Reliable emulation of complex functionals by active learning with error control

Xinyi Fang,[1] Mengyang Gu,[1, a)] and Jianzhong Wu[2, b)]
[1)]Department of Statistics and Applied Probability, University of California, Santa Barbara, CA 93106, USA
[2)]Department of Chemical and Environmental Engineering, University of California, Riverside, CA 92521, USA

(Dated: 2 November 2022)

A statistical emulator can be used as a surrogate of complex physics-based calculations to drastically reduce the computational cost. Its successful implementation hinges on an accurate representation of the nonlinear response surface with a high-dimensional input space. Conventional "space-filling" designs, including random sampling and Latin hypercube sampling, become inefficient as the dimensionality of the input variables increases, and the predictive accuracy of the emulator can degrade substantially for a test input distant from the training input set. To address this fundamental challenge, we develop a reliable emulator for predicting complex functionals by active learning with error control (ALEC). The algorithm is applicable to infinite-dimensional mapping with high-fidelity predictions and a controlled predictive error. The computational efficiency has been demonstrated by emulating the classical density functional theory (cDFT) calculations, a statistical-mechanical method widely used in modeling the equilibrium properties of complex molecular systems. We show that ALEC is much more accurate than conventional emulators based on the Gaussian processes with "space-filling" designs and alternative active learning methods. Besides, it is computationally more efficient than direct cDFT calculations. ALEC can be a reliable building block for emulating expen-sive functionals owing to its minimal computational cost, controllable predictive error, and fully automatic features.

I. INTRODUCTION

Theoretical research in chemistry and materials science is often hampered by computational bottlenecks in applying complex physical models to predict the microscopic structure and physicochemical properties of many-body systems. Statistical and machine learning (ML) methods, such as Gaussian process (GP) regression, basis expansion methods, and neural network models, have been widely used as a surrogate of complex physical models to emulate the atomic force fields, density functionals, chemical reactions, and diverse properties of materials and chemical systems[1–8]. Here a typical aim of a ML approach is to learn the map with a high dimensional input or a functional input from observations. Trained on a set of pre-specified inputs, a statistical emulator can obtain accurate predictions when a test point is close to the training inputs[9–11].

Constructing a statistical emulator often starts with selecting a set of "space-filling" samples in the input space, such as uniform samples or Latin hypercube samples (LHS)[12]. The statistical emulator will be trained based on the outputs of the simulation at these pre-selected inputs. The idea is to ensure that for each test point in the input space, there are a sufficiently large number of inputs near this test point, where the outcomes (such as the potential energy or atomic force in emulating the first-principles calculations) were observed. Using the space-filling samples and assuming a set of regularity conditions, the predictive error of some ML methods, such as the GP regression, is guaranteed to converge to zero, when the sample size goes to infinity[13]. The space-filling samples have two main drawbacks in emulating physics-based calculations such as molecular dynamics simulation (MD). First, the input values, such as atomic positions or pairwise inter-atomic distances in MD, may not be directly controlled. Second, physics-based modeling of molecules and materials often involves a high or infinite dimensional input space, such as the external potential function and atomic configurations. If the statistical emulator is trained in a small parameter subspace, the predictive accuracy of the emulator can be dramatically degraded outside the subspace where no training input is available. This scenario often occurs when the outcome is required in a slightly different system or at another experimental condition, such as different temperatures or pressures, whereas the statistical emulator becomes inaccurate if it is only trained on one set of conditions. Obtaining accurate predictions of any test input with a controlled predictive error is important to bypass expensive computations based on complex physical models.

To address the fundamental difficulty in predicting functions and functionals with a high-dimensional input space, active learning has become one of the most promising techniques to sequentially reinforce the emulation of physics-based modeling[14]. The active learning approach can be implemented in two broad scenarios. The first one is an online scenario or "on-the-fly" prediction, where the test inputs sequentially come and one does not

[a)]Corresponding author: mengyang@pstat.ucsb.edu
[b)]Corresponding author: jwu@engr.ucr.edu

know what future input needs to be predicted[15–17]. The online prediction scenario has many applications, such as Bayesian optimization[18,19] and model calibration[20,21], where the outcome of the simulation needs to be sequentially predicted for every sampled input. The second scenario is to sequentially select the samples to run a computer simulation for predicting the entire input space, where the uncertainty of the emulator is often used for selecting the next design run[22]. In this work, we focus on the online prediction scenario and test our approach using the classical density functional theory (cDFT) calculations for one-dimensional (1D) hard-rod systems, with different external potentials. The availability of exact densities and free-energy functional for various 1D systems of hard rods allows us to validate the efficiency of the surrogate model with the ground truth. Here the fundamental difficulty is on a user-specified external potential–an arbitrary functional input with infinite dimensions. For constructing statistical emulators, we assume that no parametric form of this functional input is available. We demonstrate that the new integrated approach provides a high-fidelity prediction of particle density profiles and grand potentials with a controlled error in the probabilistic sense, more accurate than a GP emulator with conventional "space-filling" designs, and it is more computationally scalable than direct cDFT calculations.

We highlight the key novelty of this work in developing the surrogate model and error control criteria for functional mapping. First, while some popular criteria, such as the D-optimality, were introduced in the active learning framework[15,23], the threshold of determining whether a test input can be reliably predicted may be hard to choose, as the threshold may not be directly related to the error in prediction. In practice, one may need to tune the threshold for different systems for a given error bound. Here, we use the internal uncertainty assessment of the GP emulator to decide whether a test input can be predicted precisely. For any given threshold of predictive error and probability tolerance bound, our approach automatically defines the criterion to control the predictive error below the threshold with a large probability satisfying the probability tolerance bound (see Section III B). We demonstrate that our approach can identify whether an upcoming input can be reliably predicted, and thus it has much better performance than the conventional statistical emulation by training the emulator with a space-filling design, such as random sampling. Furthermore, we derive an iterative formula that reduces the computational complexity for Gaussian process models with augmented samples. In each iteration, updating our algorithm only requires $O(n^2)$ operations, while a direct approach requires $O(n^3)$ operations. Other emulation approaches, such as those enforcing the invariance symmetries, can be easily included in our integrated framework for reliable predictions with a controlled error.

The remainder of the paper is structured as follows: In Section II, we provide a brief overview of the classical density functional theory in the context of the 1-dimensional (1D) hard-rod model used in this work. In Section III A, we introduce the parallel partial Gaussian process for emulating physics-based calculations with vectorized output on massive grid points, such as particle densities from cDFT calculations. Then, in Section III B, we introduce our proposed method to control the predictive error. The computationally scalable update of the algorithm and its computational cost are discussed in Sections III C and III D, respectively. The numerical comparison in Section IV demonstrates that our approach outperforms the conventional GP emulators with random-sample designs and the active learning approach with the D-optimality criterion. We conclude the paper along with perspectives on some future research directions in Section V.

II. CLASSICAL DENSITY FUNCTIONAL THEORY

Classical density functional theory (cDFT) is a statistical-mechanical method that describes the thermodynamic properties of equilibrium systems in terms of the density profiles of individual particles[24,25]. In a nutshell, cDFT calculations are based on the minimization of a grand potential functional that, for systems consisting of only one type of particle, can be expressed as

$$[] = F_{id}[] + F_{ex}[] + \int dr(r)[V^{ext}(r) \quad]; \qquad (1)$$

where (r) denotes the particle density at position r, $F[]_{id}$ is the free-energy functional of an ideal gas, and $F_{ex}[]$ is called the excess free-energy functional, $V^{ext}(r)$ denotes a one-body external potential, and is the chem-ical potential of the particles. The ideal-gas functional has an exact form

$$F_{id}[] = \int dr(r)f\ln[(r)^3] \quad 1g; \qquad (2)$$

where is the thermal wavelength and $= 1=(k_B T)$ with k_B being the Boltzmann constant and T the ab-solute temperature. While T and are thermodynamic variables that define the equilibrium condition, and $F_{ex}[]$ depend on the intrinsic properties of the system such as the particle mass and inter-particle potential. In this paper, we set $= = 1$ as the units of energy and length.

One key premise of cDFT calculations is that $V^{ext}(r)$ is uniquely determined by (r) such that, given an analytical expression for $F_{ex}[]$, (r) can be solved by minimizing
[] and, subsequently, all thermodynamic prop-erties of the system can be derived. The equilibrium density profile minimizes
and thus satisfies the Euler-Lagrange equation:

$$(r) = \exp \quad F_{ex}= \quad V^{ext}(r) : \qquad (3)$$

Eq.(3) provides a functional map between the one-body density (r) and the one-body external potential $V^{ext}(r)$.

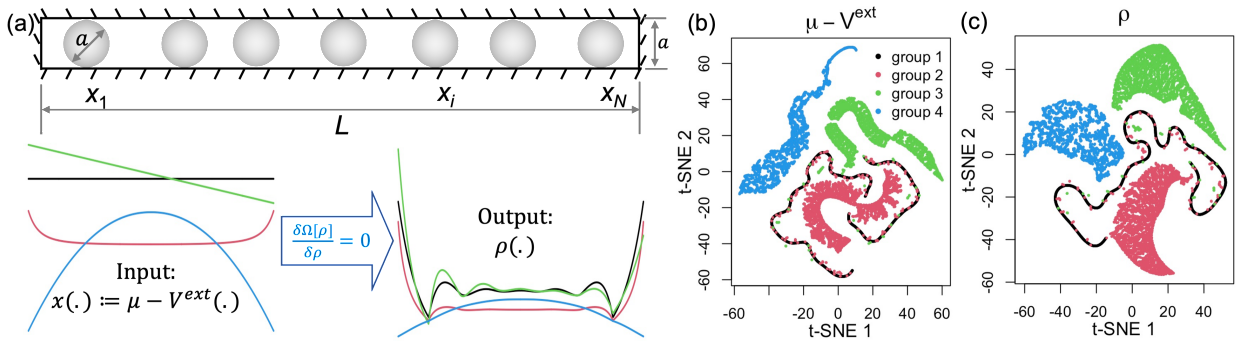

FIG. 1. (a) Functional learning in the context of cDFT for one-dimensional hard rods of uniform size a in an external field $V^{ext}(:)$ defined in a domain of size L. (b)-(c) t-distributed stochastic neighbor embedding (t-SNE) for visualization of input $V^{ext}(:)$ (middle panel) and output (:) (right panel). For each group, 2000 density profiles are generated based on different combinations of and parameters in $V^{ext}(:)$.

Because in general F_{ex} is a complicated functional of (r), solving (r) from the Euler-Lagrange equation is often computationally demanding. Besides, the formulation of an accurate expression for F_{ex} is a formidable task for complex molecular systems. In this work, we demonstrate that statistical emulation will be helpful to solve both problems.

For proof of concept, we consider a statistical emulator of cDFT for one-dimensional (1D) systems consisting of identical hard rods (HR). For such systems, the excess free-energy functional, thus the grand potential, is exactly known[26]. The analytical results were derived decades ago and are reproduced in Appendix A. Given any external potential $V^{ext}(r)$ and chemical potential , the density profile of hard rods can be numerically solved (e.g., by Picard iteration) from Eq. (3), and it is plugged into Eqs. (3), (A2), and (A3) for computing the grand potential and other thermodynamic quantities.

For a given system defined by and $V^{ext}(r)$, statistical emulation aims to predict the particle density and free-energy functionals as in direct cDFT calculations (illustrated in Fig. 1(a)). Previous work in cDFT[5,27,28] and Kohn-Sham density functional theory (KS-DFT)[29,30] demonstrate the performance of ML models with a single parametric form for the external potential with different choices of parameters. In this work, we show that the active learning procedure is applicable to multiple classes of external potential similar to cDFT calculations with the same intrinsic Helmholtz energy functional. We demonstrate that, when presented with a new class of external potentials, the model can discern whether this new input can be accurately predicted. As mentioned above, functional mapping represents one of the biggest obstacles in statistical emulation as well as machine learning. Filling the entire functional input space directly with a limited number of simulation runs, such as random sampling or the LHS, is not generally possible. When the external potential function is away from the training input set, both statistical emulators and machine learning methods become inaccurate. In this work, we provide a solution to this fundamental problem by integrating direct cDFT calculations with a Gaussian process (GP) emulator through an active-learning framework. We derive a probabilistic bound to control the overall prediction error of the particle density based on the internal assessment of the uncertainty from the statistical emulator, making our approach applicable to functional learning in infinite dimensional spaces, as well as reducing the computational costs for inputs that can be accurately predicted. Here our algorithm can reliably learn multiple classes of external potentials, while earlier applications of ML methods in both KS-DFT (e.g.[30]) and cDFT (e.g.[5,28]) only demonstrate the performance of their models on one particular class of the external potential. Besides, the active learning scheme developed in this work enables the ML approaches to discern whether outcomes of a test input can be reliably predicted. This ability allows us to control predictive errors as well as substantially reduce the computational cost.

In Fig. 1(b) and (c), the input $V^{ext}(:)$ and out-put (:) of the statistical emulator are projected onto two t-distributed Stochastic Neighborhood Embedding (t-SNE) factors as commonly used for visualizing high-dimensional data[31]. It is evident that the inputs generated from different groups of input functions form different clusters. Similar patterns can be observed for the corresponding particle densities. In addition, group 1 forms a curve in Fig. 1(b) due to the fact that the only change of the input in group 1 is , which only results in a shift in the input space, making group 1 mapping the simplest to learn for the statistical emulator, as to be demonstrated later in the section on numerical results. We noticed several points in groups 2 and 3 are close to those in group 1. These points correspond to samples whose input parameters are close to 0 and are thus nearly equivalent to group 1's inputs. These patterns demonstrate the input-output relationship can be captured by distance metrics, as the smaller distance between chemical potentials leads to the higher similarity between the particle densities. This validates the use of kernels in

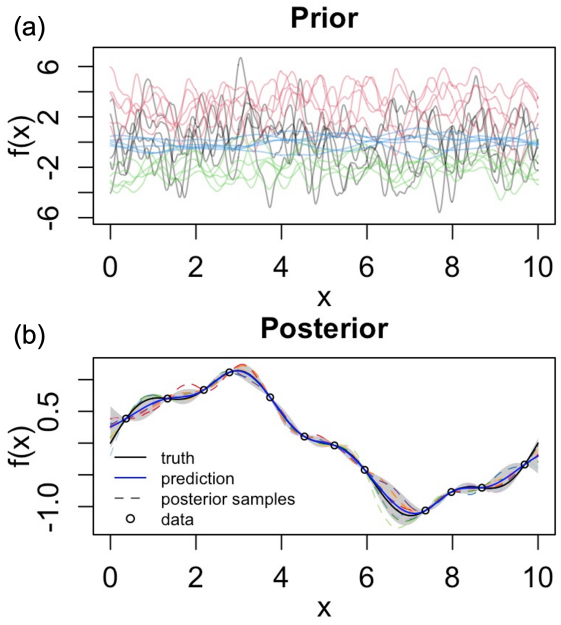

FIG. 2. Illustration of Bayesian updating in a GP emulator. (a) Prior samples from 4 groups (indicated by different colors) with constant mean and Matén kernel in (4). The different parameters are specified as = 0:2; = 2; = 0 (black curves); = 0:25; = 1:5; = 3 (red curves); = 0:33; = 1; = 2 (green curves); = 1; = 0:5; = 0 (blue curves). (b) The truth (black curve), the predictive mean (blue curve), the 95% predictive credible interval (grey region), and 10 poste-rior predictive samples (colored dashed lines) after observing 12 data points (black circles).

the Gaussian process to map the distance between inputs (chemical and external potentials) to the correlation between output (particle densities), as the smaller distance in the input space indicates a larger correlation or similarity between outputs. Unlike the dimension reduction approach shown in Fig. 1, we will model the entire density profile jointly in a statistical emulator to encode all information.

III. A RELIABLE GAUSSIAN PROCESS EMULATOR BY ACTIVE LEARNING WITH ERROR CONTROL

A. Parallel partial Gaussian process emulation for vectorized outputs

For demonstration purposes, our statistical emulator is built upon a parallel partial Gaussian Process (PP-GP) emulation approach[32] for representing the map between the particle density, a vectorized output on massive grid points, and the input function from the chemical and external potentials. Other emulators that produce uncertainty quantification for predictions can be used as well. The GP emulator is one of the most commonly used surrogate models for expensive physics-based calculations, and it has been widely used for emulating molecular simulation, the potential energy surface, and atomic force fields[1,7,33–35]. In combination with the dimension reduction technique, the GP emulator has also been used as a surrogate model for approximating the KS-DFT calculation of electron density[30].

Fig. 2 illustrates the Bayesian updating formulation in GP from prior to posterior with one-dimensional inputs and outputs. As shown in Fig. 2(a), the GP prior is a very flexible class of functions, that include smooth or wiggly functions with large or small fluctuation, using different choices of parameters. After assuming 12 Latin hypercube samples from a function $f(x) = \sin(2x=10) + 0{:}2\sin(2x=2{:}5)$, in Fig. 2(b), we plot the predictive mean from GP (blue curve) and the actual value of $f(x)$ (black curve), as well as 10 predic-tive posterior samples that fit the data points with opti-mized parameters and the 95% predictive interval (gray area), using a GP emulator implemented in[36], where the parameters are estimated by the posterior mode. First, we found a small number of samples can constrain the predictive mean and posterior samples to be around the truth. Second, the truth are almost all covered by the 95% predictive interval, indicating appropriate quantification of the predictive uncertainty. This internal uncertainty assessment by the GP emulator enables us to develop the criterion to control the predictive error for emulating functions with high-dimensional inputs in a probabilistic way, which will be illustrated in section III B.

We use the PP-GP emulator to model the particle density in cDFT. To highlight the salient feature of the PP-GP emulator, let $x = V^{ext}(:)$ denote the functional input discretized at p spatial grid points. For any input x, the particle density is defined at k grid points, written as $(x) = [_1(x); _2(x); :::; _k(x)]$. The number of spatial grid points used in discretizing external poten-tial and density can be different but, for simplicity, here we let $p = k$. Given any n cDFT calculations with in-puts $fx_1; :::; x_ng$, the density vector at any spatial grid point is assumed to follow a multivariate normal distribu-tion $_j = [_j(x_1); _j(x_2); :::; _j(x_n)]^T$ $MN(_j; ^2R)$; where $_j = [_j(x_1); :::; _j(x_n)]^T$ is the mean vector, ^{2} is a variance parameter, and R is an n n correlation matrix between density at any grid point over n input sce-narios, with $(i; j)$ entry parameterized by a kernel func-tion $K(x_i; x_j)$.

At any input x, the mean of the output is commonly modelled as $_j(x) = h(x)_j$, where $_j$ is a q-dimensional vector, and $h(x) = [h_1(x); :::; h_q(x)]$ is a row vector of mean basis function. Here we only use the intercept, i.e. $h(x) = 1$ and $q = 1$. Note that the mean parameter and variance parameter are distinct for each spatial grid point of the particle density. Having different mean and variance parameters makes the model more flexible to capture the variability of particle densities at different grid points.

We assume that the kernel function is isotropic, i.e., it is a function of the Euclidean distance between any pair of inputs. The correlation is commonly modeled by the

power exponential kernel functions or the Matérn kernel functions[37]. In this work, we use the Matérn kernel function with roughness parameter 5=2 to model the correlation between input x_a and x_b:

$$K(x_a; x_b) = \left(1 + \frac{\sqrt{5}d}{\rho} + \frac{5d^2}{3\rho^2}\right)\exp\left(-\frac{\sqrt{5}d}{\rho}\right) \quad (4)$$

where $d = ||x_a - x_b||$, with $||\cdot||$ denoting the Euclidean distance, and ρ is a range parameter that can be estimated by the training data. The sample path of the GP with the Matérn kernel function in (4) is twice mean differentiable[37], assuring good predictive performance in emulating smooth functions.

Assume we have run cDFT calculations at n different input scenarios $\{x_1; \ldots; x_n\}$, leading to a $k \times n$ particle density matrix $\Omega = [\omega_1; \ldots; \omega_n]$. We compute the predictive distribution after integrating out the mean and variance parameters for making predictions with the reference prior of the mean and variance parameters[38], $(\mu_j; \sigma^2) \propto 1/\sigma^2$, for spatial grid point $j = 1; \ldots; k$. Conditional on estimated range parameter $\hat{\rho}_j$ and output density at the jth coordinate ω_j, the predictive distribution of $\omega_j(x)$ at any new x and coordinate j, follows a non-centered scaled Student's t-distribution with $n - q$ degrees of freedom:

$$\omega_j(x)|\hat{\rho}_j; \omega_j \sim T(\hat{m}(x); \hat{\sigma}_j^2 K; n - q); \quad (5)$$

where

$$\hat{m}_j(x) = h(x)^T \hat{\beta}_j + r^T(x) R^{-1} (\omega_j - H\hat{\beta}_j); \quad (6)$$

$$K = K + h(x)^T (H^T R^{-1} H)^{-1} h(x); \quad (7)$$

with $h(x) = h(x) - H^T R^{-1} r(x)$, and

$$\hat{\beta}_j = (H^T R^{-1} H)^{-1} H^T R^{-1} \omega_j; \quad (8)$$

$$\hat{\sigma}_j^2 = (\omega_j - H\hat{\beta}_j)^T R^{-1} (\omega_j - H\hat{\beta}_j)/(n - q): \quad (9)$$

Here the $n \times q$ input mean basis matrix follows $H = [h^T(x_1); \ldots; h^T(x_n)]^T$, and $K = K(x; x) - r^T(x) R^{-1} r(x)$ denotes the correlation between test input and training input, with $r(x) = [K(x; x_1); \ldots; K(x; x_n)]^T$. The derivation of Eq. (5) can be found in the supplement materials. In PP-GP, the distribution of the output is assumed to be independent at two grid points, and thus we have $p(\omega(x) | \hat{\rho}; \cdot) = p(\omega_j(x) | \hat{\rho}_j; \cdot)$, for any j and x, which allows us to use Eq. (5) for computing the posterior predictive distribution. For further justification of the assumption, see Theorem 6.1 in[32]. The PP-GP emulator was implemented in the "RobustGaSP" package[36], where the marginal posterior mode estimator is used as a default method to estimate the range parameter [32].

In comparison with alternative methods, the PP-GP emulator discussed above has advantages in terms of computational scalability and internal uncertainty assessment. First, constructing the covariance matrix requires

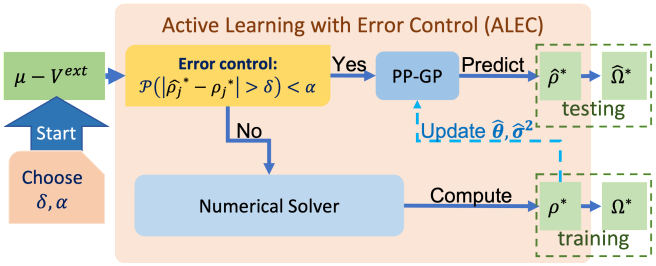

FIG. 3. An overview of the active learning with error control framework.

$O(n^2 p)$ operations, while computing the predictive mean in Eq. (6) only takes $O(nk) + O(n^3)$ operations, where n, p, and k denote the number of training simulation runs, the number of discretization points in the input functions and particle densities, respectively. The $O(n^3)$ operations come from the Cholesky decomposition for computing the inversion and determinant of the covariance matrix. In contrast, building a separate emulator independently of particle densities at each grid point takes $O(n^3 k)$ operations for computing the predictive mean, due to the inversion of the distinct covariance matrices at each spatial grid point. Approaches that model the spatial covariance matrix generally take $O(k^3)$ operations, which is even slower as the number of spatial grid points k of the density is large. Second, the uncertainty in predictions can be directly assessed as any quantiles and percentiles of the predictive distributions in (5) have closed-form expressions. The assessed uncertainty allows us to control the predictive error through integrating a numerical solver and an emulator into the algorithm, discussed below.

B. ALEC: active learning with error control approach

Active learning is a sequential design approach in statistics. In an "on-the-fly" online prediction scenario, there is no information regarding the future input to be predicted. If a configuration is new to the current training set, i.e., it is located in a sparsely sampled region, we may not be able to accurately predict this sample by the emulator. For a large functional input space, it is almost inevitable to extrapolate the sampled input space, as the number of simulated runs that populate this input space is sparse. Therefore, it is necessary to establish a criterion for determining whether an upcoming configuration can be reliably predicted.

In this work, we develop the active learning with error control (ALEC) emulator that can automatically control predictive error for simulations with high or infinite dimensional input space. A few criteria were studied before to detect the test case hard to be predicted, such as the D-optimality criterion[15] and the predictive variance criterion[17]. Yet, the threshold of these strategies often requires to be chosen empirically, as they may not be di-

rectly related to the predictive error. In our automatic approach, a new criterion has been established such that the threshold can be chosen automatically for any given predictive error bound and probability tolerance bound, based on the internal uncertainty assessment of the emulator. Fig. 3 shows an overview of our strategy. For a given testing input x, if the criterion that controls the predictive error is satisfied as shown in the yellow box, the particle density profile will be predicted; otherwise, a numerical solver will be called for computing the particle density corresponding to x, and this new information will be added to the training set and used to update the PP-GP emulator.

Let us first consider controlling the predictive error for particle density at a fixed coordinate j based on the PP-GP algorithm. Given a prespecified error bound $_j > 0$, we aim to have the predictive error of a test input x smaller than $_j > 0$ for more than 1 of the predictions, where is a small value (e.g. = 0:05). This means

$$P\ j_j(x) \quad \hat{}\ (x)j > {}_j\ ; \tag{10}$$

i.e., the probability that the absolute error between the predicted and true densities at the jth coordinate exceeds $_j$ is less than . Since small predictive variance from the emulator indicates high-fidelity in prediction, we predict the test input x if

$$\hat{}^2_j K \quad t_{=2}(n \quad q)\frac{j.}{} \tag{11}$$

where $t_{=2}(n \quad q)$ is the upper =2 quantile of a Student's t-distribution with n q degrees of freedom, K and $\hat{}^2_j$ are given in Eqs. (7) and (9), respectively. Otherwise, we will compute this particle density directly from cDFT using a numerical solver, as the assessed uncertainty for predicting this test input is large. This strategy satisfies the probabilistic error control outlined in Eq. (10), as shown in Appendix C.

We have a few remarks regarding this strategy. First, the error bound $_j$ and probability tolerance have interpretations. Suppose, for instance, one expects to see the predictive error in more than 95% of predictions smaller than 0:01, one can then let = 0:01 and = 0:05. Second, in the active learning strategy, one reduces the sample space from the original space X to $X_{F;j} = fx : \hat{}^2_j K \quad {}_j=t_{=2}(n \quad q)g$, where the sub-script F denotes the input set that can be reliably pre-dicted. A numerical solver for cDFT needs to be called for any input x that does not satisfy Eq. (11). Though selecting a smaller or leads to a smaller predictive er-ror, more evaluations from the numerical solver would be required, which increases the computational cost. Thus, a balance between the size of the cut-off value and com-putational time is needed in practice. Third, the de-gree of freedom of the Student's t-distribution increases with the sample size. When the sample size is sufficiently large, the t-distribution can be accurately approximated by a standard normal distribution. For instance, when n q = 300 and = 0:05, $t_{=2}(n \quad q)$ 1:968 and the up-per =2 quantile of a normal distribution $Z_{=2}$ 1:960, which only differs from that of a t-distribution by around 0:5%. Thus one can use the normal approximation if the sample size is large. Finally, the error control from Eq. (11) means that if the statistical emulator is a correct rep-resentation of the reality, and the number of test samples is infinitely large, we have at least 1 probability such that the absolute predictive error is less than . This does not preclude the cases where in slightly more than 1 of the predictions, the predictive error is larger than , due to model misspecification or the limited number of test samples. Fortunately, the predictive error at these inputs is typically close to , as the assessed uncertainty of these inputs is smaller than the threshold.

We can extend the strategy for controlling the error of the density at a fixed coordinate j to the overall error of density prediction. Consider that we use a uniform cutoff value , and that we make predictions for a test input x if each coordinate satisfies Eq. (11), i.e.,

$$\frac{}{\max_j f\hat{}^2_j K g} \quad t_{=2}(n \quad q)\frac{}{}: \tag{12}$$

We call the criterion in Eq. (12) the maximum variance selection criterion. Under this criterion, we are able to derive the probabilistic bound to control the predictive error of particle density at each coordinate,

$$P\ j_j(x) \quad \hat{}\ (x)j > \quad ; \qquad \text{for } j = 1;:::;k; \tag{13}$$

and the root of mean squared error (RMSE), defined as $\text{RMSE} = \sqrt{\sum_{j=1}^{k}(_j(x) \quad \hat{}\ (x_j))^2=k}$, follows

$$P\ \text{RMSE} > \ : \tag{14}$$

The detailed derivation of Eq. (13) and (14) is given in Appendix C. Eqs. (13) and (14) imply that the overall predictive error can be controlled in a probabilistic way under the maximum variance selection criterion in Eq. (12).

In practice, the cut-off in (12) is too conservative, and we rarely see more than samples larger than the claimed threshold. A less restrictive criterion to control the predictive error is by predicting any input set x such that

$$:= \quad \frac{\sqrt{\sum_{j=1}^{k} \hat{}^2_j K}}{k} \quad t_{=2}(n \quad q)\frac{}{}: \tag{15}$$

We call the criterion in (15) the average variance selection criterion. Eq. (15) controls the predictive error of vectorized outputs (particle densities) in the average sense, while it does not guarantee the error control as in Eq. (12). In practice, one may select a large number of points for numerical solvers to run using criterion (12), thereby requiring a larger computational cost, while the predictive error control can be achieved using criterion (15) with much less training samples.

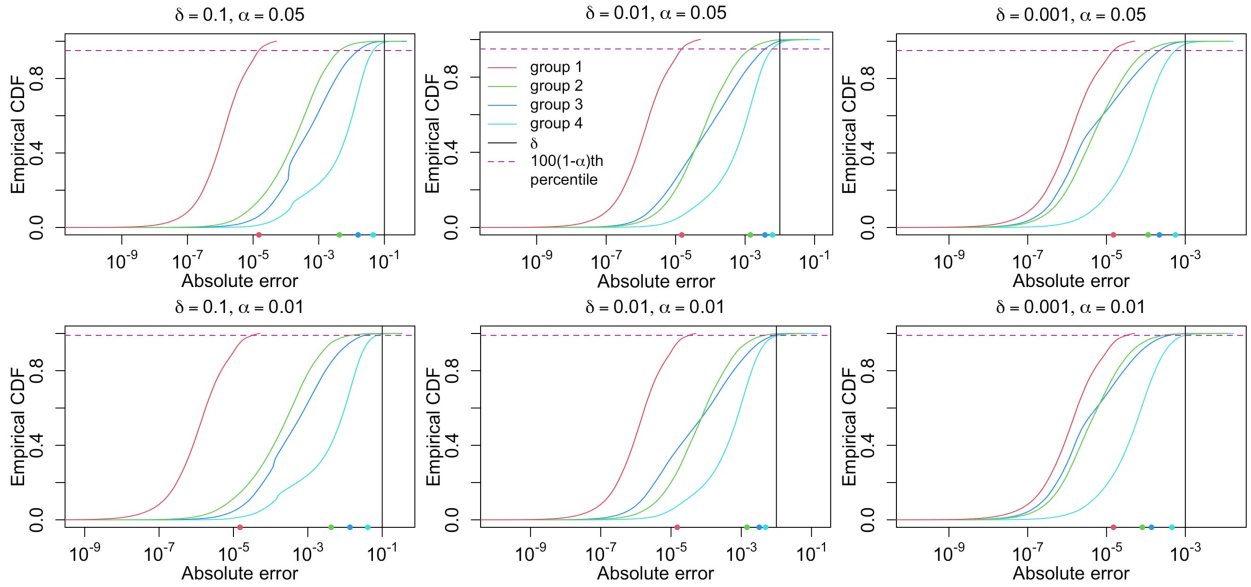

FIG. 4. Empirical CDF of absolute error of particle densities $j_j(x_i) - \hat{}(x_{ij})j$ for 4 classes of input functions at all grid points using the criterion (15) with different error bounds and probability tolerance thresholds. The horizontal line and the vertical line are 1 and , respectively. On the x-axis, the 1 percentiles of the absolute errors are recorded, all of which are less than .

To quantify whether the error is indeed controlled based on our threshold, we plot the empirical cumulative distribution function (CDF) of the absolute errors in Fig. 4 using the criterion (15) with different error bounds and probability tolerance thresholds for all 4 classes of input functions considered herein. In the upper middle panel, for instance, the vertical solid line marks the chosen threshold = 0:01 and the purple dashed line labels the percentile corresponding to 1 = 0:95. On the x-axis, we mark the values of the 95th percentile of the absolute errors for each group of input, all of which are less than 0.01. In the remaining panels of Fig. 4, the empirical CDFs of the absolute errors for various and values are illustrated. In general, the smaller error bound or probability tolerance threshold, the smaller the predictive error is, while more samples are required for achieving this accuracy level. Thus, a balance between the computational cost and accuracy level is generally required.

C. Sequential update of the ALEC emulator

When calculating the predictive mean and predictive correlation in Eqs. (6)-(7), one needs to solve $R^{-1}z$, for an n-dimensional real-valued vector $z \in R^n$. In the PP-GP emulator[32], we implement the Cholesky decomposition of the correlation matrix $R = LL^T$, and use the forward and backward substitution algorithms to compute $R^{-1}z$. In the ALEC emulator, whenever a sample $x \in R^p$ is added to the previous training set with n samples, the PP-GP model needs to be updated accordingly. Directly applying this approach will require $O(n^3)$ operations for computing the Cholesky decomposition each time when the sample size is n. To reduce the computational cost, we introduce a new approach that takes only $O(n^2)$ operations in performing the Cholesky decomposition.

To update L with the addition of x, we denote the correlation matrix of previous n design elements as $R_n = L_n L_n^T$, with L_n being the Cholesky decomposition of R_n. Next, we write the updated correlation matrix R_{n+1}:

$$R^{n+1} = \begin{matrix} R & r(x) \\ r^T(x) & K(x,x) \end{matrix}_n ;$$

where $r_n(x) = [K(x;x_1);:::;K(x;x_n)]^T$. In evaluating $R_{n+1} = L_{n+1}L^T_{n+1}$, it is clear that the first n rows of the lower triangular part of L_{n+1} is identical to the lower triangular part of L_n

$$L_{n+1} = \begin{matrix} L_n & O_n \\ v_n & l_n \end{matrix} :$$

Therefore, only the last row of L_{n+1} needs to be computed:

$$v_n(j) = \frac{1}{L_n(j;j)} \left[r_n(j) - \sum_{m=1}^{j-1} v_n(m)L_n(j;m) \right]; \quad (16)$$

$$l_n = \sqrt{K(x;x) - \sum_{m=1}^{n} v_n(m)^2}; \quad (17)$$

with $v_n(j)$ and $r_n(j)$ denoting the jth element of v_n and $r_n(x)$, respectively, and $L_n(i;j)$ denoting the (i;j) entry

of L_n. Eqs. (16) and (17) indicate that updating L_{n+1} requires $O(n^2)$ computational operations, for $i; j = 1; :::; n$. After obtaining the Cholesky decomposition L_{n+1} we can update other terms required in computing the predictive mean and variance.

D. Algorithm and the computational cost

We summarize the ALEC emulator in Algorithm 1. Given the initial PP-GP emulator, predictive error bound , and probability tolerance threshold , the ALEC emulator determines whether the particle densities can be reliably predicted for a new test input. The output contains the predictive particle densities for all test inputs and they are plugged into the energy functionals to compute the energy. Note that the range parameter in the kernel needs to be numerically estimated, which has the largest computational cost in the algorithm. When more training samples are added, the previously estimated range parameter ^ may no longer be accurate if our initial model has a small training sample size. To compromise the computational cost and accuracy in predictions, the range parameter in the ALEC emulator is re-estimated for every 50 training samples, if the training size is less than 350. When the training size is large, the estimated range parameter does not change much. In general, the frequency of re-estimating the kernel parameter depends on the computational budget.

Denote the initial training size as n_{ini} and the number of augmented samples as n_{aug}. The computational cost of constructing the Cholesky decomposition of the initial ALEC emulator is $O(n_{ini}^3)$, and the total computational order for updating the Cholesky decomposition is then $O(\sum_{n=n_{ini}+1}^{n_{ini}+n_{aug}} n^2)$. For computing the predictive mean and predictive variance, it takes $O(\sum_{n=n_{ini}+1}^{n_{ini}+n_{aug}} nk)$ and $O(\sum_{n=n_{ini}+1}^{n_{ini}+n_{aug}} n^2k)$, respectively.

The computational complexity of the online updating approach reduces the cost compared to directing computing the Cholesky decomposition at each iteration, which costs $O(\sum_{n=n_{ini}+1}^{n_{ini}+n_{aug}} n^3)$. The computational complexity is significantly faster than the time required to calculate the density using a numerical solver for computing the particle density at each test input. Indeed, the largest computational cost of our approach comes from the numerical solver for the training input set, as will be seen in Fig. 8. Since only a small fraction of samples are used as the initial training and augmented sample, the computational cost of the ALEC emulator is much smaller than running numerical solvers for each density.

Algorithm 1 ALEC emulator

Input: The number of test samples n_t, testing inputs X, Cholesky factor L of R in the initial PP-GP model, estimated variance parameters $\hat{\sigma}^2 = [\hat{\sigma}_1^2; :::; \hat{\sigma}_k^2]$ and mean parameters $\hat{\lambda} = [\hat{\lambda}_1; :::; \hat{\lambda}_k]$ in the initial PP-GP model, probability level and threshold chosen by the user.

$n_{aug} \leftarrow 0$;
for $i = 1$ to n_t do
  Set ith test sample in X as x and calculate $\hat{\sigma}_j^2 K$ for $j = 1; :::; k$ and via (7) and (15), respectively;
  if $< = t_{=2}(n \quad q)$ then
    Predict for the predictive distribution in (5) for this x, and record predictive mean (x) and scale $^2K; \hat{}$
  else
    $n_{aug} \leftarrow n_{aug} + 1$;
    Using the numerical solver to compute the corresponding output (x);
    if the training size is a multiple of 50 and the training size 350 then
      Rebuild the PP-GP emulator with retrained the range parameter in Matérn kernel;
    else
      Update PP-GP emulator with the added input x;
      Update parameters $\hat{}^2$ and $^2; \hat{}$
    end if
  end if
end for

Output: The number of augmented size n_{aug}, predicted mean, and variance of particle densities for remaining $n_t \quad n_{aug}$ samples.

IV. NUMERICAL RESULTS

Here we evaluate the predictive performance of our surrogate model based on particle densities generated from four classes of external potentials (closed-forms shown in Appendix B) with the length of hard rods $a = 1$ and the domain size $L = 9$. The values of chemical exter-nal input and parameters in each external potential are changed when generating the data. We assume the external potential functions and chemical potentials are used as inputs, while the parametric forms of the external potential functions are not known. We will test the predictive performance separately for each class of external potential functions, and jointly by combining all classes of external potential functions, in Section IV A and Section IV B, respectively. Section IV C gives the predictive performance of our approach for predicting a new class of densities.

We record the out-of-sample root mean squared error (RMSE) of particle density and the grand potential of

the test samples from

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{k}(\hat{\rho}_j(x_i) - \rho_j(x_i))^2}{nk}};$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{\quad} - \quad)^2}{n}};$$

where $\hat{\rho}_j(x_i)$ is the predicted density of ith hold out sample at jth coordinate, and $\hat{\quad}(x)$ and (x) are the pre-dicted and true grand potential for ith hold out sample, respectively. Other criteria, such as the proportion of the test samples covered in the 95% predictive intervals and the average length of the predictive intervals, are provided in supplementary materials. Note that we only compute the predictive error on those inputs we predict, not including those initial samples or the augmented sam-ples computed by the numerical solver.

We generate 2000 samples for each group with LHS, and compare our surrogate model with three other commonly used approaches. The first two approaches use conventional random sample (RS) designs, where we randomly draw n_{train} samples from the LHS samples as the training dataset and use the remaining samples as testing data. We have two distinct training sample sizes for RS samples. The sample size in the first RS sample (RS1) is specified as the initial training size in ALEC, while the sample size in the second RS sample (RS2) is specified as the final training size in ALEC. For predicting any test input, the training sample size in active learning is neither smaller than the training sample size in RS1 nor larger than the training sample size in RS1. We have also implemented the active learning algorithm with D-optimality[15], where the details are given in Appendix D.

A. Particle density and grand potential prediction for each group of external potential field separately

We first test different approaches for four families of functional inputs separately. The density profiles are initially trained using PP-GP with $n_{ini} = 20$ inputs per group. Then following our ALEC algorithm, the surrogate model is sequentially updated by augmenting the train data with samples selected by the average variance selection criterion in Eq. (15). After obtaining the predicted density ^ for the remaining test samples, we predict the grand potential by plugging ^ into the energy functional in Eq. (A2) in Appendix A.

Figure 5 displays the overall performance of different surrogate models (more detailed results can be found in the table in the supplementary materials). Here the threshold in the ALEC algorithm is set as = 0:01 and = 0:05, representing a strategy for having less than 5% of the coordinate to have an absolute error of density prediction larger than 0:01. The first group of external potentials has the least flexibility, as the only change in the input is . Thus, predicting the density profile for

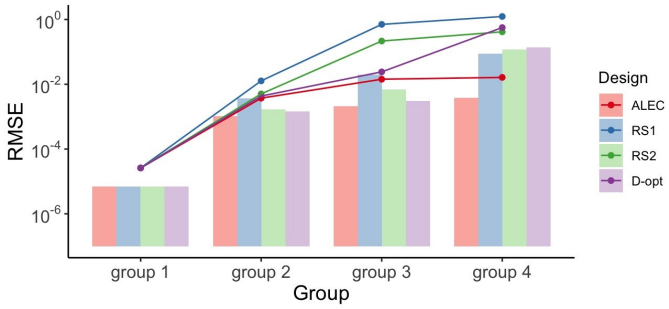

FIG. 5. The bar plots and line plots show the out-of-sample RMSE of density , and RMSE of grand potential , respec-tively. The ALEC emulator is compared with other meth-ods with three other designs: RS1, RS2, and D-opt (Active learning with D-optimality). For the ALEC emulator, we use = 0:01 and = 0:05 as the threshold, and the augmented sizes for groups 1-4 are 0, 7, 21, and 504, respectively. In D-opt, the augmented sizes for groups 1-4 are 0, 7, 21, and 582, respectively. The number of training sample sizes from RS1 and RS2 are the same as the initial sample size and terminal sample size in ALEC.

the first group of functions is the easiest. Models with only 20 inputs in the conventional RS design can predict particle densities relatively well, and no augmented sample is needed in the ALEC emulator. For the other three functional groups, the active learning algorithm has the smallest predictive RMSE on density and grand potential among all four designs. It is worth noting that the number of training inputs from active learning is always not larger than that in the RS2 design, but the predictive error by active learning is much smaller than the one by RS2. This is because the active learning strategy accurately identifies test samples hard to be predicted, and a numerical solver is called for computing these samples. Thus the computing budget is more efficiently spent with respect to the accuracy of predictions.

Figure 6 presents more detailed prediction results of the particle densities and grand potentials for the fourth class of external potentials. The boxplot in Fig. 6(a) gives the out-of-sample RMSE for predicting the particle densities. Many densities are inaccurately predicted based on the samples from RS1, RS2, or D-opt, resulting in large errors in predicting the grand potential (Fig. 6(b)). In comparison, the ALEC emulator identifies the samples that are difficult to be predicted and it uses them to enhance the emulator, thus providing accurate predictions for the remaining test samples. Although the total computational cost of the ALEC is similar to the methods with RS2 and D-opt schemes, the ALEC emulator results in better predictive accuracy.

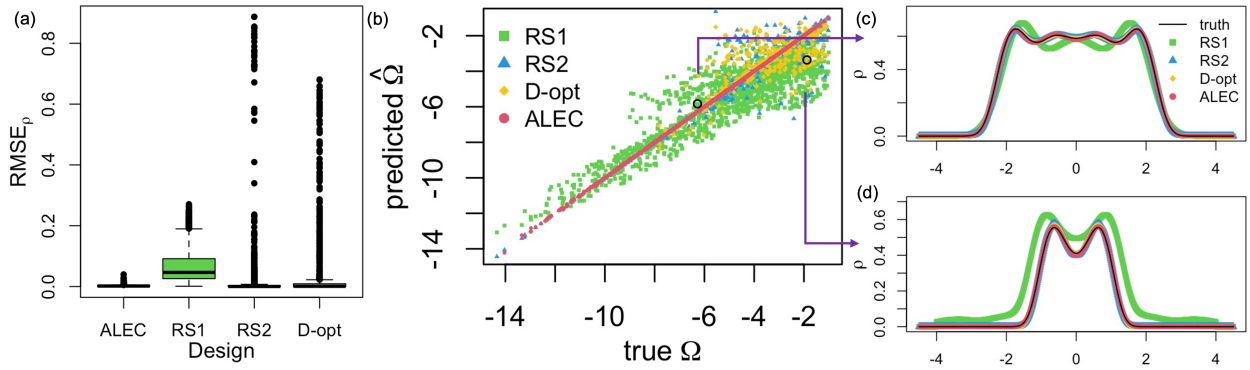

FIG. 6. A comparison of four surrogate models for predicting the group 4 functions: Active learning with error control (ALEC), random sampling with the number of training points as the initial number of training points in ALEC (RS1) and with the number of training points as the final number of training points in ALEC (RS2), and active learning with D-optimality (D-opt). (a) Boxplots of predictive RMSE of each predicted density in the testing dataset. (b) Predicted grand potential $\hat{\Omega}$ vs. the truth . (c)-(d) True and predicted densities for two selected density profiles.

B. Predicting particle densities and grand potentials for all groups of external potentials

In this subsection, we describe a more challenging scenario, where we aim to predict four groups of samples together. We assume that only the chemical potential and the external potential are used as input, while the parametric forms of the external potential are not known. The density profile and grand potential are harder to be predicted accurately in this case, as the input contains more degrees of freedom, leading to larger variability of the density and energy functionals, compared to the ones from any single class of input function alone. Since the inputs are a random sample from four classes of external potentials, we use the first $n_{ini} = 80$ samples as initial inputs to build the ALEC emulator. When a test sample comes, we use the criterion from Eq. (15) to determine whether the test sample can be accurately predicted by the ALEC emulator or a numerical solver will be needed for solving the system directly.

We show the emulator's overall performance in the left-most group (named as all groups) in Fig. 7 (detailed results are displayed in the supplementary materials). On average, the ALEC emulator performs much better, where the predictive RMSE of both particle densities and grand potentials is at least one magnitude smaller than all other approaches. A total of 873 training samples are added through the active learning algorithm, which is more than the total number of training samples added for building the emulator separately. This is because having a larger input space containing all 4 classes of external potentials increases the difficulty of learning densities, requiring more samples given the same threshold of the error. Note that the good predictive performance by ALEC is achieved using a similar or smaller sample size in training the emulator compared to the D-opt and RS2 approaches, by efficiently allocating the computing budget on computing particle densities hard to be predicted,

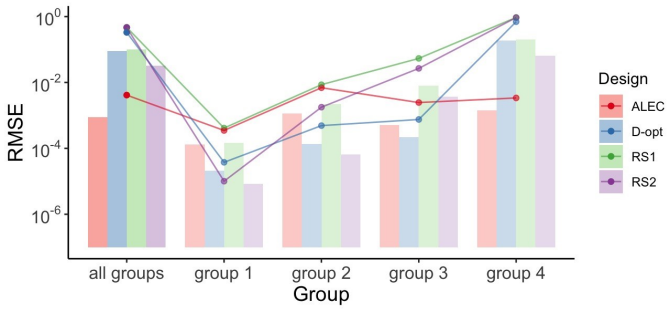

FIG. 7. Out-of-sample RMSE of particle density (bar plots) and grand potential (line plots) for active learning with er-ror control, RS1 design, RS2 design and active learning with D-optimality. The overall RMSE for combined all groups (solid bars and dots) and the RMSE of each individual group (shadow bars and dots) are both recorded. ALEC contains 80 inputs as initial training size, and the total number of aug-mented samples in the ALEC emulator (= 0:01, = 0:05) is 873, with 0, 1, 58, and 814 augmented samples in groups 1 to 4, respectively. In D-opt, the total number of augmented samples is 935, and the augmented sizes for groups 1-4 are 59, 73, 393, and 410, respectively. RS1 contains a total of the first 80 training samples after uniformly mixing all four groups of density together. RS2 contains the first 953 training inputs, with 245, 221, 238, and 249 for groups 1-4, respectively. Since the ALEC emulator identifies the hardest region (group 4) to be predicted, thereby augmenting most samples from group 4, it has the smallest predictive error overall and predictive errors of all groups are controlled.

and accurately predicting the rest of the densities with a negligible computational cost.

When examining the performance of different approaches in each function group, we found that the RMSE of particle densities from ALEC is homogenous across groups, while RS2 and D-opt approaches have smaller predictive errors for the first two groups, but have much worse performance for the fourth group. This

is because RS2 and D-opt select more points from the first two groups to train the emulator, thereby yielding a similar or better result than ALEC in these two groups. However, the error from the first two groups can be controlled below the threshold of 0:01 for more than 95% of the test samples with no augmented samples or only a small number of augmented samples. In the ALEC emulator, for instance, the augmented samples from groups 1 and 2 are 0 and 1, respectively. On the contrary, the particle densities and grand potential energy from group 4 are the hardest to be predicted. This is automatically identified by ALEC, and the most computing budget (814 augmented samples) is spent on the fourth group to ensure the predictive errors of most samples are controlled below the threshold. This is exactly what we intend to accomplish. The ALEC can identify input regions that are difficult to predict, and then put more computing resources on those regions to control the predictive error. Consequently, the predictive error is controlled below the threshold for all subclasses, even if we do not know the labels of subclasses from the test samples, as all input classes are combined for predictions.

Another interesting finding is that the additional training samples from other input functional classes do not substantially improve the predictive performance for a specific input class. This is because the inputs from one class are distant from another class, as shown in Fig. 1(b), and thus the correlation between different input classes is small. It may be of interest to automatically identify the cluster in the training and split the training sample when constructing the emulator. This is useful as splitting the samples can reduce the computational complexity of the emulator. To explore this idea, we utilized a simple input decomposition technique. When the size of a cluster reaches 400, we divide it into two sub-clusters using the K-nearest neighbors (KNN) algorithm[39]. In this approach, the number of augmented samples and predictive RMSE of particle densities are 779 and 9:73 10 ^{4}, respectively. It achieves similar accuracy with 94 fewer training samples than the ALEC algorithm without input decomposition. Developing a more comprehensive way for input decomposition approach can further enhance the efficiency and accuracy of the surrogate model.

Fig. 8 compares the computational time required in the ALEC approach and in calling a numerical solver (NS) each time. The largest computational cost in the ALEC algorithm comes from calling the NS for computing the particle densities of the initial and augmented training samples, while the time in constructing the ALEC emulator and making predictions is negligible, as the training sample size is not large. Since the ALEC emulator only requires a fraction of the samples to be numerically solved, it is much faster than running a numerical solver for each test sample. Note that RS2 and D-opt compared herein have a similar computational cost as the ALEC emulator, as the number of density profiles to be solved by NS is similar, yet the predictive error was not controlled in those approaches.

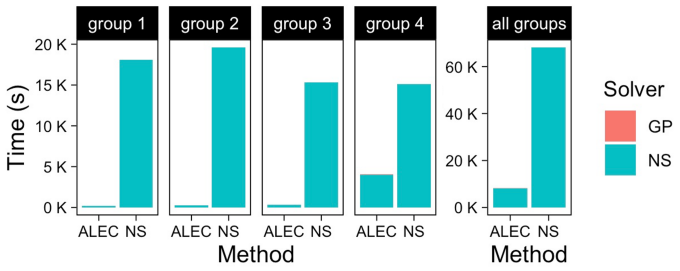

FIG. 8. The running time for each function group using the ALEC algorithm and using numerical solver (NS). As the running time for building the GP model in ALEC is very small, the bars for GP are nearly invisible. Other approaches, such as RS2 and D-opt, have similar computational costs as ALEC as the training and augmented sample sizes are similar.

C. Predicting a new class of external potential

In this section, we examine the performance of the ALEC approach to predict inputs from a new functional form that is not in the training data set. To test this case, in addition to the existing 8000 samples, we generated 2000 new samples based on the weighted inputs between groups 2 and 3 as explained in Appendix B. We test the performance of the ALEC emulator for predicting the new input class, starting with the final GP emulator developed for groups 1-4 and all groups combined.

The left panel of Fig. 9 displays empirical CDF plots of the absolute errors of the ALEC emulator using the average variance selection criterion in Eq. (15). First, the predictive error of most of the test samples in the new group is small, for any initial emulator built upon other groups. Less than = 5% of the absolute errors are greater than the chosen error bound = 0:01 for models using groups 1-3 and all groups combined. When we start from the initial emulator built from group 4 in-put, 11:2% of predictive errors are greater than = 0:01, which is larger than the probability tolerance threshold = 5%. A close examination reveals that the percent-age of the absolute error greater than is typically large for coordinates at the tail of the particle densities (Fig. 9 middle panel), because of a rapid change in density at both ends, making it distinct from existing densities in group 4. The difference in the particle densities in the new class and group 4 increases the model discrep-ancy and makes the uncertainty in predictions harder to be quantified. Among the 11% of the test samples with RMSE larger than the threshold, however, most RMSEs are still very close to the threshold value, making this criterion ideal if the computational budget is not large.

On the other hand, the right panel of Fig. 9 shows the CDF of the RMSEs based on the maximum variance selection criterion in Eq. (12). Much less than probability of the predictive error exceeds the threshold for

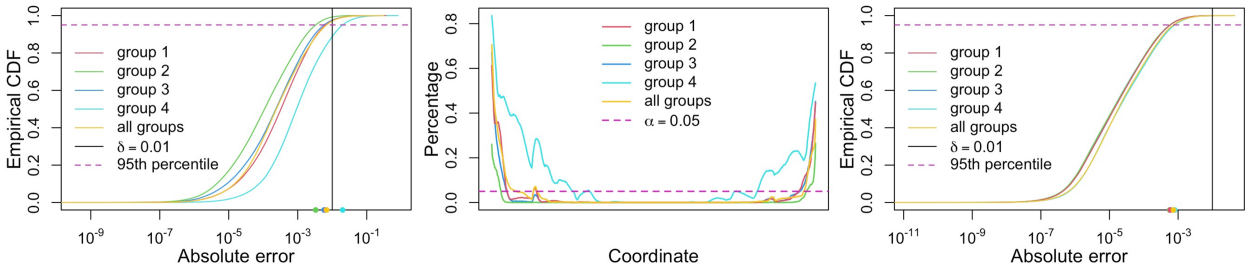

FIG. 9. Prediction results from the ALEC emulator on a new class of external potentials where ALEC models are trained by the previous four functional groups separately and by all functional groups together. = 0:01 and = 0:05 are used. Left panel: the empirical CDF of absolute error using the average variance selection criterion in Eq. (15). The number of augmented samples using the model for groups 1-4 and all groups combined are 92, 122, 67, 48, and 15, respectively. Middle panel: the percentage that the absolute difference is greater than for each coordinate, i.e. $\sum_{i=1}^{n} 1\{f_j(x)_i - \hat{}(x_j) > g\}/n; j = 1; :::; k$. Right panel: the empirical CDF of absolute error using the maximum variance selection criterion in Eq. (12). The number of augmented samples using the model for groups 1-4 and all groups combined are 408, 395, 384, 362, and 263, respectively.

all the cases shown here, as the maximum variance selection criterion is conservative, yielding more augmented samples and higher predictive accuracy. When the model discrepancy is large, one may use the maximum variance selection criterion to achieve a higher level of accuracy in predictions.

V. CONCLUSION

In this work, we propose the ALEC emulator, an active learning framework for controlling the predictive error of approximating computationally expensive functionals, based on the internal uncertainty assessment of the statistical surrogate model. This result has a probabilistic interpretation and it is fully automatic. We test the ALEC emulator on cDFT calculations of one-dimensional hard rods with different chemical potentials and external potentials. Through the numerical simulations of the density profiles and grand potentials for different groups of functions, we show that ALEC outperforms the conventional random sample designs, as well as the active learning approach with the D-optimality criterion in terms of prediction accuracy. With a fraction of the computational cost of running a numerical solver for each test input, we are able to predict functionals accurately with a controlled predictive error using the ALEC emulator. The ALEC emulator is designed as a reliable building block to be integrated for solving challenging simulation tasks, such as predicting expensive cDFT calculations on functional input, since the predictive error of the ALEC emulator is controlled for the test samples.

There are several future research directions. First, the ALEC emulator was built upon the PP-GP surrogate model for predicting vectorized output, while the approach can be generalized to using other surrogate models with uncertainty assessment in predictions. Physical symmetries, for instance, can be incorporated to define a new descriptor and distance metric to constrain the surrogate model. For instance, a common way to ensure translational invariance is to use pairwise distances of input, instead of the input function itself. Existing functionals found in physics[40] may be integrated as the mean of the emulator to improve its accuracy. Furthermore, dimension reduction of the input space through orthogonal representation can be helpful for reducing computational costs and improving the accuracy of predicting systems on 3D coordinates. Potential approaches include basis representation for reducing the dimensionality of the output space[30] and orthogonal projection of the inputs by maintaining large gradients of outcomes for reducing the dimensionality of the input space[41]. Finally, one may jointly model the grand potential energy and densities for predicting both quantities. This is particularly useful for applications with complex molecular systems, where computing the potential energies from particle densities is expensive.

SUPPLEMENTARY MATERIALS

The supplementary materials contain the derivation of predictive t-distribution in Eq. (5), and additional prediction results of particle density and grand potential.

ACKNOWLEDGEMENTS

This study is partially supported by the National Science (NSF) Foundation under Award No. DMS-2053423. X.F. acknowledged partial support from the NSF BioPACIFIC Materials Innovation Platform (MIP) under Award No. DMR-1933487 and the UCSB academic senate faculty research grants program. J.W. acknowledged partial support by the NSF Harnessing the Data Revolution (HDR) Big Ideas Program under Grant No. NSF 1940118.

X.F. and M.G. contributed equally to this work.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Appendix A: Analytical forms in cDFT

The closed-form expressions of F_{ex}, $\frac{\delta F_{ex}[\rho]}{\delta \rho(x)}$, and F^{ex} for 1D HR[26,42] are summarized here:

$$\frac{\delta F_{ex}[\rho]}{\delta \rho(x)} = \int_x^{x+a} \frac{R_z(z)}{1 - \int_{z-a}^{x} \rho(y)dy} dz \quad (A1)$$

$$R_x[\rho] = \int_{x-a}^{x} \ln\left[1 - \int_{x-a}^{x} \rho(y)dy\right] dx + \int_{x}^{x+a} \frac{(x+a) - z}{1 - \int_{z-a}^{x} \rho(y)dy} dz; \quad (A2)$$

$$F_{ex}[\rho] = \int \rho(x) R_x[\rho] dx; \quad (A3)$$

where a is the length of each hard rod.

Appendix B: External potentials and chemical potentials

Let L be the distance between two hard walls, and a is the length of the hard rods (HR). We consider the following four parametric forms of the external potential in this paper.

The first functional form applies to uniform hard rods confined between hard walls[43]:

$$V^{ext}(s) = \begin{cases} 1; & s < a/2; s > L - a/2; \\ 0; & a/2 < s < L - a/2: \end{cases} \quad (B1)$$

In the second functional form, the hard rods experience a van der Waals-like attraction from the hard walls

$$V^{ext}(s) = \begin{cases} 1; & s < a/2; s > L - a/2; \\ \epsilon\left[\left(\frac{a}{s+a/2}\right)^3 + \left(\frac{a}{L+a/2-s}\right)^3\right]; & a/2 < s < L - a/2: \end{cases} \quad (B2)$$

Eq.(B2) is adopted from[43]. With fixed a and L, ϵ is a parameter in the range of $\epsilon \in (0.1; 2.2)$. Note that when $\epsilon = 0$, we have the same potential as the first one.

The third functional form consists of hard-wall potentials and a repulsion linearly proportional to the distance

$$V^{ext}(s) = \begin{cases} 1; & s < a/2; s > L - a/2; \\ m_g s; & a/2 < s < L - a/2: \end{cases} \quad (B3)$$

Intuitively, the linear form mimics the gravitational potential.[44] This is the only asymmetric potential considered in this work. In training the machine-learning models, the range of m_g is chosen to be $(0.1; 3)$.

The power-law external potential[44] is the fourth functional class considereed in this work. This external potential has more flexibility as it has three parameters, $u_0; x_0$, and a_0 that can be used to modify the range of interactions:

$$V^{ext}(s) = \begin{cases} 1; & s < a/2; s > L - a/2; \\ u_0^{\frac{s - L/2 - a_0}{x_0}}; & a/2 < s < L - a/2; \end{cases} \quad (B4)$$

where $a_0 > 0$. In this work, we use $u_0 \in (1; 3); x_0 \in (1; 3)$ and $a_0 \in (2; 5)$. Fig. 10 shows five examples of generated densities for each external potential.

In section IV B, the performance of our strategy is evaluated on a new set of densities. The external potential of these densities is generated from the weighted average of (B2) and (B3):

$$V^{ext}(s) = \begin{cases} 1; & s < a/2; s > L - a/2; \\ w\left[\left(\frac{a}{s+a/2}\right)^3 + \left(\frac{a}{L+a/2-s}\right)^3\right] + (1-w) m_g s; & a/2 < s < L - a/2; \end{cases} \quad (B5)$$

where $w \in (0; 1)$.

Appendix C: Derivation of probabilistic upper bounds

First, we derive the probabilistic upper bound for the absolute difference of densities. For a fixed coordinate j, based on the predictive distribution in (5), we have $P\left(|\rho_j(x) - \hat{\rho}_j(x)| > \sqrt{\hat{\sigma}_j^2 K} t_{\alpha/2}(n-q)\right) = \alpha$. Since $\rho_j(x)$ is predicted for sample x only if $\sqrt{\hat{\sigma}_j^2 K} / t_{\alpha/2}(n-q)$, the probability of predictive error larger than error bound ϵ_j follows

$$P\left(|\rho_j(x) - \hat{\rho}_j(x)| > \epsilon_j\right) = P\left(|\rho_j(x) - \hat{\rho}_j(x)| > \epsilon_j \frac{t_{\alpha/2}(n-q)}{\sqrt{\hat{\sigma}_j^2 K} t_{\alpha/2}(n-q)}\right) = P\left(|\rho_j(x) - \hat{\rho}_j(x)| > \frac{\epsilon_j}{\sqrt{\hat{\sigma}_j^2 K}} t_{\alpha/2}(n-q)\right) = \alpha.$$

Next, let us consider controlling the grid-level predictive error using the maximum variance selection criterion in (12), where $\hat{\rho}(x)$ is predicted for x only if $\sqrt{\max_j \{\hat{\sigma}_j^2 K\}} / t_{\alpha/2}(n-q)$. Then for any $j, j = 1; \ldots; k$,

$$P\left(|\rho_j(x) - \hat{\rho}_j(x)| > \epsilon\right) \le P\left(|\rho_j(x) - \hat{\rho}_j(x)| > \frac{\epsilon}{\sqrt{\max_j \hat{\sigma}_j^2 K} t_{\alpha/2}(n-q)}\right) \le P\left(|\rho_j(x) - \hat{\rho}_j(x)| > \frac{\epsilon}{\sqrt{\hat{\sigma}_j^2 K} t_{\alpha/2}(n-q)}\right) = \alpha;$$

from which equation (13) is proved.

Third, let us derive the probabilistic bound for root mean squared error (RMSE) under the maximum variance selection criterion in Eq. (12), where RMSE =

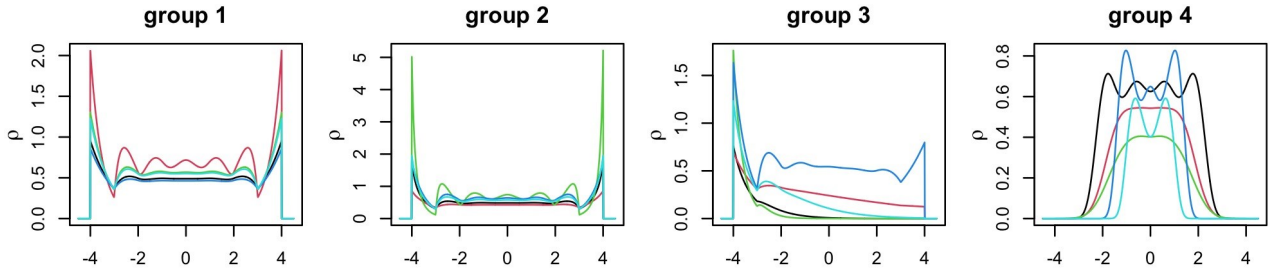

FIG. 10. Each plot gives 5 examples of generated densities from external potential in B1-B4.

$q \sum_{j=1}^{k} (\phi_j(x) - \hat{\phi}(x))^2/k$. For $j = 1, \ldots, k$, we have

$$P(\mathrm{RMSE} > \epsilon) = P\left(\sum_{j=1}^{k} (\phi_j(x) - \hat{\phi}_j(x))^2 > k\epsilon^2\right)$$

$$P\left(\max_j (\phi_j(x) - \hat{\phi}(x))^2 > \epsilon^2\right)$$

$$P\left(\max_j (\phi_j(x) - \hat{\phi}(x))^2 > \max_j \hat{\sigma}^2 K t^2_{\alpha/2}(n_j - q)\right);$$

where the third inequality follows from the condition in Eq. (12). Supposing the left hand side inside the probability is maximized when $j = j$, then we have

$$P(\mathrm{RMSE} > \epsilon) \quad P\left((\phi_j(x) - \hat{\phi}(x))^2 > \max_j \hat{\sigma}^2 K t^2_{\alpha/2}(n_j - q)\right);$$

$$P\left((\phi_j(x) - \hat{\phi}(x))^2_j > \hat{\sigma}^2 K t_{\alpha/2}(n_j - q)\right)^2 = ;$$

from which equation (14) is proved.

Lastly, we give the upper bound for the expectation of mean squared error, $\mathrm{MSE} = \sum_{j=1}^{k} (\phi_j(x) - \hat{\phi}_j(x))^2/k$, under the average variance selection criterion (Eq. (15)),

$$E_{(x)}(\mathrm{MSE}) = E_{(x)}\left[\sum_{j=1}^{k} \frac{(\phi_j(x) - \hat{\phi}(x))^2}{k}\right]$$

$$= \sum_{j=1}^{k} \frac{\hat{\sigma}_j^2 K}{k} \frac{n - q}{n - q - 2} \frac{2}{t^2_{\alpha/2}(n - q)} \frac{n - q}{n - q - 2}:$$

For moderately large n, and small , we have $E_{(x)}(\mathrm{MSE}) \quad ^2$. For instance, for any $n \quad 20$ and constant mean basis ($q = 1$), $E_{(x)}(\mathrm{MSE}) \quad ^2$ for any $0{:}2$.

Appendix D: Details of the D-optimality criterion

Here we discuss the details of the D-optimality criterion. Consider the correlation function $K(x; x_i); i = 1; \ldots; n$; with each training set sample serving as a set of basis functions. Then, by definition, these basis functions of the training dataset form the correlation matrix R, and the basis functions with any testing sample x gives $r^T(x)$. Define the row vector $c = r^T(x)R^{-1}$ for each sample x. Set the threshold $c_{th} \quad 1$ for the maximum absolute value in c. If the uncertainty estimate $c_{max} := \max|c_i| > c_{th}$, a numerical solver will be called to solve the system and the outcomes at x will be added to the training set. Otherwise, the emulator will be used to predict the test input x. Since the D-optimality criterion is not directly related to the predictive error, selecting threshold c_{th} may be performed in a case-by-case manner. In order to make a comparison with our method, the threshold is adjusted to make the number of augmented samples selected from D-optimality at least as large as the one in our strategy. For more discussion about the D-optimality criterion in emulating molecular simulations, we refer the readers to[15] and section 15.2.3 in[14].

REFERENCES

[1] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, "Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons," Phys. Rev. Lett. 104, 136403 (2010).

[2] S. Chmiela, H. E. Sauceda, K.-R. Müller, and A. Tkatchenko, "Towards exact molecular dynamics simulations with machine-learned force fields," Nat. Comm. 9, 1–10 (2018).

[3] A. V. Shapeev, "Moment tensor potentials: A class of systematically improvable interatomic potentials," Multiscale Modeling & Simulation 14, 1153–1173 (2016).

[4] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, "Schnet–a deep learning architecture for molecules and materials," The Journal of Chemical Physics 148, 241722 (2018).

[5] S.-C. Lin, G. Martius, and M. Oettel, "Analytical classical density functionals from an equation learning network," The Journal of Chemical Physics 152, 021102 (2020).

[6] L. Li, S. Hoyer, R. Pederson, R. Sun, E. D. Cubuk, P. Riley, K. Burke, et al., "Kohn-Sham equations as regularizer: Building prior knowledge into machine-learned physics," Physical review letters 126, 036401 (2021).

[7] V. L. Deringer, A. P. Bartók, N. Bernstein, D. M. Wilkins, M. Ceriotti, and G. Csányi, "Gaussian process regression for materials and molecules," Chemical Reviews 121, 10073–10141 (2021).

[8] P. Yatsyshin, S. Kalliadasis, and A. B. Duncan, "Physics-constrained bayesian inference of state functions in classical

density-functional theory," The Journal of Chemical Physics 156, 074105 (2022).

[9] J. Westermayr, M. Gastegger, K. T. Schütt, and R. J. Maurer, "Perspective on integrating machine learning into computational chemistry and materials science," The Journal of Chemical Physics 154, 230903 (2021).

[10] O. T. Unke, S. Chmiela, H. E. Sauceda, M. Gastegger, I. Poltavsky, K. T. Schütt, A. Tkatchenko, and K.-R. Müller, "Machine learning force fields," Chemical Reviews 121, 10142–10186 (2021).

[11] M. Sajjan, J. Li, R. Selvarajan, S. H. Sureshbabu, S. S. Kale, R. Gupta, V. Singh, and S. Kais, "Quantum machine learning for chemistry and physics," Chemical Society Reviews (2022).

[12] T. J. Santner, B. J. Williams, and W. I. Notz, The design and analysis of computer experiments (Springer Science & Business Media, 2003).

[13] A. W. van der Vaart and J. H. van Zanten, "Adaptive Bayesian estimation using a Gaussian random field with inverse gamma bandwidth," The Annals of Statistics , 2655–2675 (2009).

[14] K. T. Schütt, S. Chmiela, O. A. von Lilienfeld, A. Tkatchenko, K. Tsuda, and K.-R. Müller, Machine learning meets quantum physics (Springer Cham, 2020).

[15] E. V. Podryabinkin and A. V. Shapeev, "Active learning of linearly parametrized interatomic potentials," Computational Materials Science 140, 171–180 (2017).

[16] E. Uteva, R. S. Graham, R. D. Wilkinson, and R. J. Wheatley, "Active learning in Gaussian process interpolation of potential energy surfaces," The Journal of chemical physics 149, 174114 (2018).

[17] J. Vandermause, S. B. Torrisi, S. Batzner, Y. Xie, L. Sun, A. M. Kolpak, and B. Kozinsky, "On-the-fly active learning of interpretable Bayesian force fields for atomistic rare events," npj Computational Materials 6, 1–11 (2020).

[18] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," Proceedings of the IEEE 104, 148–175 (2015).

[19] B. J. Shields, J. Stevens, J. Li, M. Parasram, F. Damani, J. I. M. Alvarado, J. M. Janey, R. P. Adams, and A. G. Doyle, "Bayesian reaction optimization as a tool for chemical synthesis," Nature 590, 89–96 (2021).

[20] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63, 425–464 (2001).

[21] M. Gu and L. Wang, "Scaled Gaussian stochastic process for computer model calibration and prediction," SIAM/ASA Journal on Uncertainty Quantification 6, 1555–1583 (2018).

[22] A. Sauer, R. B. Gramacy, and D. Higdon, "Active learning for deep Gaussian process surrogates," Technometrics , 1–15 (2022).

[23] X. Yue, Y. Wen, J. H. Hunt, and J. Shi, "Active learning for Gaussian process considering uncertainties with application to shape control of composite fuselage," IEEE Transactions on Automation Science and Engineering 18, 36–46 (2020).

[24] R. Evans, "Density functionals in the theory of nonuniform fluids," in Fundamentals of Inhomogeneous Fluids, edited by D. Henderson (Marcel Dekker, New York, 1992) pp. 85–175.

[25] J. Wu and Z. Li, "Density-functional theory for complex fluids," Annu. Rev. Phys. Chem. 58, 85–112 (2007).

[26] J. Percus, "Equilibrium state of a classical fluid of hard rods in an external field," Journal of Statistical Physics 15, 505–511 (1976).

[27] L. Shang-Chun and M. Oettel, "A classical density functional from machine learning and a convolutional neural network," SciPost Physics 6, 025 (2019).

[28] P. Cats, S. Kuipers, S. De Wind, R. Van Damme, G. M. Coli, M. Dijkstra, and R. Van Roij, "Machine-learning free-energy functionals using density profiles from simulations," APL Materials 9, 031109 (2021).

[29] J. C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, and K. Burke, "Finding density functionals with machine learning," Physical review letters 108, 253002 (2012).

[30] F. Brockherde, L. Vogt, L. Li, M. E. Tuckerman, K. Burke, and K.-R. Müller, "Bypassing the Kohn-Sham equations with machine learning," Nat. Comm. 8, 1–10 (2017).

[31] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," Journal of machine learning research 9 (2008).

[32] M. Gu and J. O. Berger, "Parallel Partial Gaussian Process Emulation for Computer Models with Massive Output," Annals of Applied Statistics 10, 1317–1347 (2016).

[33] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld, "Fast and accurate modeling of molecular atomization energies with machine learning," Physical review letters 108, 058301 (2012).

[34] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, "Machine learning of accurate energy-conserving molecular force fields," Sci. Adv. 3, e1603015 (2017).

[35] H. Li, M. Zhou, J. Sebastian, J. Wu, and M. Gu, "Efficient force field and energy emulation through partition of permutationally equivalent atoms," The Journal of Chemical Physics 156, 184304 (2022).

[36] M. Gu, J. Palomo, and J. O. Berger, "RobustGaSP: Robust Gaussian Stochastic Process Emulation in R," The R Journal 11, 112–136 (2019).

[37] C. E. Rasmussen, Gaussian processes for machine learning (MIT Press, 2006).

[38] J. O. Berger, V. De Oliveira, and B. Sansó, "Objective Bayesian analysis of spatially correlated data," Journal of the American Statistical Association 96, 1361–1374 (2001).

[39] E. Fix and J. L. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," International Statistical Review/Revue Internationale de Statistique 57, 238–247 (1989).

[40] H. Ma, A. Narayanaswamy, P. Riley, and L. Li, "Evolving symbolic density functionals," Science Advances 8, eabq0279 (2022), https://www.science.org/doi/pdf/10.1126/sciadv.abq0279.

[41] P. G. Constantine, E. Dow, and Q. Wang, "Active subspace methods in theory and practice: applications to kriging surfaces," SIAM Journal on Scientific Computing 36, A1500–A1524 (2014).

[42] P. Tarazona, J. A. Cuesta, and Y. Martínez-Ratón, "Density functional theories of hard particle systems," in Theory and simulation of hard-sphere fluids and related systems (Springer, 2008) pp. 247–341.

[43] T. Vanderlick, H. Davis, and J. Percus, "The statistical mechanics of inhomogeneous hard rod mixtures," The Journal of chemical physics 91, 7136–7145 (1989).

[44] B. Bakhti, M. Karbach, P. Maass, and G. Müller, "Monodisperse hard rods in external potentials," Physical Review E 92, 042112 (2015).

Supplementary Materials

1. DERIVATION OF THE PREDICTIVE T-DISTRIBUTION

Here we present the derivation of the predictive t-distribution of $r_j(\mathbf{x})$ at jth grid point with a new input $\mathbf{x}$ in Eq. (5). Conditional on (r_j, q_j, s^2_j, g), we have

$$r_j(\mathbf{x}) | r_j, q_j, s^2_j, g \sim MN(m_j(\mathbf{x}), s^2 K(\mathbf{x},\mathbf{x})), \tag{S1}$$

where $\hat{m}_j(\mathbf{x}) = \mathbf{h}(\mathbf{x})q_j + \mathbf{r}(\mathbf{x})^T\mathbf{R}^{-1}(r_j - \mathbf{H}q_j)$ and $K(\mathbf{x},\mathbf{x}) = K(\mathbf{x},\mathbf{x}) - \mathbf{r}(\mathbf{x})^T\mathbf{R}^{-1}\mathbf{r}(\mathbf{x})$. Denote the precision parameter $f_j = 1/s^2_j$ and the prior becomes $p(q_j, f_j) \propto 1/f_j$. The posterior distribution of mean and variance parameters (q_j, f_j) follow

$$\begin{aligned}
p(q_j, f_j | r, \hat{g}) &\propto p(q_j, f_j)\, p(r_j, \hat{g} | q_j, f_j) \\
&\propto \frac{1}{f_j} f_j^{n/2} \exp\left[-\frac{f_j}{2}(r_j - \mathbf{H}q_j)^T\mathbf{R}^{-1}(r_j - \mathbf{H}q_j)\right] \\
&\propto f_j^{n/2-1} \exp\left[-\frac{f_j}{2}(r_j^T\mathbf{R}^{-1}r_j - 2r_j^T\mathbf{R}^{-1}\mathbf{H}q_j + q_j^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}q_j)\right] \\
&\propto f_j^{q/2} \exp\left[-\frac{f_j}{2}(q_j - \hat{q}_j)^T(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})(q_j - \hat{q}_j)\right] \\
&\quad \cdot f_j^{(n-q)/2-1} \exp\left[-\frac{f_j}{2}(r_j - \mathbf{H}\hat{q}_j)^T\mathbf{R}^{-1}(r_j - \mathbf{H}\hat{q}_j)\right],
\end{aligned}$$

where $p(q_j | r_j, f_j, \hat{g})$ and $p(f_j | r_j, \hat{g})$.

where $\hat{q}_j = (\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{R}^{-1}r_j$ and the last step follows from adding and subtracting $\hat{q}_j^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\hat{q}_j$ in the exponent. Thus, $(q_j \mid r_j, f_j, g)$ and $(f_j \mid r_j, g)$ are distributed as a multivariate normal distribution and a gamma distribution, respectively,

$$q_j \mid r_j, f_j, \hat{g} \sim MN\left(\hat{q}_j, (f_j\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\right), \tag{S2}$$

$$f_j \mid r_j, \hat{g} \sim \text{Gamma}\left(\frac{n-q}{2}, \frac{(r_j - \mathbf{H}\hat{q}_j)^T\mathbf{R}^{-1}(r_j - \mathbf{H}\hat{q}_j)}{2}\right). \tag{S3}$$

Third, we marginalize out q_j in Eq. (S1). Since both Eqs. (S1) and (S2) are normal distributions, we have

$$r_j(\mathbf{x}) \mid r_j, s^2_j, \hat{g} \sim N(\hat{r}_j(\mathbf{x}), f_j^{-1}K_j(\mathbf{x},\mathbf{x})), \tag{S4}$$

where the mean and variance can be obtained by the law of total expectation and variance:

$$\begin{aligned}
&\mathbf{E}[r_j(\mathbf{x}) | r_j, s^2_j, \hat{g}] \\
&= \mathbf{E}[\mathbf{E}[r_j(\mathbf{x}) | r_j, q_j, s^2_j, \hat{g}]] \\
&= \mathbf{E}[\mathbf{h}(\mathbf{x})q_j + \mathbf{r}(\mathbf{x})^T\mathbf{R}^{-1}(r_j - \mathbf{H}q_j) | r_j, s^2_j, \hat{g}] \\
&= \mathbf{h}(\mathbf{x})\hat{q}_j + \mathbf{r}(\mathbf{x})^T\mathbf{R}^{-1}(r_j - \mathbf{H}\hat{q}_j) := \hat{r}_j(\mathbf{x})
\end{aligned}$$

$$\begin{aligned}
&\text{Var}[r_j(\mathbf{x}) | r_j, s^2_j, \hat{g}] \\
&= \mathbf{E}[\text{Var}[r_j(\mathbf{x}) | r_j, q_j, s^2_j, \hat{g}]] + \text{Var}[\mathbf{E}[r_j(\mathbf{x}) | r_j, q_j, s^2_j, \hat{g}]] \\
&= \mathbf{E}[f_j^{-1}K(\mathbf{x},\mathbf{x}) | r_j, s^2_j, \hat{g}] + \text{Var}[\mathbf{h}(\mathbf{x})q_j + \mathbf{r}(\mathbf{x})^T\mathbf{R}^{-1}(r_j - \mathbf{H}q_j) | r_j, s^2_j, \hat{g}] \\
&= f_j^{-1}K(\mathbf{x},\mathbf{x}) + (\mathbf{h}(\mathbf{x}) - \mathbf{r}(\mathbf{x})^T\mathbf{R}^{-1}\mathbf{H})(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}(\mathbf{h}(\mathbf{x}) - \mathbf{r}(\mathbf{x})^T\mathbf{R}^{-1}\mathbf{H})^T := f_j^{-1}K_j(\mathbf{x},\mathbf{x}).
\end{aligned}$$

Lastly, we marginalize out f_j using lemma 1 shown at the end of this section with $D = 1$, $m = \hat{r}_j(\mathbf{x})$, $S = K(\mathbf{x}, \mathbf{x})$, $n = n - q$ and $s^2 = (m - q)^{-1}(\boldsymbol{r}_j - \mathbf{H}\boldsymbol{q}_j)^T \mathbf{R}^{-1}(\boldsymbol{r}_j - \mathbf{H}\boldsymbol{q}_j)$ to get the results shown in Eq. (5).

Lemma 1. Let $\boldsymbol{\gamma} | f \sim MN(\mathbf{m}, \mathbf{S}/f)$ (D-variate normal) and $f \sim Gamma(n/2, ns^2/2)$. Then $\boldsymbol{\gamma}$ has a D dimensional multivariate t distribution

$$\boldsymbol{\gamma} \sim t(\mathbf{m}, s^2\mathbf{S}, n),$$

with density

$$p(\boldsymbol{\gamma}) \propto \left[1 + \frac{1}{n}\frac{(\boldsymbol{\gamma} - \mathbf{m})^T \mathbf{S}^{-1} (\boldsymbol{\gamma} - \mathbf{m})}{\hat{s}^2}\right]^{-\frac{D+n}{2}}.$$

Proof. The joint density of $\boldsymbol{\gamma}$ and f is

$$p(\boldsymbol{\gamma}, f) \propto f^{D/2} \exp\left(-\frac{f}{2}(\boldsymbol{\gamma} - \mathbf{m})^T \mathbf{S}^{-1}(\boldsymbol{\gamma} - \mathbf{m})\right) f^{n/2-1} \exp(-nf\hat{s}^2/2) \propto$$
$$f^{(D+v)/2-1} \exp\left(-\frac{f}{2}((\boldsymbol{\gamma} - \mathbf{m})^T \mathbf{S}^{-1}(\boldsymbol{\gamma} - \mathbf{m}) + n\hat{s}^2)\right),$$

which is a $Gamma((D + v)/2, ((\boldsymbol{\gamma} - \mathbf{m})^T \mathbf{S}^{-1}(\boldsymbol{\gamma} - \mathbf{m}) + n\hat{s}^2)/2$. Then the distribution of $\boldsymbol{\gamma}$ follows from marginalizing out f:

$$\begin{aligned} p(\boldsymbol{\gamma}) &\propto \int p(\boldsymbol{\gamma}, f)\, df \\ &\propto ((\boldsymbol{\gamma} - \mathbf{m})^T \mathbf{S}^{-1}(\boldsymbol{\gamma} - \mathbf{m}) + n\hat{s}^2)^{-(n+D)/2} \\ &\propto (1 + (\boldsymbol{\gamma} - \mathbf{m})^T (\hat{s}^2\mathbf{S})^{-1}(\boldsymbol{\gamma} - \mathbf{m})/n)^{-(n+D)/2}, \end{aligned}$$

which gives the D dimensional multivariate t distribution $\mathbf{t}(\mathbf{m}, \hat{s}^2 S, n)$. □

2. OUT OF SAMPLE PREDICTION RESULTS OF PARTICLE DENSITY AND GRAND POTENTIAL

The performance of the ALEC emulator is compared to the performance with three other designs: random sampling using the same number of training size as the initial training size in ALEC (RS1), random sampling using the same number of training size as the final training size in ALEC (RS2), and active learning algorithm with D-optimality criterion. Suppose we have a total of n testing samples remaining after running the model. The performance is compared based on root mean squared error (RMSE) of density and energy, and 95% interval coverage and length. Suppose we have a total of n testing samples remaining after running the model, then the criteria we adopt are:

$$\begin{aligned} RMSE_r &= \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{k}(r_j(\mathbf{x}_i) - \hat{r}_j(\mathbf{x}_i))^2}{n\,k}}, \\ coverage_r &= \frac{1}{n\,k}\sum_{i=1}^{n}\sum_{j=1}^{k} 1\{r_j(\mathbf{x}_i) \in CI_{i,j}(95\%)\}, \\ length_r &= \frac{1}{n\,k}\sum_{i=1}^{n}\sum_{j=1}^{k} length\{CI_{i,j}(95\%)\}, \\ RMSE_W &= \sqrt{\frac{\sum_{i=1}^{n}(W(\mathbf{x}_i) - \hat{W}(\mathbf{x}_i))^2}{n}}, \end{aligned}$$

where $\hat{r}_j(\mathbf{x}_i)$ is the predicted density of ith hold out sample at jth coordinate, and $CI_{i,j}(95\%)$ is the 95% confidence interval of $\hat{r}_j(\mathbf{x}_i)$ based on the predictive distribution.

		training n	$RMSE_r$	$coverage_r$	$length_r$	$RMSE_W$
RS1	Group 1	20	$6.97\ 10^{6}$	87.7%	$2.47\ 10^{5}$	$2.60\ 10^{5}$
	Group 2	20	$3.62\ 10^{3}$	93.9%	$6.79\ 10^{3}$	$1.27\ 10^{2}$
	Group 3	20	$2.00\ 10^{2}$	92.0%	$8.81\ 10^{2}$	$7.07\ 10^{1}$
	Group 4	20	$8.81\ 10^{2}$	91.2%	$3.69\ 10^{1}$	1.24
		$n_{ini} + n_{aug}$	$RMSE_r$	$coverage_r$	$length_r$	$RMSE_W$
ALEC	Group 1	20+0	$6.97\ 10^{6}$	87.7%	$2.47\ 10^{5}$	$2.60\ 10^{5}$
	Group 2	20+7	$\mathbf{1.02\ 10^{3}}$	**95.6%**	$\mathbf{1.97\ 10^{3}}$	$\mathbf{3.73\ 10^{3}}$
	Group 3	20+21	$\mathbf{2.12\ 10^{3}}$	94.2%	$\mathbf{4.25\ 10^{3}}$	$\mathbf{1.43\ 10^{2}}$
	Group 4	20+504	$\mathbf{3.79\ 10^{3}}$	85.1%	$\mathbf{6.61\ 10^{3}}$	$\mathbf{1.62\ 10^{2}}$
		training n	$RMSE_r$	$coverage_r$	$length_r$	$RMSE_W$
RS2	Group 1	20	$6.97\ 10^{6}$	87.7%	$2.47\ 10^{5}$	$2.60\ 10^{5}$
	Group 2	27	$1.71\ 10^{3}$	96.3%	$2.53\ 10^{3}$	$5.05\ 10^{3}$
	Group 3	41	$6.89\ 10^{3}$	**94.9%**	$8.97\ 10^{3}$	$2.16\ 10^{1}$
	Group 4	524	$1.14\ 10^{1}$	**96.6%**	$3.94\ 10^{1}$	$4.20\ 10^{1}$
		$n_{ini} + n_{aug}$	$RMSE_r$	$coverage_r$	$length_r$	$RMSE_W$
D-opt	Group 1	20+0	$6.97\ 10^{6}$	87.7%	$2.47\ 10^{5}$	$2.60\ 10^{5}$
	Group 2	20+7	$1.44\ 10^{3}$	94.9%	$2.27\ 10^{3}$	$4.32\ 10^{3}$
	Group 3	20+21	$3.00\ 10^{3}$	93.6%	$4.53\ 10^{3}$	$1.43\ 10^{2}$
	Group 4	20+582	$1.37\ 10^{1}$	98.1%	$3.73\ 10^{1}$	$5.64\ 10^{1}$

Table S1. The prediction results for building models on four groups separately. In ALEC emulator, we ues $d = 0.01$ and $a = 0.05$ as threshold. In D-opt, we adjust the threshold c_{th} such that it has the similar number of augmented samples in AL, and c_{th} for group 1-4 are, 30, 3.4, 2.4 and 1.03, respectively.

		training n	$RMSE_r$	$coverage_r$	$length_r$	$RMSE_W$
RS1	All	80	$1.02\ 10^{1}$	**94.7**%	$3.44\ 10^{1}$	$4.70\ 10^{1}$
	Group 1	24	$1.45\ 10^{4}$	94.9%	$2.66\ 10^{4}$	$4.13\ 10^{4}$
	Group 2	15	$2.18\ 10^{3}$	93.0%	$1.94\ 10^{3}$	$8.57\ 10^{3}$
	Group 3	23	$7.92\ 10^{3}$	99.5%	$7.07\ 10^{2}$	$5.37\ 10^{2}$
	Group 4	18	$2.04\ 10^{1}$	91.4%	1.30	$9.38\ 10^{1}$
		$n_{ini} + n_{aug}$	$RMSE_r$	$coverage_r$	$length_r$	$RMSE_W$
ALEC	All	80+873	$\mathbf{8.76\ 10^{4}}$	91.1%	$\mathbf{1.70\ 10^{3}}$	$\mathbf{4.14\ 10^{3}}$
	Group 1	24+0	$1.30\ 10^{4}$	90.6%	$1.25\ 10^{4}$	$3.49\ 10^{4}$
	Group 2	15+1	$1.13\ 10^{3}$	80.3%	$7.11\ 10^{4}$	$6.93\ 10^{3}$
	Group 3	23+58	$5.12\ 10^{4}$	99.1%	$2.84\ 10^{3}$	$2.46\ 10^{3}$
	Group 4	18+814	$1.42\ 10^{3}$	97.0%	$4.17\ 10^{3}$	$3.39\ 10^{3}$
		training n	$RMSE_r$	$coverage_r$	$length_r$	$RMSE_W$
RS2	All	953	$3.25\ 10^{2}$	93.3%	$8.42\ 10^{2}$	$4.64\ 10^{1}$
	Group 1	245	$8.29\ 10^{6}$	74.3%	$1.16\ 10^{5}$	$1.01\ 10^{5}$
	Group 2	221	$6.50\ 10^{5}$	100%	$7.25\ 10^{4}$	$1.78\ 10^{3}$
	Group 3	238	$3.76\ 10^{3}$	100%	$2.92\ 10^{2}$	$2.67\ 10^{2}$
	Group 4	249	$6.51\ 10^{2}$	98.7%	$3.09\ 10^{1}$	$9.31\ 10^{1}$
		$n_{ini} + n_{aug}$	$RMSE_r$	$coverage_r$	$length_r$	$RMSE_W$
D-opt	All	80+935	$8.91\ 10^{2}$	98.3%	$7.60\ 10^{2}$	$3.31\ 10^{1}$
	Group 1	24+59	$2.11\ 10^{5}$	92.8%	$1.42\ 10^{4}$	$3.77\ 10^{5}$
	Group 2	15+73	$1.37\ 10^{4}$	98.0%	$2.29\ 10^{4}$	$4.90\ 10^{4}$
	Group 3	23+393	$2.23\ 10^{4}$	99.9%	$1.05\ 10^{3}$	$7.53\ 10^{4}$
	Group 4	18+410	$1.88\ 10^{1}$	97.1%	$3.36\ 10^{1}$	$6.98\ 10^{1}$

Table S2. The prediction results for building models on four groups together. We use $d = 0.01$ and $\alpha = 0.05$ in the ALEC emulator. In D-opt, the threshold $c_{th} = 1.12$. For each strategy, we evaluate the overall performance, as well as the performance for each subgroup.

ALEC		$n_{ini} + n_{aug}$	$RMSE_r$	$coverage_r$	$length_r$	$RMSE_W$
average	Group 1	20+90	$5.04\ 10^{-3}$	83.9%	$4.87\ 10^{-3}$	$1.96\ 10^{-2}$
	Group 2	27+120	$2.05\ 10^{-3}$	92.2%	$4.02\ 10^{-3}$	$6.08\ 10^{-3}$
	Group 3	41+66	$4.47\ 10^{-3}$	87.7%	$4.91\ 10^{-3}$	$1.47\ 10^{-2}$
	Group 4	524+47	$1.48\ 10^{-2}$	69.7%	$6.11\ 10^{-3}$	$6.46\ 10^{-2}$
	All groups	953+15	$4.68\ 10^{-3}$	77.7%	$3.42\ 10^{-3}$	$1.83\ 10^{-2}$
maximum	Group 1	20+408	$4.34\ 10^{-4}$	94.9%	$4.87\ 10^{-3}$	$7.89\ 10^{-4}$
	Group 2	27+395	$4.69\ 10^{-4}$	95.0%	$4.02\ 10^{-3}$	$7.85\ 10^{-4}$
	Group 3	41+384	$4.39\ 10^{-4}$	95.1%	$4.91\ 10^{-3}$	$7.70\ 10^{-4}$
	Group 4	524+361	$5.73\ 10^{-4}$	90.5%	$6.11\ 10^{-3}$	$7.15\ 10^{-4}$
	All groups	953+268	$5.18\ 10^{-4}$	92.5%	$3.42\ 10^{-3}$	$7.24\ 10^{-4}$

Table S3. The prediction results of the ALEC approach, using average-sense criterion and maximum-sense criterion, on a new functional form that is not in the training data set, starting with the final GP emulator developed for groups 1-4 and all groups combined. $d = 0.01$ and $a = 0.05$ are used.