider the exploration cost, acquisition of new data, and the required computational time. In this paper, we proposed a new SIEVE, capable of efficiently exploring space and accurately estimating an unknown function. In contrast with the existing sequential sampling methods, the SIEVE is simple and fast as it does not require the estimation of the function after sampling a new point. To corroborate the theoretical results, we applied the proposed method to several simulated examples. Furthermore, we showed the effectiveness of the SIEVE in two case studies of the engine calibration scenario and optimal field coverage using mobile robots. Both the simulations and case studies showed that the accuracy of the estimated function by using the SIEVE is comparable with that of other benchmarks, while the SIEVE is considerably faster. In this paper, we focused on the single-agent sequential sampling problem. Extension of this approach to a multi-agent setting would be a potential direction for future research. Furthermore, within our approach, we opt for the simplicity of selecting new points at the midpoint of each edge. However, as part of our future research objectives, we aim to investigate the possibility of choosing sample points from the internal regions of the simplices.

## ACKNOWLEDGMENTS

The work of A.B. and M.P. was supported by the Ministry of University and Research of Italy's "Innovative Research Fellowships with Industrial Characterization" under the National Operational Program for Research and Innovation. S.G. would like to acknowledge support from NSF CAREER Award 2239824.

## DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

## ORCID

Alessia Benevento  <https://orcid.org/0009-0003-7346-4922>

Pouya Ahadi  <https://orcid.org/0000-0002-6055-9688>

Kamran Paynabar  <https://orcid.org/0000-0002-6906-3611>

## REFERENCES

- Hernández JD, Moll M, Vidal E, Carreras M, Kavraki LE. Planning feasible and safe paths online for autonomous underwater vehicles in unknown environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*; 2016:1313-1320.
- Kemna S, Rogers JG, Nieto-Granda C, Young S, Sukhatme GS. Multi-robot coordination through dynamic Voronoi partitioning for informative adaptive sampling in communication-constrained environments. *IEEE International Conference on Robotics and Automation (ICRA)*; 2017:2124-2130.
- Benevento A, Santos M, Notarstefano G, Paynabar K, Bloch M, Egerstedt M. Multi-robot coordination for estimation and coverage of unknown spatial fields. *IEEE*; 2020:7740-7746.
- Liu M, Cheung C, Cheng CH, Lee W. A Gaussian process data modelling and maximum likelihood data fusion method for multi-sensor CMM measurement of freeform surfaces. *Appl Sci*. 2016;6(12):409.
- Colosimo BM, Pacella M, Senin N. Multisensor data fusion via Gaussian process models for dimensional and geometric verification. *Precis Eng*. 2015;40:199-213.
- Sadaoui SE, Mehdi-Souzani C, Lartigue C. Multisensor data processing in dimensional metrology for collaborative measurement of a laser plane sensor combined to a touch probe. *Measurement*. 2022;188:110395.

7. He G, Sang Y, Pang K, Sun G. An improved adaptive sampling strategy for freeform surface inspection on CMM. *Int J Adv Manuf Technol*. 2018;96:1-15.
8. Ren J, Jian Z, Wang X, Mingjun R, Zhu L, Jiang X. Complex surface reconstruction based on fusion of surface normals and sparse depth measurement. *IEEE Trans Instrum Meas*. 2021;70:1-13.
9. Martinez-Morales JD, Palacios-Hernandez ER, Velázquez-Carrillo GA. Modeling and multi-objective optimization of a gasoline engine using neural networks and evolutionary algorithms. *J Zhejiang Univ Sci A*. 2013;14(9):657-670.
10. Wong PK, Wong KI, Vong CM, Cheung CS. Modeling and optimization of biodiesel engine performance using kernel-based extreme learning machine and cuckoo search. *Renew Energy*. 2015;74:640-647.
11. Aliramezani M, Koch CR, Shahbakhti M. Modeling, diagnostics, optimization, and control of internal combustion engines via modern machine learning techniques: a review and future directions. *Prog Energy Combust Sci*. 2022;88:100967.
12. Sun Y, Lv L, Lee P, Cai Y. Prediction of in-cylinder pressure of diesel engine based on extreme gradient boosting and sparrow search algorithm. *Appl Sci*. 2022;12(3):1756.
13. Zeng Y, Xiang K. Adaptive sampling for urban air quality through participatory sensing. *Sensors*. 2017;17(11):2531.
14. Ezzat AA, Pourhabib A, Ding Y. Sequential design for functional calibration of computer models. *Technometrics*. 2018;60(3):286-296.
15. Diao H, Wang Y, Wang D. A D-optimal sequential calibration design for computer models. *Mathematics*. 2022;10(9):1375.
16. Yondo R, Andres E, Valero E. A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses. *Prog Aerosp Sci*. 2018;96:23-61.
17. Zhang Y, Peng P, Liu C, Xu Y, Zhang H. A sequential resampling approach for imbalanced batch process fault detection in semiconductor manufacturing. *J Intell Manuf*. 2022;33:1-16.
18. Wong PK, Gao XH, Wong KI, Vong CM. Efficient point-by-point engine calibration using machine learning and sequential design of experiment strategies. *J Frankl Inst*. 2018;355(4):1517-1538.
19. Lu L, Anderson-Cook CM, Martin M, Ahmed T. Practical choices for space-filling designs. *Qual Reliab Eng Int*. 2022;38(3):1165-1188.
20. Johnson ME, Moore LM, Ylvisaker D. Minimax and maximin distance designs. *J Stat Plan Inference*. 1990;26(2):131-148.
21. Santner TJ, Williams BJ, Notz WI. Space-filling designs for computer experiments. *The Design and Analysis of Computer Experiments*. Springer; 2018:145-200.
22. Kucherenko S, Albrecht D, Saltelli A. Exploring multi-dimensional spaces: a comparison of Latin hypercube and quasi Monte Carlo sampling techniques. *arXiv preprint arXiv:1505.02350*; 2015.
23. Meibody M, Naseh H, Ommi F. Progressive Latin hypercube sampling-based robust design optimisation (PLHS-RDO). *Aust J Mech Eng*. 2022;20(3):660-667.
24. Galetz A, Loukrezis D, Georg N, De Gerssem H, Römer U. An *hp*-adaptive multi-element stochastic collocation method for surrogate modeling with information re-use. *Int. J. Numer. Methods Eng*. 2023;124(12):2902-2930.
25. Gahrooei MR, Paynabar K, Pacella M, Colosimo BM. An adaptive fused sampling approach of high-accuracy data in the presence of low-accuracy data. *IISE Trans*. 2019;51(11):1251-1264.
26. Liu Y, Chen S, Wang F, Xiong F. Sequential optimization using multi-level cokriging and extended expected improvement criterion. *Struct Multidiscip Optim*. 2018;58:1155-1173.
27. Huang H, Liu Z, Zheng H, Xu X, Duan Y. A proportional expected improvement criterion-based multi-fidelity sequential optimization method. *Struct Multidiscip Optim*. 2023;66(2):30.
28. Che Y, Müller J, Cheng C. Dispersion-enhanced sequential batch sampling for adaptive contour estimation. *Qual Reliab Eng Int*. 2022;40(1):131-144.
29. Zhou X, Lin DK, Hu X, Ouyang L. Sequential Latin hypercube design with both space-filling and projective properties. *Qual Reliab Eng Int*. 2019;35(6):1941-1951.
30. Joseph VR, Dasgupta T, Tuo R, Wu CJ. Sequential exploration of complex surfaces using minimum energy designs. *Technometrics*. 2015;57(1):64-74.
31. Kim H, Vastola JT, Kim S, Lu JC, Grover MA. Batch sequential minimum energy design with design-region adaptation. *J Qual Technol*. 2017;49(1):11-26.
32. Qian J, Yi J, Cheng Y, Liu J, Zhou Q. A sequential constraints updating approach for Kriging surrogate model-assisted engineering optimization design problem. *Eng Comput*. 2020;36:993-1009.
33. Jiang C, Qiu H, Gao L, Wang D, YaFng Z, Chen L. Real-time estimation error-guided active learning Kriging method for time-dependent reliability analysis. *Appl Math Modell*. 2020;77:82-98.
34. Wong PK, Tam LM, Ke L. Automotive engine power performance tuning under numerical and nominal data. *Control Eng Pract*. 2012;20(3):300-314.
35. Wong KI, Wong PK, Cheung CS, Vong CM. Modeling and optimization of biodiesel engine performance using advanced machine learning methods. *Energy*. 2013;55:519-528.
36. Turkson RF, Yan F, Ali MKA, Hu J. Artificial neural network applications in the calibration of spark-ignition engines: an overview. *Eng Sci Technol, Int J*. 2016;19(3):1346-1359.
37. Singh A, Krause A, Kaiser WJ. Nonmyopic adaptive informative path planning for multiple robots. Proceedings of the 21st International Joint Conference on Artificial Intelligence. 2009. 1843-1850.
38. Chai R, Niu H, Carrasco J, Arvin F, Yin H, Lennox B. Design and experimental validation of deep reinforcement learning-based fast trajectory planning and control for mobile robot in unknown environment. *IEEE Trans Neural Netw Learn Syst*. 2022;35:5778-5792.

39. Reich J, Sklar E. Robot-sensor networks for search and rescue. *IEEE International Workshop on Safety, Security and Rescue Robotics*. 2006;22.
40. Manuel MP, Faied M, Krishnan M, Paulik M. Robot platooning strategy for search and rescue operations. *Intell Serv Robot*. 2022;15:1-12.
41. Schwager M, Dames P, Rus D, Kumar V. A multi-robot control policy for information gathering in the presence of unknown hazards. *Robotics Research*. Springer; 2017:455-472.
42. Brunke L, Greeff M, Hall AW, et al. Safe learning in robotics: from learning-based control to safe reinforcement learning. *Annu Rev Control, Robot Auton Syst*. 2022;5:411-444.
43. Williams CK, Rasmussen CE. *Gaussian Processes for Machine Learning*. Vol 2. MIT Press; 2006.
44. Sasso F, Coluccia A, Notarstefano G. An empirical Bayes approach for distributed estimation of spatial fields. *IEEE European Control Conference (ECC)*; 2018:2206-2211.
45. Boots BN. *Voronoi (Thiessen) Polygons*. Vol 45. Geo Books; 1986.
46. Rahmati Z, King V, Whitesides S. Kinetic data structures for all nearest neighbors and closest pair in the plane. *Proceedings of the twenty-ninth annual symposium on computational geometry*. 2013:137-144.
47. Watson DF. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *Comput J*. 1981;24(2):167-172.
48. Lee DT, Schachter BJ. Two algorithms for constructing a Delaunay triangulation. *Int J Comput Inf Sci*. 1980;9(3):219-242.
49. Van Kreveld M, Schwarzkopf O, Berg dM, Overmars M. *Computational Geometry Algorithms and Applications*. Springer; 2000.
50. Srinivas N, Krause A, Kakade SM, Seeger MW. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Trans Inf Theory*. 2012;58(5):3250-3265.
51. Le Gratiet L, Garnier J. Asymptotic analysis of the learning curve for Gaussian process regression. *Mach Learn*. 2015;98(3):407-433.
52. Lam CQ. *Sequential Adaptive Designs in Computer Experiments for Response Surface Model Fit*. PhD thesis. The Ohio State University; 2008.
53. Zhan D, Xing H. Expected improvement for expensive optimization: a review. *J Global Optim*. 2020;78(3):507-544.
54. Sauer A, Gramacy RB, Higdon D. Active learning for deep Gaussian process surrogates. *Technometrics*. 2023;65(1):4-18.
55. Binois M, Huang J, Gramacy RB, Ludkovski M. Replication or exploration? Sequential design for stochastic simulation experiments. *Technometrics*. 2019;61(1):7-23.
56. Rasmussen CE, Nickisch H. Documentation for GPML Matlab Code version 4.2. 2018. <http://www.gaussianprocess.org/gpml/code/matlab/doc/>
57. Quan N, Yin J, Ng SH, Lee LH. Simulation optimization via kriging: a sequential search using expected improvement with computing budget constraints. *IIE Trans*. 2013;45(7):763-780.
58. Turner R, Turner MR. deldir: Delaunay Triangulation and Dirichlet (Voronoi) Tessellation. 2024. R package version 2.0-4.
59. Picheny V, Ginsbourger D, Roustant O, Binois M, Marmin S, Wagner T. DiceOptim: Krining-based optimization for computer experiments. 2021. R package version 2.1.1.
60. Booth AS. deepgp: Deep Gaussian Processes using MCMC. 2023. R package version 1.1.
61. Wang D, Joseph VR. mined: Minimum Energy Design. 2022. R package version 1.0-3.
62. Reşitoğlu İA, Altinişik K, Keskin A. The pollutant emissions from diesel-engine vehicles and exhaust aftertreatment systems. *Clean Technol Environ Policy*. 2015;17(1):15-27.
63. Swanton C, Boffetta P, Peston R, Soria JC. Environmental emissions, public health and lung cancer risk. *Ann. Oncol.* 2016;27(2):211-212.
64. Zhao J, Wong PK, Xie Z, Ma X, Hua X. Design and control of an automotive variable hydraulic damper using cuckoo search optimized PID method. *Int J Automot Technol*. 2019;20:51-63.
65. Aliramezani M, Norouzi A, Koch CR. Support vector machine for a diesel engine performance and NOx emission control-oriented model. *IFAC-PapersOnLine*. 2020;53(2):13976-13981.
66. Edmonds M, Yi J. Efficient multi-robot inspection of row crops via kernel estimation and region-based task allocation. *IEEE International Conference on Robotics and Automation (ICRA)*; 2021:8919-8926.
67. Lindqvist B, Karlsson S, Koval A, et al. Multimodality robotic systems: Integrated combined legged-aerial mobility for subterranean search-and-rescue. *Rob Autom Syst*. 2022;154:104134.
68. Cortes J, Martinez S, Karatas T, Bullo F. Coverage control for mobile sensing networks. *IEEE Trans Rob Autom*. 2004;20(2):243-255.
69. Bullo F, Cortes J, Martinez S. *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Vol 27, Princeton University Press; 2009.
70. Chen J, Li T, Wang J, Silva dCW. WSN sampling optimization for signal reconstruction using spatiotemporal autoencoder. *IEEE Sens J*. 2020;20(23):14290-14301.
71. Santos M, Diaz-Mercado Y, Egerstedt M. Coverage control for multirobot teams with heterogeneous sensing capabilities. *IEEE Robot Autom Lett*. 2018;3(2):919-925.
72. Breitenmoser A, Schwager M, Metzger JC, Siegwart R, Rus D. Voronoi coverage of non-convex environments with a group of networked robots. *IEEE International Conference on Robotics and Automation*; 2010:4982-4989.
73. Schwager M, Vitus MP, Rus D, Tomlin CJ. Robust adaptive coverage for robotic sensor networks. *Robotics Research*. Springer; 2017:437-454.
74. Wilson S, Glotfelter P, Wang L, et al. The Robotarium: globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems. *IEEE Control Syst Mag*. 2020;40(1):26-44.
75. Aurenhammer F. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Comput Surv. (CSUR)*. 1991;23(3):345-405.

76. Lederer A, Umlauf J, Hirche S. Posterior variance analysis of Gaussian processes with application to average learning curves. *arXiv preprint arXiv:1906.01404*; 2019.

**How to cite this article:** Benevento A, Ahadi P, Gupta S, Pacella M, Paynabar K. Sequential sampling for functional estimation via SIEVE. *Qual Reliab Eng Int.* 2024;40:3253–3279. <https://doi.org/10.1002/qre.3557>

## APPENDIX A: EMPIRICAL BAYES FRAMEWORK

In this section, we want to show a probabilistic model, based on a centralized version of the Empirical Bayes paradigm proposed in Ref. 44, that allows us to suitably fuse the performed measurements in order to estimate the spatial field. From the assumption that noises are i.i.d. Gaussian variables, it follows that

$$p(\mathbf{y}_t | \mathbf{f}_t) = \prod_{i=1}^t p(y_i | f_i), \quad p(y_i | f_i) \sim \mathcal{N}(f_i, \sigma^2).$$

Without loss of generality, we can assume that the function  $f$  is a Gaussian process with parametrized mean and covariance functions, namely  $\mu(\cdot, \alpha)$  and  $K(\cdot, \cdot, \beta)$ , respectively. The parameters  $\alpha_t$  and  $\beta_t$  are estimated from the data through Maximum Likelihood (ML) as follows:

$$(\alpha_t, \beta_t) = \operatorname{argmax}_{\alpha, \beta} p(\mathbf{y}_t; \alpha, \beta).$$

The solution of this problem can be achieved by maximizing a likelihood function, as, for example, the log-likelihood function proposed in Ref. 43:

$$\log(p(\mathbf{y}_t; \alpha, \beta)) := -\frac{1}{2}(\boldsymbol{\mu}_t(\alpha) - \mathbf{y}_t)^\top (K_{\mathbf{xx},t}(\beta) + \sigma^2 I)^{-1}(\boldsymbol{\mu}_t(\alpha) - \mathbf{y}_t) - \frac{1}{2} \log |K_{\mathbf{xx},t}(\beta) + \sigma^2 I| - \frac{t}{2} \log(2\pi) \quad (\text{A1})$$

where  $\boldsymbol{\mu}_t(\alpha) = (\mu(\mathbf{x}_1; \alpha), \dots, \mu(\mathbf{x}_t; \alpha))^\top$ ,  $I \in \mathbb{R}^{t \times t}$  is the identity matrix and  $K_{\mathbf{xx},t}(\beta) \in \mathbb{R}^{t \times t}$  is the covariance matrix with entries  $[K_{\mathbf{xx},t}]_{si} = K(\mathbf{x}_s, \mathbf{x}_i; \beta)$ .

As a consequence, the ML estimation of  $\alpha$  and  $\beta$  at step  $t$  is given by

$$(\alpha_t, \beta_t) := \operatorname{argmin}_{\alpha, \beta} (\boldsymbol{\mu}_t(\alpha) - \mathbf{y}_t)^\top (K_{\mathbf{xx},t}(\beta) + \sigma^2 I)^{-1}(\boldsymbol{\mu}_t(\alpha) - \mathbf{y}_t) + \log |K_{\mathbf{xx},t}(\beta) + \sigma^2 I|. \quad (\text{A2})$$

Once the parameters  $\alpha_t$  and  $\beta_t$  are available, we can compute  $\mu(\cdot; \alpha_t)$  at any regression point  $\mathbf{r}_i$  and the covariance  $K(\cdot, \cdot; \beta_t)$  in any pair of regression points  $\mathbf{r}_i, \mathbf{r}_j$ . In particular, the mean and the covariance functions depend on the measurement points as  $\alpha_t$  and  $\beta_t$  depend on them. With the mean and covariance values at regression points at hand the posterior function, which is a multivariate Gaussian, can be evaluated. Defining

$$\begin{aligned} f_c^{\mathbf{r}} &= f(\mathbf{r}_c), \quad \mathbf{f}^{\mathbf{r}} = (f_1^{\mathbf{r}}, \dots, f_R^{\mathbf{r}})^\top, \\ \boldsymbol{\mu}_{c,t}^{\mathbf{r}} &= \mu(\mathbf{r}_c; \alpha_t), \quad \boldsymbol{\mu}_t^{\mathbf{r}} = (\mu_{1,t}^{\mathbf{r}}, \dots, \mu_{R,t}^{\mathbf{r}})^\top \end{aligned}$$

the MAP estimator in the regression points is given by

$$\hat{\mathbf{f}}_t(\mathbf{r}) = \operatorname{argmax}_{\mathbf{f}^{\mathbf{r}} \in K_{\mathbf{r}}^{\mathbf{r}}} p(\mathbf{f}^{\mathbf{r}} | \mathbf{y}_t; \alpha_t, \beta_t), \quad (\text{A3})$$

where each  $K_{\mathbf{r}}$  is a compact set. In particular, the MAP estimator can be computed at each point  $\mathbf{r}$  with respect to the mean and the covariance functions (see Ref. 44 for more details):

$$\hat{\mathbf{f}}_t(\mathbf{r}) = \boldsymbol{\mu}_t^{\mathbf{r}} + K_{\mathbf{xr},t}^\top (K_{\mathbf{xx},t} + D_t)^{-1}(\mathbf{y}_t - \boldsymbol{\mu}_t)$$

where  $K_{\mathbf{xx},t} \in \mathbb{R}^{t \times t}$  and  $K_{\mathbf{xr},t} \in \mathbb{R}^{t \times R}$  are matrices with entries  $[K_{\mathbf{xx},t}]_{sr} = K(\mathbf{x}_s, \mathbf{x}_r)$  and  $[K_{\mathbf{xr},t}]_{sj} = K(\mathbf{x}_s, \mathbf{r}_j)$  and  $D_t = \sigma^2 I_{T,T}$ .

## APPENDIX B: DELAUNAY TESSELLATION

Computational geometry is a branch of geometry that aims of solving practical problems by exploiting basic geometrical objects and finding structures concealed in a set of data.<sup>75</sup> Numerous applications are in computer graphics, robotics, geographic information systems, and finite elements analysis.<sup>49</sup> The problem of analyzing a branch of data is often faced via computational geometry, in particular, by inducing a subdivision of the domain under consideration into regions defined through the data. An important geometric structure typically used for the analysis of data is domain triangulation. Triangulation or tessellation is a way to subdivide the domain in multi-dimensional triangles, even known as  $d$ -simplices. In general, this can be obtained in different ways. One of the approaches to solve the problem is the Delaunay tessellation. It has the property that the circumsphere of each  $d$ -simplex in the triangulation does not contain any point in its interior. The circumsphere of a  $d$ -simplex is a hypersphere in  $\mathbb{R}^{d+1}$  passing through all the vertices of the simplex. The main advantage of the Delaunay triangulation is that it simultaneously maximizes the number of triangles and produces triangles that are as equiangular as possible. This implies that triangle edge lengths are minimized. In the specific context of this paper, minimizing the lengths of the edges reflects on minimizing the distance between the pairs of data considered in the maximization step (1).

In this section, we summarize the steps of a known iterative algorithm to compute the Delaunay tessellation given  $N$  points in a  $d$ -dimensional Euclidean space.<sup>47</sup> The algorithm assumes an initial configuration of  $d + 1$  points forming a  $d$ -simplex. Formally,  $d + 1$  points in  $\mathbb{R}^d$  form a  $d$ -simplex if and only if they are affinely independent, that is, if they do not belong to a unique  $d - 1$ -plane. Note that the probability of  $d + 1$  points to be affinely dependent is almost surely 0. In particular, the algorithm presented in this paper requires a set of available points that are the vertices of a convex domain. As a  $d$ -simplex is the convex envelope of its vertices, it is enough to take  $d + 1$  vertices of the domain to avoid degeneracies in the algorithm initialization. Given a  $d$ -simplex, there exists a unique circumsphere passing through all its vertices. During the implementation of the algorithm, a list of empty circumspheres is maintained. The circumspheres describing any current triangulation must remain empty, that is, they should not contain any other query point in the interior. When a point is added in the configuration, the circumspheres containing the new point are flagged. All the edges belonging to two or more flagged circumspheres are removed from the triangulation, while the new point is connected with all the points lying on the flagged circumspheres. Note that, from a practical point of view, checking whether a point belongs or not to a circumsphere is straightforward: maintain the list of the circumcenters of each circumsphere as well as its radius, and then measure the distance between the new point and the circumcenters. A distance less than or equal to the radius implies that the new point belongs to the circumsphere. The computational complexity of the triangulation of a set of  $N$  points is  $O(N^{(2d-1)/d})$ .<sup>47</sup> The number of edges of the triangles, corresponding with the number of pairs of neighbors points, is linear in  $N$ .<sup>48</sup> We defer the reader to Ref. 47 for further technical details on computing Delaunay triangulations iteratively such as the efficiency of the method. The pseudo-code describing this algorithm is summarized in Algorithm 1, while a visualization of the algorithm in a two-dimensional domain is shown in Figure 4.

## APPENDIX C: PROOF OF THE THEOREM

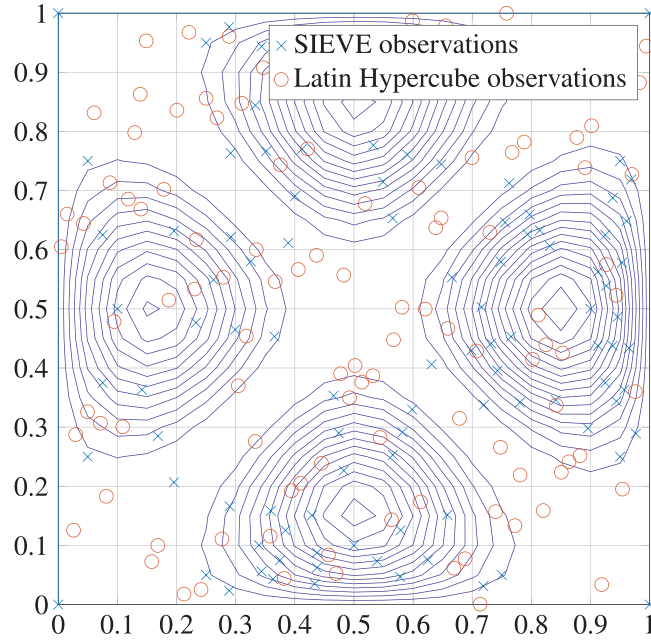
In this section, we provide proof of Theorem 1. First, we need some preparatory results. We remind that a kernel function  $K(\cdot, \cdot)$  is *isotropic* if it only depends on the distance between its arguments:  $K(x, y) = K(\|x - y\|)$ .

The following proposition shows a theoretical result that bounds the error for the posterior variance. The key idea in deriving the following upper bound is that the posterior variance is likely small in the interval of two close observations in the domain, and it increases as observations get farther apart.<sup>43</sup> Thus, a natural choice is considering only sampled data close to a test point as more data are observed.

**Lemma C.1** Corollary 3.1 in Ref. 76. Consider a  $\hat{f} \sim GP$  with isotropic, decreasing covariance kernel  $K(\cdot)$ , an input training data set  $\mathbb{D}_T^x = \{x_t\}_{t=1}^T$  and observation noise variance  $\sigma^2$ . Let  $\mathbb{B}_\rho(\mathbf{r}) = \{x' \in \mathbb{D}_T^x : \|x' - \mathbf{r}\| \leq \rho\}$  denote the training data set restricted to a ball around  $x$  with radius  $\rho$ . Then, for each  $\mathbf{r} \in X$ , the posterior variance is bounded by

$$\sigma_{t,f}^2(\mathbf{r}) \leq K(0) - \frac{K^2(\tau)}{K(0) + \frac{\sigma^2}{|\mathbb{B}_\rho(\mathbf{r})|}}, \quad (\text{C1})$$

where  $\tau$  is the minimal Euclidean distance between  $\mathbf{r}$  and the set of the sampled points.



**FIGURE D1** Distribution of the observed points in the two-dimensional domain, sampled with SIEVE and with the Latin Hypercube algorithm.

The previous result shows an upper bound for the posterior variance depending on the number of sampled points. The consistency Theorem 1 shows that the upper bound (C1) goes to 0 when the points are collected via SIEVE. We are now ready to give proof of the theorem.

*Proof of Theorem 1, Section 4.* We want to prove that the number of points in a ball of radius  $\rho|\mathbb{B}_\rho(\mathbf{r})|$  goes to infinity as  $t$  goes to infinity, that is, as more and more observations are added in the SIEVE process. By construction, SIEVE always collects new points that have never been sampled before, thus the number of sampled points is strictly increasing. Fix  $\rho > 0$  and suppose that there exist two neighbors  $x', x''$  in the set of sampled data such that  $d(x', x'') \geq \rho > 0$ . By contradiction, suppose that the midpoint among them is never sampled. This implies that the points are sampled in the other parts of the domain. Due to Lipschitz continuity,  $\operatorname{argmax}_{i,j \in \mathcal{P}_t \setminus \{x', x''\}} \|\mathbf{x}_i - \mathbf{x}_j\| + \lambda|y_i - y_j| \leq (\lambda L + 1)\|\mathbf{x}_i - \mathbf{x}_j\|$  goes to 0 when  $t$  goes to infinity. Therefore, there is a point such that  $\rho$  becomes greater than  $\operatorname{argmax}_{i,j \in \mathcal{P}_t \setminus \{x', x''\}} \|\mathbf{x}_i - \mathbf{x}_j\| + \lambda|y_i - y_j|$  and the midpoint  $x' + x''/2$  has to be sampled.

Thus,  $|\mathbb{B}_\rho(\mathbf{r})|$  is increasing and, when a new point in the ball is considered, the minimal Euclidean distance between  $\mathbf{r}$  and the set of the sampled points  $\tau$  is halved and will go to 0 by iterating the process. This implies that

$$\lim_{t \rightarrow \infty} \sigma_{t, \hat{f}}^2(\mathbf{r}) \leq \lim_{t \rightarrow \infty} K(0) - \frac{K^2(\tau)}{K(0) + \frac{\sigma^2}{|\mathbb{B}_\rho(\mathbf{r})|}} = K(0) - \frac{K^2(0)}{K(0) + 0} = 0,$$

that concludes the proof.  $\square$

#### APPENDIX D: ADDITIONAL SIMULATION RESULTS

In this section, we offer a couple of additional simulation results. In particular, we compare SIEVE with the one-shot Latin Hypercube algorithm and with a simplified version of SIEVE that performs only the exploration of the domain.

We consider the function previously considered in Section 5:

$$f(x_1, x_2) = -5(1 - (2x_1 - 1)^2)(1 - (2x_2 - 1)^2) \times (4 + 2x_1 - 1) \left( 0.05^{(2x_1 - 1)^2} - 0.05^{(2x_2 - 1)^2} \right)^2.$$

Figure D1 shows the distribution of points sampled by using 100 iterations of SIEVE and by sampling 100 points by the Latin Hypercube algorithm. As expected, the Latin Hypercube algorithm samples the points uniformly in the domain,



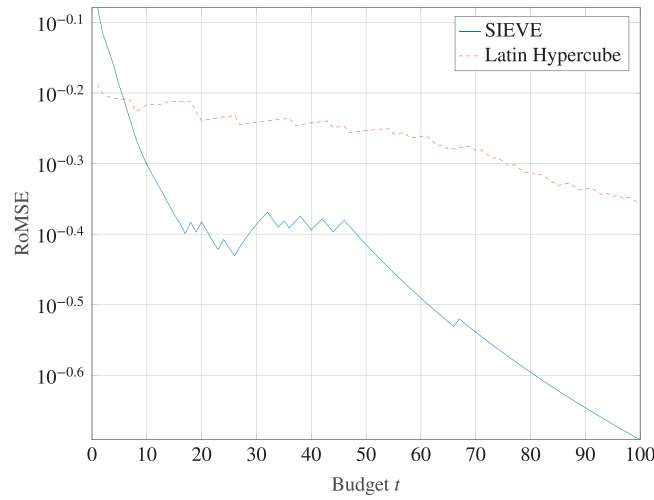


FIGURE D2 Comparison between the convergence of SIEVE and the Latin Hypercube algorithm in terms of the normalized RoMSE.

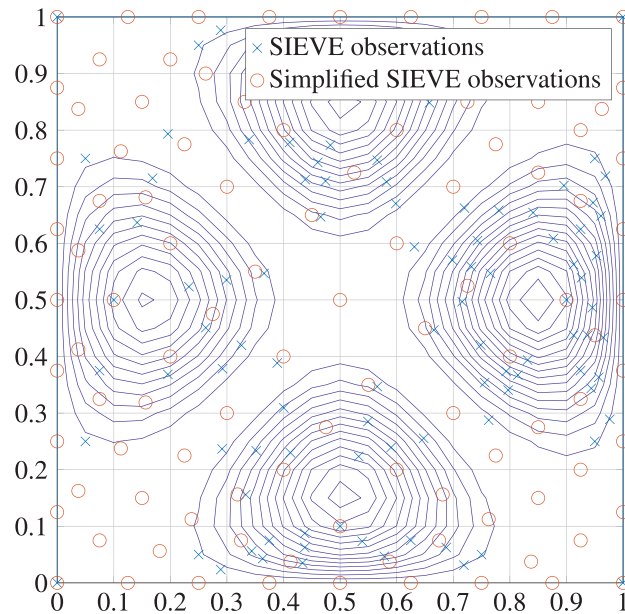


FIGURE D3 Distribution of the observed points in the two-dimensional domain, sampled with SIEVE and with a simplified version of it.

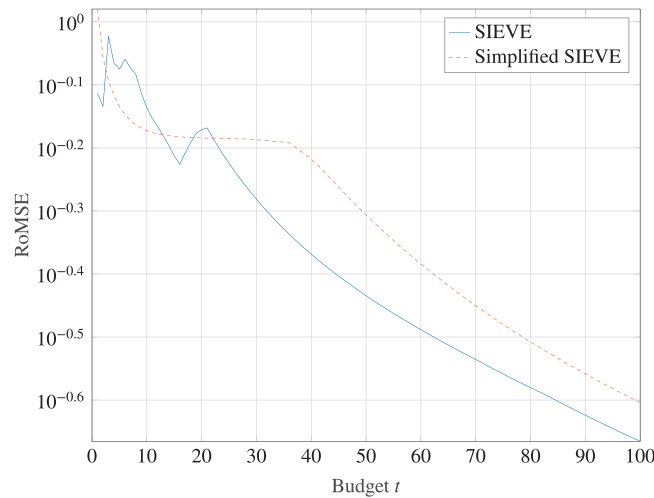


FIGURE D4 Comparison between the convergence of the two methods in terms of the normalized RoMSE.

while SIEVE tends to avoid the area of the domain less informative, that is, in this example, where the function is 0. The advantage in terms of accuracy of choosing an observation-based algorithm is evident in Figure D2.

In the second example, we consider a simplified version of SIEVE, where we remove the exploitation of the observed values of the unknown function. In particular, we consider only the distribution of the observed data in the domain by maximizing  $\max_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|$ , instead of Equation (1). The distribution of the sampled points is shown in Figure D3. The figure shows the tendency of SIEVE to exploit the data, while its simplified counterpart tends to explore the domain and to sample points uniformly distributed. The benefits of the exploitation part of the algorithm are shown in Figure D4 in terms of accuracy.

## AUTHOR BIOGRAPHIES



**Alessia Benevento** is a post-doc Researcher in the Department of Economics at the University of Salento, Italy. She had her Bachelor's Degree and Master's Degree in Mathematics and her Ph.D in Complex Systems Engineering from the University of Salento. The main topics of her research interests include defining clustering techniques to merge temporal and spatial dependence in the analysis of time series and exploiting Gaussian Processes for learning and optimization. Her works find applications in the context of mobile robotics and in the environmental context.



**Pouya Ahadi** is a Ph.D. student at the H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA. He received a B.S. degree from the Sharif University of Technology, Tehran, Iran, in 2018 and an M.Sc. degree in Industrial Engineering from Oklahoma State University in 2021. His research interests are focused on active machine learning, adaptive sampling, and high-dimensional data analysis.



**Swati Gupta** is an Assistant Professor at the MIT Sloan School of Management in the Operations Research and Statistics Group since July 2023. Prior to this, she held a Fouts Family Early Career Professorship as an Assistant Professor at the Stewart School of Industrial and Systems Engineering at Georgia Tech from 2018 to 2023. She also served as the lead of Ethical AI in the NSF AI Institute on Advances in Optimization, awarded in 2021. She received a Ph.D. in Operations Research from MIT in 2017, and joint Masters and B.Tech in Computer Science from IIT Delhi in 2011. Her research interests include optimization and machine learning, with a focus on algorithmic fairness. Her work spans various domains such as hiring, admissions, e-commerce, quantum optimization, and energy. She has received the NSF CAREER Award, the Class of 1934: Student Recognition of Excellence in Teaching in 2020 and 2021 at Georgia Tech, the JP Morgan Early Career Faculty Recognition in 2021, the NSF CISE Research Initiation Initiative Award in 2019, and the Google Women in Engineering Award (India) in 2011. She was also awarded the prestigious Simons-Berkeley Research Fellowship in 2017–2018. Her research and students have received recognition at various venues like INFORMS Doing Good with OR 2022, INFORMS Undergraduate Operations Research 2018, INFORMS Computing Society 2016, and INFORMS Service Science Student Paper 2016.



**Massimo Pacella** is an Associate Professor in the Department of Engineering for Innovation at the University of Salento, Italy. He received an M.Sc in Computer Engineering from the University of Lecce, Italy, and a Ph.D in manufacturing and production systems from the Polytechnic of Milan, Italy. He was awarded a Fulbright Fellowship, and he was a research scholar at the Department of Industrial and Operations Engineering, University of Michigan, USA. His major research interests are in functional data processing and profile monitoring, including applied and methodological aspects of machine learning and statistical modeling, integrated with engineering principles. The primary applications of his research are in manufacturing and automotive. He is a member of the Italian Association for Manufacturing Technology (AITeM).



**Kamran Paynabar** is a Fouts Family Professor in the H. Milton Stewart School of Industrial and Systems Engineering at Georgia Tech. His research interests comprise both applied and methodological aspects of statistical machine learning with the focus on the analysis of high-dimensional data for predictive modeling, monitoring, diagnosis, and prognosis. He served as the president of Quality and Reliability divisions of IISE and INFORMS. He is an Associate Editor for Technometrics, INFORMS Journal of Data Science, a Department Editor for IISE-Transactions, and a member of editorial board for JQT.