University of California
Santa Barbara

Efficient Forward Prediction and Inverse Optimization in High-dimensional Spaces with Physical Constraints

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Statistics and Applied Probability

by

Hao Li

Committee in charge:

Professor Mengyang Gu, Chair
Professor Michael Ludkovski
Professor Ruimeng Hu
Professor M. Scott Shell

December 2023

The Dissertation of Hao Li is approved.

Professor Michael Ludkovski

Professor Ruimeng Hu

Professor M. Scott Shell

Professor Mengyang Gu, Committee Chair

October 2023

Efficient Forward Prediction and Inverse Optimization in High-dimensional Spaces with Physical Constraints

by

Hao Li

To my cherished wife, whose unwavering love and warm companionship have filled the past five years with countless sweet memories.

To my beloved parents, for their love, encouragement, and steadfast support in my personal growth.

To my dear sister, for her boundless care, invaluable guidance, and unwavering support.

And lastly, to my sweetheart daughter, whose innocence and chuckles illuminate the path to the next chapter in my life.

Acknowledgements

I would like to sincerely express my appreciation to the following individuals and groups who have played pivotal roles in my academic journey and the completion of this doctoral research:

First and foremost, I extend my heartfelt gratitude to my advisor, Prof. Mengyang Gu, for his unwavering support, insightful guidance, and passionate mentorship throughout my doctoral research. Our numerous meetings, thought-provoking discussions, and extensive email exchanges have played a pivotal role in shaping the foundation of this dissertation. My sincere thanks also go to my committee members, Prof. M. Scott Shell, Prof. Michael Ludkovski, and Prof. Ruimeng Hu for their invaluable contributions to my research. Their expertise, feedback, and willingness to answer my questions have been invaluable in the successful completion of my research projects.

I would like to thank all my professors at UCSB for their dedicated teaching and valuable advice during my five years of study. Their help has been instrumental in shaping my academic and research abilities. I extend my gratitude to Prof. Jianzhong Wu and Dr. Musen Zhou, for their support in data generation and their assistance with background knowledge in the field of chemistry. Their collaboration significantly enriched the scope and depth of my research.

I would like to thank my manager, Scott Geller, for his guidance and support on my career path planning during my internship. I am grateful to my mentors, Hui Yu, and Jeff Bliss, for their professional guidance and support on my internship project. I extend my appreciation to my team buddy, Yijun Wu, for the support and assistance during those three months.

I am deeply appreciative of my friends, roommates, and neighbors who have enriched my life in numerous ways. Their support, encouragement, and shared experiences have

provided balance and inspiration during this challenging academic journey.

This dissertation would not have been possible without the collective support and encouragement of these individuals and groups. Each of you has made an indelible mark on my academic and personal growth, and I am immensely grateful for your contributions to this milestone in my life.

Curriculum Vitæ

Hao Li

Education

2023	Ph.D. in Statistics and Applied Probability (Expected), University of California, Santa Barbara.
2018	M.S. in Financial Mathematics, University of Minnesota, Twin Cities.
2016	B.E. in Electrical Engineering, Nankai University.
2016	B.S. in Finance, Nankai University.

Experience

2018 - 2023	Teaching Assistant, Department of Statistics and Applied Probability, University of California, Santa Barbara.
2020, 2021	Graduate Student Researcher, Department of Statistics and Applied Probability, University of California, Santa Barbara.
2022	Data Scientist Intern, Lyft Inc.

Publications

"Efficient force field and energy emulation through partition of permutationally equivalent atoms." Li, Hao, Musen Zhou, Jessalyn Sebastian, Jianzhong Wu, and Mengyang Gu. The Journal of Chemical Physics 156, no. 18 (2022).

Presentation

Efficient Machine Learning Force-fields, Graduate Student Research Showcase, University of California, Santa Barbara, CA, March 2021.

Abstract

Efficient Forward Prediction and Inverse Optimization in High-dimensional Spaces with Physical Constraints

by

Hao Li

Computer models and simulators are widely used for understanding complex processes, whereas the computational cost can be large. This thesis introduces new efficient surrogate models for predicting expensive simulations, and adaptive designs for inversely optimizing system properties, particularly focusing on scenarios where the dimension of the inputs or outputs is large. One such scenario is the molecular simulation, which is an indispensable component in understanding the intricate relationship between molecular configurations and their associated properties. Constructing an efficient surrogate model can expedite predictions of molecular behaviors of many new chemical structures. Furthermore, it can accelerate discoveries in critical fields such as materials design, drug discovery, and chemoinformatics, through efficient designs by utilizing predictions from surrogate models and their associated assessment of prediction uncertainty.

Gaussian process (GP) emulator has been used as a surrogate model for both scalar-valued and vectorized outputs from computer models. In applications like predicting molecular force fields and potential energy in *ab initio* molecular dynamics simulation, the GPs can substantially improve predictive accuracy using both gradient information and functional values whereas conventional approximation methods may not work well. The computational cost of GPs, however, can be substantial. To address this challenge, Chapter 2 introduces a new approach, termed the atomized force field (AFF) model, which combines force and energy prediction in a computationally efficient manner. This

model establishes a forward mapping from molecule configuration to the molecular force field and potential energy, substantially reducing computational demands by exploiting the naturally sparse covariance structure that adheres to energy conservation and atom permutation symmetry constraints. Built upon the AFF model, Chapter 3 explores novel methods to construct reverse mapping of 3D molecule structures from molecular force fields and potential energy. This approach offers fast solutions to some applications, such as finding the equilibrium state of the molecular through the minimization of atomic forces. Additionally, a Python package of the surrogate model called PyRobustGaSP is introduced in Chapter 4, for emulating computer models with massive data with robust parameter estimation and predictions.

Contents

Chapter 1

Introduction

Computer models or simulators are widely used for generating data to understand natural and social processes. Molecular simulations provide access to microscopic details of physical, chemical, and biological processes. Computationally scalable methods, such as classical molecular dynamics (cMD), often lack accuracy, while *ab initio* molecular dynamics (AIMD) prioritizes accuracy over computational scalability to provide more detailed molecular simulations. In principle, machine learning (ML) approaches can provide a surrogate model to achieve accuracy at the levels of AIMD and computational scalability even faster than cMD, thus providing new applications that would not be achievable by conventional simulations. While recent years have witnessed enormous development in ML potentials, the field is still rapidly evolving. Many theoretical and computational issues remain to be addressed to efficiently represent potential-energy surfaces, particularly in the context of computationally expensive simulations for large systems. The first goal of this thesis is to construct a computationally scalable surrogate model that efficiently learns the mapping between system properties, including the potential energy and atomic forces, and input parameters such as atom positions and types, all within a high-dimensional space. Our particular focus is on the utilization of physical constraints

to enhance the computational scalability and efficiency of the estimation process. Statistical or ML emulators, which harness machine learning and statistical techniques for predicting molecular properties, were developed in recent years for predicting chemical properties [1, 2, 3, 4, 5, 6, 7, 8]. These emulators established a forward mapping from chemical space to functional space. In [5, 6], the local atomic neighborhood information is employed as input to emulate the local atomic energy and then approximate the total energy. Global molecule information, such as pairwise atomic distance matrix, is used in [3, 4] and has shown the ability to emulate the energy and atomic force with AIMD accuracy. However, the large computational complexity concerning the size of the molecule restricts its application to larger systems. Our proposed method reduces the computational cost while maintaining the same level of accuracy by leveraging the physical constraints.

Estimating an inverse map from system properties to the input space is also another critical task in various applications, including energy minimization, materials design, and drug design. For example, the inverse map of potential energy and atomic force field may help find the minimized energy molecule structure in energy minimization. In other cases, the inverse map of molecule properties can facilitate the new molecules with desired properties [9, 10]. The exhaustive exploration of the extensive chemical space is unattainable due to the prohibitively high computational cost imposed by the intricate, high-dimensional nature of large 3D molecular structures. Drawing from those data-driven approaches that comprehend the forwarding mapping from sampled molecular configurations, the challenge of the inverse mapping can be reframed as an optimization problem, subject to specific constraints rooted in the physical setting. The Bayesian optimization (BO) is designed to optimize functions that are source-intensive to evaluate, and it doesn't depend on the gradient information, which is often challenging to obtain for certain black-box functions. A secondary goal of this thesis is to introduce an innovative

approach that leverages the proposed forward method and Bayesian optimization. This approach aims to unravel the inverse mapping from potential energy and atomic force fields back to molecular configurations.

Many existing works have developed statistical or machine learning emulators using Gaussian processes [6, 11, 12, 13, 14, 15, 16] and neural networks [7, 8, 17, 18, 19, 20]. In general, accurate calculations of molecular dynamic simulation often come at substantial computational costs. This limitation restricts the scale of sampled configurations for training emulators. Consequently, kernel methods such as Gaussian process regression (GPR) and kernel ridge regression (KRR), known for their proficiency in making precise predictions with fewer samples, have gained popularity. GPR-based emulators not only offer flexible and efficient predictions of approximating complex, high-dimensional functions but also provide information about their uncertainties. This makes them invaluable in various fields such as physical sciences, geostatistics, and surrogate modeling. The third goal of this thesis is to develop a Python-based software package to facilitate the construction of fast and reliable GP emulators. In the forthcoming section, we will provide a brief overview of the Gaussian process emulator and highlight some of its features that we will later utilize in subsequent chapters.

1.1 Scalar-valued Gaussian Process Emulator

To begin, consider $y(\mathbf{x}) \in \mathbb{R}$ as a real-valued outcome associated with a p-dimensional vector of real-valued inputs $\mathbf{x} \in \mathbb{R}^p$. Initially, let us assume $y(\mathbf{x})$ represents a deterministic output generated by a computationally expensive simulator when provided with the given inputs. We will later extend the model to accommodate stochastic outputs, which may arise from numerical errors in the simulator or from observation data that includes measurement noise. A Gaussian stochastic process (GaSP) or Gaussian process (GP) models

this real-value output $y(\cdot)$ as an unknown random function $y(\cdot) \sim \text{GP}\left(\mu(\cdot),\, \sigma^2 K(\cdot,\cdot)\right)$ with a mean function $\mu(\cdot)$, variance parameter σ^2, and correlation function $K\left(\cdot,\cdot\right)$. The set of physical inputs $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, where $\mathbf{x}_i$ is a $p \times 1$ vector, is commonly selected using some "space filling" technique, such as Latin-hypercube design [21, 22], over the input domain. For any physical inputs $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, the marginal distribution follows a multivariate normal distribution

$$\left(y(\mathbf{x}_1), \ldots, y(\mathbf{x}_M)\right)^T \mid \boldsymbol{\theta}, \sigma^2, \mathbf{R} \sim \mathcal{MN}\left(\left(\mu(\mathbf{x}_1), \ldots, \mu(\mathbf{x}_M)\right)^T, \sigma^2 \mathbf{R}\right), \tag{1.1}$$

where σ^2 is the unknown variance and $\mathbf{R}$ is the correlation matrix with (i, j) element modeled by a correlation function $K\left(\mathbf{x}_i, \mathbf{x}_j\right)$. It is common to model the mean function $\mu(\cdot)$ via linear combinations of basis functions,

$$\mu(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\theta} = \sum_{t=1}^{q} h_t(\mathbf{x})\theta_t, \tag{1.2}$$

where $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), h_2(\mathbf{x}), \cdots, h_q(\mathbf{x}))$ is a vector of basis functions and $\boldsymbol{\theta} = (\theta_1, ..., \theta_q)^T$ is a vector of unknown trend parameters. The GP model without the noise is an "interpolator", which means the prediction of the model at an observed or design input $\mathbf{x}_i$ is exactly the same as the output value at any design point. This property will be shown in Section 1.1.2. The function $K(\cdot,\cdot)$ is often referred to as a kernel function or a correlation function. It encodes prior knowledge about the underlying function $y(\cdot)$, such as the smoothness and periodicity. This correlation function typically gets larger when two inputs get closer in the input space, indicating that the functional values are more similar for inputs that are closer to each other. This property holds true for any continuous function, as the difference between functional values diminishes when two inputs approach each other.

Generally, there are two types of correlation functions. The first type is known as the isotropic correlation function, where the correlation function is a function of the Euclidean distance between any two inputs

$$K(\mathbf{x}_i, \mathbf{x}_j) = K(||\mathbf{x}_i - \mathbf{x}_i||), \tag{1.3}$$

where $||\mathbf{x}_i - \mathbf{x}_j|| = (\sum_{l=1}^{p}(x_{il} - x_{jl})^2)^{1/2}$ is the Euclidean distance between any inputs $\mathbf{x}_i$ and $\mathbf{x}_j$. The isotropic correlation function is commonly used in spatial statistics. The Euclidean distance metric is also used for emulating molecular simulation for input parameters such as positions of atoms [2, 4, 23].

Another widely used correlation for computer model emulation and calibration is the product correlation [24, 25, 26], which is a product of p one-dimensional correlation functions:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \prod_{l=1}^{p} K_l(x_{il}, x_{jl}), \tag{1.4}$$

with $K_l(\cdot, \cdot)$ being a one-dimensional correlation function for the l-th coordinate of the input vector. With the product correlation function, the correlation matrix can be written as follows

$$\mathbf{R} = \mathbf{R}_1 \circ \mathbf{R}_2 \circ \cdots \circ \mathbf{R}_p, \tag{1.5}$$

where $\mathbf{R}_l$ is the correlation matrix with (i,j)-th element being $K_l(x_{il}, x_{jl})$, for $l = 1, ..., p$, and $\circ$ denotes the Hadamard product.

Table 1.1 provides an overview of commonly used correlation functions for the product correlation at any coordinate l. The isotropic correlation functions can be similarly defined by using the Euclidean distance and having one range and one roughness parameter for each correlation. One of the most frequently employed correlation functions is the power exponential correlation function with $\alpha = 2$, commonly referred to as Gaussian

correlation. This choice of correlation function imparts infinite differentiability to the GP process, a desirable property in many applications. However, the correlation matrix with a Gaussian correlation function often exhibits a low rank [27]. This low-rank property prohibits computing the inversion of the covariance required for computing the likelihood function and predictive distribution. To overcome this difficulty, the power exponential correlation function with α_l close to 2, such as $\alpha_l = 1.9$ [28], is often used in practice. However, a GP with power exponential correlation with $\alpha_l < 2$ is not once differential, which may not be ideal for some applications. Another popular choice of correlation is the Matérn correlation function. The Matérn correlation function possesses a closed-form expression when the roughness parameter takes the form $\alpha_l = \frac{2k+1}{2}$, where $k \in \mathbb{N}$. For instance, the Matérn correlation function with $\alpha_l = \frac{1}{2}$ is the power exponential correlation function with $\alpha_l = 1$. As α_l goes to infinity, the Matérn correlation function converges to the Gaussian correlation. Notably, a GP with the Matérn correlation function is $\lceil \alpha_l \rceil - 1$ times mean squared differentiable [29], an appealing property as the differentiability of the process is directly controlled by the roughness parameter. One of the widely used Matérn correlation functions is the one with $\alpha_l = \frac{5}{2}$, which has the expression below

$$K(x_{il}, x_{jl}) = \left(1 + \frac{\sqrt{5}d_l}{\gamma_l} + \frac{5d_l^2}{3\gamma_l^2}\right) \exp\left(\frac{\sqrt{5}d_l}{\gamma_l}\right), \tag{1.6}$$

where $d_l = |x_{il} - x_{jl}|$ in the product correlation function. The range parameter γ_l typically controls the decay of pointwise correlations for l-th coordinate as a function of distance. The larger value of the range parameter implies the correlation between two points is large. In contrast, a smaller value of the range parameter means the correlation is smaller. In building a statistical emulator, the roughness parameter α is typically predefined, and the range parameter γ_l is typically estimated from data.

Let $\mathbf{y} = (y(\mathbf{x}_1), \ldots, y(\mathbf{x}_M))^T$ be a vector of observations from the simulator at M

Table 1.1: Popular choices of correlation functions K_l using in Equation (1.4) for $l = 1, ..., p$. Here, α_l is a roughness parameter, γ_l is a range parameter and $d_l = |x_{il} - x_{jl}|$ is the distance between the lth coordinate of two inputs. $\Gamma(\cdot)$ is the gamma function and $\mathcal{K}(\cdot)$ is the modified Bessel function of the second kind.

Correlation Function	**Formula**
Power exponential	$\exp\left(-(\frac{d_l}{\gamma_l})^{\alpha_l}\right), \quad \alpha_l \in (0, 2]$
Matérn	$\frac{1}{2^{\alpha_l - 1}\Gamma(\alpha_l)} \left(\frac{d_l}{\gamma_l}\right)^{\alpha_l} \mathcal{K}_{\alpha_l}\left(\frac{d_l}{\gamma_l}\right), \quad \alpha_l \in (0, +\infty)$
Rational Quadratic	$\left(1 + (\frac{d_l}{\gamma_l})^2\right)^{-\alpha_l}, \quad \alpha_l \in (0, +\infty)$
Spherical	$\left(1 - 1.5(\frac{d_l}{\gamma_l}) + 0.5(\frac{d_l}{\gamma_l})^3\right) \mathbf{1}_{[d_l/\gamma_l \leq 1]}$

inputs $\mathbf{x}_1, ..., \mathbf{x}_M$, and let $y(\mathbf{x}^*)$ be the output at an input $\mathbf{x}^*$ to be predicted. Conditional on all parameters, the joint distribution $\mathbf{y}$ and $y(\mathbf{x}^*)$ follows

$$\begin{pmatrix} \mathbf{y} \\ y(\mathbf{x}^*) \end{pmatrix} \sim \mathcal{MN}\left(\begin{pmatrix} \mathbf{H}\boldsymbol{\theta} \\ \mathbf{h}(\mathbf{x}^*)\boldsymbol{\theta} \end{pmatrix}, \sigma^2 \begin{pmatrix} \mathbf{R} & \mathbf{r}(\mathbf{x}^*) \\ \mathbf{r}^T(\mathbf{x}^*) & K(\mathbf{x}^*, \mathbf{x}^*) \end{pmatrix}\right), \tag{1.7}$$

where $\mathbf{H} = \left(\mathbf{h}^T(\mathbf{x}_1), ..., \mathbf{h}^T(\mathbf{x}_M)\right)^T$ is an $M \times q$ dimensional basis matrix, and $\mathbf{r}(\mathbf{x}^*) = (K(\mathbf{x}^*, \mathbf{x}_1), \cdots, K(\mathbf{x}^*, \mathbf{x}_M))^T$ is a column vector with size M.

1.1.1 Parameter Estimation of Gaussian Process Emulation

In this section, we review parameter estimation in a GP emulator, using the general product correlation function as an example. The parameter estimation for the isotropic correlation is similar. The parameters include the mean parameters $\boldsymbol{\theta} = (\theta_1, ..., \theta_p)^T$, σ^2 and range parameter $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_p)^T$ is the correlation. The maximum likelihood method is one of the most frequently used approaches for parameter estimation. The

likelihood function in the GP model follows

$$\mathcal{L}(\boldsymbol{\theta}, \sigma^2, \boldsymbol{\gamma}) = (2\pi\sigma^2)^{-\frac{M}{2}} |\mathbf{R}|^{-\frac{1}{2}} \exp\left(-\frac{(\mathbf{y} - \mathbf{H}\boldsymbol{\theta})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{H}\boldsymbol{\theta})}{2\sigma^2}\right). \tag{1.8}$$

Given the range parameters, the MLE of $\boldsymbol{\theta}$ and σ^2 follows

$$\hat{\boldsymbol{\theta}}_{MLE} = (\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{R}^{-1} \mathbf{y}, \tag{1.9}$$

and

$$\hat{\sigma}^2_{MLE} = \frac{S^2}{M}, \tag{1.10}$$

where $S^2 = (\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\theta}}_{MLE})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\theta}}_{MLE})$. After plugging the maximum likelihood estimator $(\hat{\boldsymbol{\theta}}_{MLE}, \hat{\sigma}^2_{MLE})$, the likelihood function in Equation (1.8) reduces to the profile likelihood after ignoring the constant term

$$\mathcal{L}(\hat{\boldsymbol{\theta}}_{MLE}, \hat{\sigma}^2_{MLE}, \boldsymbol{\gamma}) \propto |\mathbf{R}|^{-\frac{1}{2}} (S^2)^{-\frac{M}{2}}. \tag{1.11}$$

The MLE of $\boldsymbol{\gamma}$ typically does not have a closed-form expression. These parameters are often estimated through numerical optimization by maximizing the profile likelihood

$$\hat{\boldsymbol{\gamma}}_{MLE} = \underset{\boldsymbol{\gamma}}{\operatorname{argmax}} \left\{ \mathcal{L}(\hat{\boldsymbol{\theta}}_{MLE}, \hat{\sigma}^2_{MLE}, \boldsymbol{\gamma}) \right\}. \tag{1.12}$$

Quasi-Newton optimization methods [30, 31] are often used for numerical optimization in Equation (1.12).

The MLE of the parameters for GPs has been found to be unstable, particularly when the sample size is large or when the underlying functions have many modes [32, 33, 34]. As a result, it is common to utilize a Bayesian prior to obtain robust estimation. The

objective prior of the GP model used in practice has the expression below [35]

$$\pi(\boldsymbol{\theta}, \sigma^2, \boldsymbol{\gamma}) \propto \frac{\pi(\boldsymbol{\gamma})}{\sigma^2}, \tag{1.13}$$

where $\pi(\boldsymbol{\gamma})$ is a prior of the range parameters. Integrating out the mean and variance parameters, $p(\mathbf{y} \mid \boldsymbol{\gamma}) \propto \int p(\mathbf{y} \mid \boldsymbol{\theta}, \sigma^2, \boldsymbol{\gamma}) \pi(\boldsymbol{\theta}, \sigma^2, \boldsymbol{\gamma}) d\boldsymbol{\theta} d\sigma^2$, the marginal likelihood follows [35]

$$\mathcal{L}(\boldsymbol{\gamma}) \propto |\mathbf{R}|^{-\frac{1}{2}} |\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}|^{-\frac{1}{2}} \left(S^2\right)^{-\frac{(M-q)}{2}}, \tag{1.14}$$

where $S^2 = (\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\theta}})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\theta}})$ and $\hat{\boldsymbol{\theta}} = \left(\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}\right)^{-1} \mathbf{H}^T \mathbf{R}^{-1} \mathbf{y}$. The range parameters of the correlation function were sometimes estimated by maximizing the marginal likelihood function in Equation (1.14), and the resulting estimator is commonly referred to as the maximum marginal likelihood estimator (MMLE) [36]. However, both MLE and MMLE can be unstable in practice [34], as the maximum value of $\boldsymbol{\gamma}$ can approach the boundary of the parameter space for either the profile likelihood in Equation (1.11) or the marginal likelihood in Equation (1.14).

One approach that helps reduce unstable estimations is to use the maximum marginal posterior model estimator with robust parameterization of the reference prior [34]. The reference prior of the GP model with isotropic covariances is introduced in [35]. It has been shown to yield a proper posterior distribution, whereas the commonly used Jeffreys prior can lead to an improper posterior distribution. The reference prior has also been extended for the GP models with noisy measurements [37, 38], product kernel functions [39] and vectorized outputs [40]. The reference prior of a GP model with a product kernel function has the expression below

$$\pi^R(\boldsymbol{\theta}, \sigma^2, \boldsymbol{\gamma}) \propto \frac{\pi^R(\boldsymbol{\gamma})}{\sigma^2}, \tag{1.15}$$

with $\pi^R(\boldsymbol{\gamma}) \propto |\mathbf{I}^*(\boldsymbol{\gamma})|^{\frac{1}{2}}$, where $\mathbf{I}^*(\cdot)$ is the expected Fisher information matrix of the marginal likelihood given by:

$$\mathbf{I}^*(\boldsymbol{\gamma}) = \begin{pmatrix} M-q & \operatorname{tr}(\mathbf{W}_1) & \operatorname{tr}(\mathbf{W}_2) & \cdots & \operatorname{tr}(\mathbf{W}_p) \\ & \operatorname{tr}(\mathbf{W}_1^2) & \operatorname{tr}(\mathbf{W}_1\mathbf{W}_2) & \cdots & \operatorname{tr}(\mathbf{W}_1\mathbf{W}_p) \\ & & \operatorname{tr}(\mathbf{W}_2^2) & \cdots & \operatorname{tr}(\mathbf{W}_2\mathbf{W}_p) \\ & & & \ddots & \vdots \\ & & & & \operatorname{tr}(\mathbf{W}_p^2) \end{pmatrix}, \tag{1.16}$$

where $\mathbf{W}_l = \dot{\mathbf{R}}_l\mathbf{Q}$, $\dot{\mathbf{R}}_l$ is the partial derivative of the correlation matrix $\mathbf{R}$ with respect to the l-th range parameter for $1 \leq l \leq p$, and $\mathbf{Q} = \mathbf{R}^{-1}\mathbf{P}_\mathbf{R}$ with $\mathbf{P}_\mathbf{R} = \mathbf{I}_M - \mathbf{H}\left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1}$. The marginal posterior of $\boldsymbol{\gamma}$ with regard to this reference prior is

$$p(\boldsymbol{\gamma} \mid \mathbf{y}) \propto L(\boldsymbol{\gamma} \mid \mathbf{y})|\mathbf{I}^*(\boldsymbol{\gamma})|^{\frac{1}{2}}. \tag{1.17}$$

The Metropolis algorithm can be used to sample the posterior for full Bayesian analysis, whereas computing the marginal posterior function takes $O(M^3)$ operations, which can be prohibitively slow when we need thousands of posterior samples. Thus, it is common to estimate $\boldsymbol{\gamma}$ using its marginal posterior mode given by

$$(\hat{\gamma}_1, \cdots, \hat{\gamma}_p) = \underset{(\gamma_1,\cdots,\gamma_p)}{\operatorname{argmax}} \{p(\boldsymbol{\gamma} \mid \mathbf{y})\}, \tag{1.18}$$

where the estimator is often referred to as the maximum marginal posterior estimator (MMPE). As shown in [34], the marginal posterior with reference prior using parameter $\boldsymbol{\gamma}$ or log inverse parameters $\xi_l = \log(1/\gamma_l)$ for $l = 1, ..., p$ typically results in robust estimation. Computing the reference prior can be more computationally expensive than the marginal likelihood. To reduce the computational cost, another prior called the jointly

robust (JR) prior was introduced in [41] for the inverse range parameter $\beta_l = 1/\gamma_l$. The overall JR prior follows $\pi^{JR}(\boldsymbol{\theta}, \sigma^2, \boldsymbol{\beta}) \propto \pi^{JR}(\boldsymbol{\beta})/\sigma^2$, where

$$\pi^{JR}(\boldsymbol{\beta}) \propto \left(\sum_{l=1}^{p} \beta_l\right)^a \exp(-bC_l\beta_l), \tag{1.19}$$

with a, b and C_l being the prior parameters for $l = 1, ..., p$. The JR prior is a proper prior with closed-form normalizing constant and the MMPE with the JR prior can help reduce unstable estimation from MLE and MMLE. After obtaining the estimation $\hat{\boldsymbol{\beta}}$ by MMPE, the estimation of range parameter can be obtained $\hat{\gamma}_l = 1/\hat{\beta}_l$ for $l = 1, ..., p$.

Another commonly used approach for estimating range parameters in the machine learning community is cross-validation (CV) [2, 3]. In this method, the observed dataset is initially randomized and divided into m folds. Each fold serves as a validation set, while the remaining $m-1$ folds as the training set. The optimal values of these parameters are then determined by numerically minimizing the predicted error by the predictions (which will be introduced in Section 1.1.2 below) and the validation set. The cross-validation reduces the number of samples, which can be less efficient than the mode estimator, such as MLE, MMLE or MMPE, particularly when the sample size is small.

Directly calculating the prior likelihood or marginal likelihood function involves matrix inversion which can be unstable when the conditioner number of the covariance matrix is large. A more numerically stable way is to utilize Cholesky decomposition of the covariance matrix and then compute the prediction distribution by solving a linear system of equations with both forward and backward steps, as described in [42].

1.1.2 Predictive Distributions of Gaussian Processes

After obtaining the estimates of range parameters $\boldsymbol{\gamma}$, the predictive distribution of $y(\mathbf{x}^*)$ for a new input $\mathbf{x}^*$, given $\mathbf{y}$, can be obtained by integrating out the parameters:

$$p(y(\mathbf{x}^*) \mid \mathbf{y}, \hat{\boldsymbol{\gamma}}) = \int p(y(\mathbf{x}^*) \mid \mathbf{y}, \boldsymbol{\theta}, \sigma^2, \hat{\boldsymbol{\gamma}}) p(\boldsymbol{\theta}, \sigma^2) d\boldsymbol{\theta} d\sigma^2. \tag{1.20}$$

Using the reference prior for the mean and variance parameters, $p(\boldsymbol{\theta}, \sigma^2) \propto 1/\sigma^2$, the predictive distribution follows a Student's t distribution with $M - q$ degrees of freedom

$$y(\mathbf{x}^*) \mid \mathbf{y}, \hat{\boldsymbol{\gamma}} \sim \mathcal{T}\left(\hat{y}(\mathbf{x}^*), \hat{\sigma}^2 K^{**}(\mathbf{x}^*, \mathbf{x}^*), M - q\right), \tag{1.21}$$

where

$$\hat{y}(\mathbf{x}^*) = \mathbf{h}(\mathbf{x}^*)\hat{\boldsymbol{\theta}} + \mathbf{r}^T(\mathbf{x}^*)\mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\theta}}\right), \tag{1.22}$$

$$\hat{\sigma}^2 = \frac{1}{(M-q)}\left(\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\theta}}\right)^T \mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\theta}}\right), \tag{1.23}$$

and

$$\begin{aligned} K^{**}(\mathbf{x}^*, \mathbf{x}^*) =& K(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{r}^T(\mathbf{x}^*)\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*) + \left(\mathbf{h}(\mathbf{x}^*) - \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}\mathbf{H}\right) \\ & \times \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\left(\mathbf{h}(\mathbf{x}^*) - \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}\mathbf{H}\right)^T, \end{aligned} \tag{1.24}$$

and $\hat{\boldsymbol{\theta}} = (\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{y}$ is the generalized least squares estimator for $\boldsymbol{\theta}$, which has the same expression of the MLE with the same range parameters. The predictive mean $\hat{y}(\mathbf{x}^*)$ is often used for predictions. Another advantage of a GP model is the availability of uncertainty assessment of the prediction, as any posterior predictive interval can be computed based on the predictive distribution in Equation (1.21).

The prediction of a GP model with noise-free measurement is an interpolator, meaning that the prediction is exactly the same as the output value. This is because for $\mathbf{x}^* = \mathbf{x}_i$,

we have

$$\begin{aligned}\hat{y}(\mathbf{x}^*) = \hat{y}(\mathbf{x}_i) &= \mathbf{h}(\mathbf{x}_i)\hat{\boldsymbol{\theta}} + \mathbf{r}^T(\mathbf{x}_i)\mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\theta}}\right) \\ &= \mathbf{h}(\mathbf{x}_i)\hat{\boldsymbol{\theta}} + \mathbf{e}_i^T(\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\theta}}) \\ &= y(\mathbf{x}_i),\end{aligned}$$

where $\mathbf{e}_i$ is a vector with value 1 in the i-th row and zeros in all other rows.

1.1.3 Gaussian Process Emulation with Noisy Measurements

In practice, $\mathbf{y} = (y(\mathbf{x}_1), \ldots, y(\mathbf{x}_M))^T$ may contain noise due to numerical errors in the simulator output or measurement errors in field observations. The output of the simulator may not change much when some input variables change. These inputs, often called the "inert inputs", can be omitted when constructing a statistical emulator [43, 41]. For all these scenarios, it is typical to include an independent normally distributed noise in the model to account for additional uncertainty in observations:

$$\tilde{y}(\mathbf{x}) = y(\mathbf{x}) + \epsilon(\mathbf{x}), \tag{1.25}$$

where $\epsilon(\mathbf{x}) \sim \mathcal{N}(0, \sigma_0^2)$ is an independent noise and $y(\cdot)$ is modeled as a GP as before: $y(\cdot) \sim \mathrm{GP}\left(\mu(\cdot),\, \sigma^2 K(\cdot,\cdot)\right)$. The covariance function for $\tilde{y}(\cdot)$ at any observations $\mathbf{x}_i$ and $\mathbf{x}_j$ follows

$$\sigma^2 \tilde{K}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \left(K(\mathbf{x}_i, \mathbf{x}_j) + \lambda 1_{\mathbf{x}_i = \mathbf{x}_j}\right). \tag{1.26}$$

Here, $\lambda = \sigma_0^2/\sigma^2$ is a parameter that represents the ratio of noise and signal variances, often referred to as a nugget parameter, and $1_{\mathbf{x}_i = \mathbf{x}_j}$ is an indicator function which equals

to 1 if $\mathbf{x}_i = \mathbf{x}_j$, and 0 otherwise. The covariance matrix becomes

$$\sigma^2\tilde{\mathbf{R}} = \sigma^2(\mathbf{R} + \lambda\mathbf{I}_M), \tag{1.27}$$

where $\mathbf{I}_M$ represents the identity matrix of size M. The reference prior for the GP model with the product correlation function follows [34, 37, 38]

$$\pi^R\left(\boldsymbol{\theta}, \sigma^2, \boldsymbol{\gamma}, \lambda\right) = \pi^R\left(\boldsymbol{\theta}, \sigma^2\right)\pi^R\left(\boldsymbol{\gamma}, \lambda \mid \boldsymbol{\theta}, \sigma^2\right) \propto \frac{\pi^R(\boldsymbol{\gamma}, \lambda)}{\sigma^2}. \tag{1.28}$$

Here, $\pi\left(\boldsymbol{\theta}, \sigma^2, \boldsymbol{\gamma}, \lambda\right) \propto |\tilde{\mathbf{I}}^*(\boldsymbol{\gamma}, \lambda)|^{\frac{1}{2}}$, and the corresponding expected Fisher information matrix has the following form:

$$\tilde{\mathbf{I}}^*(\boldsymbol{\gamma}, \lambda) = \begin{pmatrix} M-q & \operatorname{tr}\left(\tilde{\mathbf{W}}_1\right) & \operatorname{tr}\left(\tilde{\mathbf{W}}_2\right) & \cdots & \operatorname{tr}\left(\tilde{\mathbf{W}}_{p+1}\right) \\ & \operatorname{tr}\left(\tilde{\mathbf{W}}_1^2\right) & \operatorname{tr}\left(\tilde{\mathbf{W}}_1\tilde{\mathbf{W}}_2\right) & \cdots & \operatorname{tr}\left(\tilde{\mathbf{W}}_1\tilde{\mathbf{W}}_{p+1}\right) \\ & & \operatorname{tr}\left(\tilde{\mathbf{W}}_2^2\right) & \cdots & \operatorname{tr}\left(\tilde{\mathbf{W}}_2\tilde{\mathbf{W}}_{p+1}\right) \\ & & & \ddots & \vdots \\ & & & & \operatorname{tr}\left(\tilde{\mathbf{W}}_{p+1}^2\right) \end{pmatrix}, \tag{1.29}$$

where $\tilde{\mathbf{W}}_l = \dot{\tilde{\mathbf{R}}}_l\tilde{\mathbf{Q}}$, for $1 \le l \le p$, and $\dot{\tilde{\mathbf{R}}}_l$ represents the partial derivative of the correlation matrix $\tilde{\mathbf{R}}$ with respect to the l-th range parameter. Additionally, $\tilde{\mathbf{Q}} = \tilde{\mathbf{R}}^{-1}\mathbf{P}_{\tilde{\mathbf{R}}}$ with $\mathbf{P}_{\tilde{\mathbf{R}}} = \mathbf{I}_M - \mathbf{H}\left(\mathbf{H}^T\tilde{\mathbf{R}}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\tilde{\mathbf{R}}^{-1}$, and $\tilde{\mathbf{W}}_{p+1} = \tilde{\mathbf{Q}}$. Similarly, the nugget and range parameters can be estimated through the marginal posterior mode using

$$\left(\hat{\gamma}_1, \cdots, \hat{\gamma}_p, \hat{\lambda}\right) = \underset{(\gamma_1, \cdots, \gamma_p, \lambda)}{\operatorname{argmax}} \left\{L\left(\boldsymbol{\gamma}, \lambda\right)\pi^R\left(\boldsymbol{\gamma}, \lambda\right)\right\}. \tag{1.30}$$

The robust parameterization for a GP model of noisy measurements with reference prior is studied in [34]. The jointly robust prior was also developed for simplifying the

computation for GP models with noisy measurements, which has the expression below

$$\pi^{JR}(\boldsymbol{\beta}, \lambda) \propto \left(\sum_{l=1}^{p} \beta_l + \lambda\right)^a \exp(-b(C_l\beta_l + \lambda)), \tag{1.31}$$

where a, b and C_l are prior parameters for $l = 1, ..., p$. The estimation of $\boldsymbol{\gamma}$ can be obtained by transforming the estimation back.

The predictive distribution for GP models with noisy measurements is similar to the one without the noise in Equation (1.21) through replacing $\mathbf{R}$ by $\tilde{\mathbf{R}}$, as follows:

$$y(\mathbf{x}^*) \mid \mathbf{y}, \hat{\boldsymbol{\gamma}}, \hat{\lambda} \sim \mathcal{T}\left(\hat{y}(\mathbf{x}^*), \hat{\sigma}^2 K^{**}(\mathbf{x}^*, \mathbf{x}^*), M - q\right), \tag{1.32}$$

where

$$\hat{y}(\mathbf{x}^*) = \mathbf{h}(\mathbf{x}^*)\hat{\boldsymbol{\theta}} + \mathbf{r}^T(\mathbf{x}^*)\tilde{\mathbf{R}}^{-1}\left(\mathbf{y} - \mathbf{H}\tilde{\boldsymbol{\theta}}\right), \tag{1.33}$$

$$\hat{\sigma}^2 = \frac{1}{(M-q)}\left(\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\theta}}\right)^T \tilde{\mathbf{R}}^{-1}\left(\mathbf{y} - \mathbf{H}\tilde{\boldsymbol{\theta}}\right), \tag{1.34}$$

$$\begin{aligned} K^{**}(\mathbf{x}^*, \mathbf{x}^*) = K(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{r}^T(\mathbf{x}^*)\tilde{\mathbf{R}}^{-1}\mathbf{r}(\mathbf{x}^*) + \left(\mathbf{h}(\mathbf{x}^*) - \mathbf{r}(\mathbf{x}^*)^T\tilde{\mathbf{R}}^{-1}\mathbf{H}\right) \\ \times \left(\mathbf{H}^T\tilde{\mathbf{R}}^{-1}\mathbf{H}\right)^{-1}\left(\mathbf{h}(\mathbf{x}^*) - \mathbf{r}(\mathbf{x}^*)^T\tilde{\mathbf{R}}^{-1}\mathbf{H}\right)^T, \end{aligned} \tag{1.35}$$

with the generalized least squared estimator $\tilde{\boldsymbol{\theta}} = (\mathbf{H}^T\tilde{\mathbf{R}}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\tilde{\mathbf{R}}^{-1}\mathbf{y}$.

The GP models for scalar-valued noisy outputs were implemented in the RobustGaSP package available in `R` [44] and `MATLAB` [45], and in `Python` [46] developed in this thesis.

1.1.4 Connection with Kernel Ridge Regression

The predictive mean in a GP model with noisy measurements is closely connected to the kernel ridge regression estimator. Consider $\mathcal{X}$ as the p-dimensional input domain. We define the reproducing kernel Hilbert space (RKHS) as $\mathcal{H}$, which serves as the completion

of the space of all functions given by:

$$\mathbf{x} \to \sum_{i=1}^{m_1} w_i K(\mathbf{x}_i, \mathbf{x}), \quad w_1, \cdots, w_{m_1} \in \mathbb{R},\ \mathbf{x}_1, \cdots, \mathbf{x}_{m_1}, \mathbf{x} \in \mathcal{X},\ m_1 \in \mathbb{N}. \tag{1.36}$$

This space is equipped with an inner product:

$$\left\langle \sum_{i=1}^{m_1} w_i K\left(\mathbf{x}_i, \cdot\right), \sum_{j=1}^{m_2} w_j K\left(\mathbf{x}_j, \cdot\right) \right\rangle_{\mathcal{H}} = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} w_i w_j K\left(\mathbf{x}_i, \mathbf{x}_j\right). \tag{1.37}$$

For any function $f(\cdot) \in \mathcal{H}$, the RKHS norm is denoted as $\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}$. Furthermore, due to the bounded linearity of evaluation maps in the RKHS, it can be deduced from the Riesz representation theorem that $f(\mathbf{x}) = \langle f(\cdot), K(\cdot, \mathbf{x}) \rangle_{\mathcal{H}}$ for each $\mathbf{x} \in \mathcal{X}$ and $f(\cdot) \in \mathcal{H}$.

Let $L_2(\mathcal{X})$ denote the space of square-integrable functions $f : \mathcal{X} \to \mathbb{R}$ with $\int_{\mathbf{x} \in \mathcal{X}} f^2(\mathbf{x}) d\mathbf{x} < \infty$, and $\langle f, g \rangle_{L_2(\mathcal{X})} := \int_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) g(\mathbf{x}) d\mathbf{x}$ represents the inner product in $L_2(\mathbf{x})$. According to Mercer's theorem, there exists an orthonormal sequence of continuous eigenfunctions $\{\phi_j\}_{j=1}^{\infty}$ with a corresponding sequence of non-increasing the non-negative eigenvalues $\{\rho_j\}_{j=1}^{\infty}$ such that

$$K\left(\mathbf{x}_a, \mathbf{x}_b\right) = \sum_{j=1}^{\infty} \rho_j \phi_j\left(\mathbf{x}_a\right) \phi_j\left(\mathbf{x}_b\right) \tag{1.38}$$

for any $\mathbf{x}_a, \mathbf{x}_b \in \mathcal{X}$.

The RKHS $\mathcal{H}$ encompasses all functions $f(\cdot) = \sum_{j=1}^{\infty} f_j \phi_j(\cdot) \in L_2(\mathcal{X})$ with $f_j = \langle f, \phi_j \rangle_{L_2(\mathcal{X})}$ and $\sum_{j=1}^{\infty} f_j^2 / \rho_j < \infty$. For any $g(\cdot) = \sum_{j=1}^{\infty} g_j \phi_j(\cdot) \in \mathcal{H}$ and $f(\cdot)$, the inner product can be expressed as $\langle f, g \rangle_{\mathcal{H}} = \sum_{j=1}^{\infty} f_j g_j / \rho_j$. We refer to Chapter 1 of [47] for more details on the properties of the RKHS.

Given a data set $\{(\tilde{y}_i, \mathbf{x}_i)\}_{i=1}^{M}$ comprising M observations with $\tilde{y}_i \in \mathbb{R}$ at input $\mathbf{x}_i \in \mathbb{R}^p$.

Consider the nonparametric regression process where

$$\tilde{y}_i = y(\mathbf{x}_i) + \epsilon(\mathbf{x}_i), \tag{1.39}$$

where $\epsilon(\mathbf{x}_i) \sim \mathcal{N}(0, \sigma_0^2)$ is an independent Gaussian noise with variance σ_0^2 and $y(\cdot)$ is an unknown deterministic function. The kernel ridge regression (KRR) [42, 48] estimates the unknown function $y(\cdot)$ by solving following optimization problem:

$$\hat{y} = \underset{y(\cdot)\in\mathcal{H}}{\operatorname{argmin}} \left\{ \frac{1}{M} \sum_{i=1}^{M} (\tilde{y}_i - y(\mathbf{x}_i))^2 + \tilde{\lambda} \|y\|_{\mathcal{H}}^2 \right\}, \tag{1.40}$$

where $\|\|_{\mathcal{H}}$ is the native norm or reproducing kernel Hibert space norm [49]. The solution simultaneously penalizes the fitting error expressed as the squared error loss and complexity of the solution characterized by the native norm of the function. The complexity can be considered as a measurement of the discontinuous level of the function. Given the same variance, a function with more local variability, the complexity of this function is typically larger than another function with fewer changes locally. The size of the penalty of the native norm is determined by the kernel function or the associated RKHS.

According to the representer lemma [29], for any $\mathbf{x}^*$, we have:

$$y(\mathbf{x}^*) = \sum_{i=1}^{M} w_i K(\mathbf{x}_i, \mathbf{x}^*). \tag{1.41}$$

Let $\mathbf{w} = (w_1, \cdots, w_M)^T$. Since $\langle K(\mathbf{x}_i, \cdot), K(\mathbf{x}_j, \cdot) \rangle_{\mathcal{H}} = K(\mathbf{x}_i, \mathbf{x}_j)$, Equation (1.40) can be simplified as follows

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left\{ \frac{1}{M} (\mathbf{y} - \mathbf{R}\mathbf{w})^T (\mathbf{y} - \mathbf{R}\mathbf{w}) + \tilde{\lambda} \mathbf{w}^T \mathbf{R} \mathbf{w} \right\}. \tag{1.42}$$

The solution of the weights follows

$$\hat{\mathbf{w}} = (\mathbf{R} + M\tilde{\lambda}\mathbf{I}_M)^{-1}\mathbf{y}. \tag{1.43}$$

Plugging the solution of $\mathbf{w}$ into Equation (1.42), the KRR estimator of $y(\mathbf{x}^*)$ takes the following form:

$$y(\mathbf{x}^*) = \mathbf{r}^T(\mathbf{x}^*)\left(\mathbf{R} + M\tilde{\lambda}\mathbf{I}_M\right)^{-1}\mathbf{y}. \tag{1.44}$$

This expression is identical to the predictive mean from the GP model of noisy measurements in Equation (1.32) when the mean parameter $\boldsymbol{\theta}$ is zero and nugget parameter $\lambda = M\tilde{\lambda}$. The equivalence of KRR and GP can be extended to models with a mean function and discrepancy functions by a GP [50] or a scaled Gaussian process in model calibration [51].

The intriguing connection underscores the interplay between GP and KRR, providing another way to interpret the predictive mean estimator in the GP model as the solution of the minimization problem in Equation (1.40). It is worth noting that the GP regression also offers closed-form predictive intervals with almost no additional cost in computation, which is useful for uncertainty assessment of predictions.

1.2 Gaussian Processes for Vectorized Outputs

Many computer simulations and real-world examples contain vectorized outputs. For instance, the TITAN2D simulator generates pyroclastic flow properties, such as flow heights at a massive number of space-time coordinates at each set of input variables [40, 52]. In molecular simulations, the atomic forces in a molecule with N atoms are represented as a 3N-dimensional vector in Cartesian coordinates.

Denote $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_M]$ a $k \times M$ observational matrix where $\mathbf{y}_i = (y_1(\mathbf{x}_i), ..., y_k(\mathbf{x}_i))^T$

for $i = 1, ..., M$. A few GP models were used for modeling the vectorized outputs. The most straightforward approach involves building a separate GP at each coordinate, requiring the estimation of k sets of parameters individually. However, this can be both time-consuming and unstable. Another way is to use a separable covariance structure, where the covariance follows $\text{Cov}(\mathbf{Y}) = \mathbf{\Sigma} \otimes \mathbf{R}$ where $\otimes$ denotes the Kronecker product, $\mathbf{\Sigma}$ is a $k \times k$ covariance matrix between coordinates and $\mathbf{R}$ is a correlation matrix across M training runs [53]. Linear models of coregionalization were also used for modeling vectorized outputs [26, 54, 55], where covariance is generally not separable.

Here we introduce the parallel partial Gaussian process (PP-GP), which is scalable to computer simulation with a massive number of observations, as detailed in [40]. The PP-GP model has inspired the development of the Atomized force field model, which will be introduced in Chapter 2. In the PP-GP model, each output coordinate contains a set of the mean $\mu_j(\mathbf{x})$ and variance parameters σ^2, which can be estimated from the data. The mean can be modeled by a linear combination of the basis functions:

$$\mu_j(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\theta}_j, \tag{1.45}$$

where $\mathbf{h}(\mathbf{x})$ is a row vector of q basis functions, and $\boldsymbol{\theta}_j$ is a q vector of the mean parameters. The $M \times M$ correlation matrix $\mathbf{R}$ is assumed to be shared across all coordinates in the PP-GP model. The assumptions of PP-GP greatly simplify the computation and improve estimation stability, as only one set of range parameters needs to be numerically estimated and all other parameters can be estimated using closed-form expressions.

The reference prior as shown in Equation (1.13) for the mean and variance parameters has the following expression

$$\pi^R(\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_k, \sigma_1^2, \cdots, \sigma_k^2, \boldsymbol{\gamma}) \propto \frac{\pi^R(\boldsymbol{\gamma})}{\prod_{j=1}^k \sigma_j^2}. \tag{1.46}$$

Similar to the scalar-valued GP model, one can integrate out the mean and variance parameters and estimate the range parameter in the kernel to obtain an estimation of the range parameter $\boldsymbol{\gamma}$. As $\hat{\boldsymbol{\gamma}}$ is the same across different coordinates, the predictive distribution of PP-GP at a new input $\mathbf{x}^*$ at any j-th coordinate follows a Student's t distribution

$$y_j(\mathbf{x}^*) \mid \mathbf{y}_j, \hat{\boldsymbol{\gamma}} \sim \mathcal{T}\left(\hat{y}_j(\mathbf{x}^*), \hat{\sigma}_j^2 K^{**}(\mathbf{x}^*, \mathbf{x}^*), M-q\right), \tag{1.47}$$

where

$$\hat{y}_j(\mathbf{x}^*) = \mathbf{h}(\mathbf{x}^*)\hat{\boldsymbol{\theta}}_j + \mathbf{r}^T(\mathbf{x}^*)\mathbf{R}^{-1}\left(\mathbf{y}_j - \mathbf{H}\hat{\boldsymbol{\theta}}_j\right), \tag{1.48}$$

$$\hat{\sigma}_j^2 = (M-q)^{-1}\left(\mathbf{y}_j - \mathbf{H}\hat{\boldsymbol{\theta}}_j\right)^T \mathbf{R}^{-1}\left(\mathbf{y}_j - \mathbf{H}\hat{\boldsymbol{\theta}}_j\right), \tag{1.49}$$

$$\begin{aligned} K^{**}(\mathbf{x}^*, \mathbf{x}^*) = K(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{r}^T(\mathbf{x}^*)\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*) + \left(\mathbf{h}(\mathbf{x}^*) - \mathbf{H}^T\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*)\right)^T \\ \times \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\left(\mathbf{h}(\mathbf{x}^*) - \mathbf{H}^T\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*)\right), \end{aligned} \tag{1.50}$$

with the generalized least squares estimator of the mean follows $\hat{\boldsymbol{\theta}}_j = \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{y}_j$, and other terms were defined in the scalar-valued GP model. Estimation and predictions of PP-GP models for noisy measurements follow similarly by replacing $\mathbf{R}$ by $\tilde{\mathbf{R}} = \mathbf{R} + \lambda\mathbf{I}_n$ in the marginal likelihood and predictive distributions, respectively.

PP-GP provides a general approach for approximating nonlinear map $\mathbf{y}(\mathbf{x})$: $\mathbb{R}^p \to \mathbb{R}^k$, which can be used for predicting high-dimensional maps in computer model simulations [40] and forecasting dynamical systems [56]. The PP-GP only requires $\mathcal{O}(M^3) + O(Mk)$ operations for computing the predictive mean, which is substantially faster than some other alternatives.

The GP emulator of scalar-valued outputs and the PP-GP emulator for vectorized outputs were implemented in the RobustGaSP package originally available in `R` and `MATLAB` [44, 45]. We developed the `Python` version of the RobustGaSP package, which will be

introduced in Chapter 4.

1.3 Gaussian Processes on Linear Functional Observations

As the Gaussian distribution is closed under any linear transformations, one of the appealing properties of a GP model is that the joint distribution and predictive distribution under the linear transformation of functions can be easily derived. Suppose we have a linear operator $\mathcal{L}$ acting on realizations of $y(\cdot) \sim \mathrm{GP}\left(\mu(\cdot), \sigma^2 K(\cdot,\cdot)\right)$. In this context, it indicates that $\mathcal{L}y(\cdot)$ continues to follow a Gaussian process. This feature enables us to conduct statistical inference about linear functionals of an unknown function modeled by GP, such as weighted averages and differential operations.

Assume that the operator $\mathcal{L}$ produces functions in $\mathbb{R}^k$ with an input in $\mathbb{R}$. The mean and covariance of $\mathcal{L}y$ are given by applying $\mathcal{L}$ to the mean and covariance of the argument:

$$\mathbb{E}[\mathcal{L}y(\mathbf{x})] = \mathcal{L}\mu(\mathbf{x}) : \mathbb{R} \to \mathbb{R}^k, \tag{1.51}$$

$$\mathrm{Cov}(\mathcal{L}y(\mathbf{x}), \mathcal{L}y(\mathbf{x}')) = \mathcal{L}\sigma^2 K(\mathbf{x}, \mathbf{x}')\mathcal{L}^T : \mathbb{R} \to \mathbb{R}^{k\times k}, \tag{1.52}$$

and the covariance follows

$$\mathrm{Cov}\left(\mathcal{L}y(\mathbf{x}), y(\mathbf{x}')\right) = \mathcal{L}\sigma^2 K(\mathbf{x}, \mathbf{x}') : \mathbb{R} \to \mathbb{R}^k. \tag{1.53}$$

Taking the directional gradient operator $\nabla_{\mathbf{x}} : \mathbb{R} \to \mathbb{R}^k$ as an example, where $\mathbf{x} \in \mathbb{R}^p$ and

$p = k$. The joint distribution of $y(\mathbf{x})$ and $\nabla_{\mathbf{x}} y(\mathbf{x})$ follows

$$\begin{pmatrix} y(\mathbf{x}) \\ \nabla_{\mathbf{x}} y(\mathbf{x}) \end{pmatrix} \sim \mathcal{MN}\left(\begin{pmatrix} \mu(\mathbf{x}) \\ \nabla_{\mathbf{x}} \mu(\mathbf{x}) \end{pmatrix}, \sigma^2 \begin{pmatrix} K(\mathbf{x}, \mathbf{x}') & \nabla_{\mathbf{x}} K(\mathbf{x}, \mathbf{x}') \\ \nabla_{\mathbf{x}'} K(\mathbf{x}', \mathbf{x}) & \nabla_{\mathbf{x}} K(\mathbf{x}, \mathbf{x}') \nabla_{\mathbf{x}'}^T \end{pmatrix}\right), \tag{1.54}$$

where $\nabla_{\mathbf{x}} K(\mathbf{x}, \mathbf{x}')$ is $1 \times k$ Jacobian matrix and $\nabla_{\mathbf{x}} K(\mathbf{x}, \mathbf{x}') \nabla_{\mathbf{x}'}^T$ is $k \times k$ Hessian matrix.

1.4 Outline

Chapter 2 introduces an efficient method called the atomic force field (AFF) for emulating atomic forces and potential energy in *ab initio* molecular dynamics simulation. We begin with an overview of the relevant methods for emulating atomic forces and discuss the computational challenges of these methods. The AFF method is then introduced for reducing the computational cost for large molecules by utilizing the natural sparsity of the designed correlation matrix of GP and maintaining high accuracy in predictions. Closed-form expressions for predictions and uncertainty quantification will be introduced in the context of emulating both force and energy. We will compare the AFF with alternatives in terms of computational costs and predictive accuracy using real-world simulators.

Chapter 3 proposes new methods that utilize Bayesian Optimization for optimizing the properties of the system with a small number of iterations. Notably, this method does not require gradient information. Recent studies of Bayesian optimization are mostly demonstrated for systems with a low dimensional input. We show the effectiveness of the new method for optimizing a large dimensional input space. We found directly minimizing the energy based on Bayesian optimization leads to unstable estimation. To address this problem, we introduce a novel inverse force design approach, showcasing its effectiveness in force and energy minimization. This method is both robust and accurate. Only a few iterations are needed to obtain the equilibrium state of the simulation. Numerical

comparison will be provided for comparing energy-minimized structures obtained via inverse force design and those generated through the simulation.

Chapter 4 discusses a `Python` package PyRobustGaSP for both scalar and vectorized outputs. We also cover the atomic force field emulator for predicting atomic force and potential energy. We provide examples to illustrate a few new applications, such as emulating high dimensional 3D electron density. Additionally, we provide example codes for emulation for computer models with both scalar outputs and a massive number of observations.

Chapter 5 summarizes the developments and outlines future work related to developing efficient statistical emulators and inverse designs with high dimensional input space. We highlight the importance of the dimension reduction approach for reducing high-dimensional input spaces, particularly in the context of molecular simulation.

Chapter 2

Forward Efficient Force Field and Energy Emulation through Partition of Permutationally Equivalent Atoms

Molecular dynamics (MD) [57] is a computer simulation method for analyzing the physical movements of atoms and molecules, and it has greatly enhanced our understanding of complex chemical and biological systems. The fundamental concept underlying MD simulations is relatively straightforward. When provided with the positions of all atoms within a molecular system, it becomes possible to compute the forces acting on each atom due to all other atoms. By applying Newton's laws of motion, one can predict the spatial coordinates of each atom as a function of time. This process involves progressing through discrete time increments, iteratively calculating the forces acting on each atom, and subsequently utilizing these forces to update both the position and velocity of every atom. In classical MD simulations, empirical potential energy functions and classical

force fields are employed. Nevertheless, a notable challenge that remains is the accuracy of underlying classical interatomic potentials, which restricts the ability to achieve truly predictive modeling of dynamics and functionality of molecular systems. One potential solution is to increase the accuracy level by AIMD simulation. AIMD simulations are rooted in quantum mechanics and rely on electronic structure theory, such as density functional theory (DFT), to compute the electronic wave function and potential energy of the system. AIMD simulations calculate quantum-mechanical forces for atomic configurations at each time step, employing density-functional approximations to solve the Schrödinger equation for a system comprising both nuclei and electrons. However, it's important to acknowledge that the AIMD simulations come with a substantial computational burden, surpassing that of classical molecular dynamics. Consequently, AIMD is typically confined to smaller systems and shorter timeframes. To address the dual challenges of accuracy and computational cost, there arises a need for a fast and precise emulator. Such an emulator can facilitate the construction of force fields with a high accuracy level of *ab initio* calculations, bridging the gap between computational efficiency and the need for precise modeling. In other words, we require an efficient method that can learn from AIMD calculations and accurately predict the potential energy and atomic forces at various timesteps from atomic positions, as plotted in Figure 2.1. The goal of the surrogate model is to achieve accuracy comparable to AIMD simulations in computing potential energy and atomic forces for a given configuration while substantially reducing computational costs.

In this chapter, we propose a new surrogate model, which is called the atomized force field (AFF) emulator, for predicting the atomic force and the potential energy in AIMD simulations. By utilizing the physical permutationally equivalent property of the molecules and the natural sparsity from the designed correlation function, we demonstrate that the atomized force field is computationally more scalable than prior approaches.

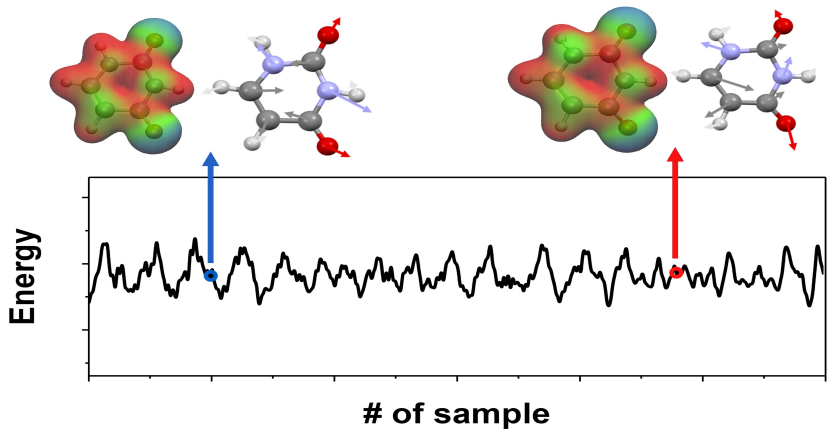

Figure 2.1: Potential energy and atomic forces of the uracil molecule from the AIMD simulation.

2.1 Literature Review

In recent years, many approaches have been proposed to learn the map between the molecule's configuration and the molecule-level information. Deep neural network (DNN) and Gaussian process regression (GPR) are two popular tools to emulate AIMD simulation containing a large number of single atoms or small molecules (such as H_2O) [5, 12, 58]. Some effective machine-learning approaches have been developed to emulate the dynamics of molecules containing a larger number of atoms with different types. The kernel ridge regression approach, which is equivalent to the predictive mean of GPR, for instance, was proposed to emulate the potential energies of organic molecules, based on pairwise diatomic positions and nuclear charge [23]. The Gaussian approximation potential framework (GAP) [6], as another example, approximates the total energy functional through the decomposition of local atomic energy functional by using self-designed atomic neighborhood information. This approach is often used along with the smooth overlap of atomic positions (SOAP) [5] to measure the local atomic neighborhood information, such that predictions satisfy translational, permutational, and rotational symmetries of atoms. The inducing point sparse approximation [59] is often used to improve compu-

tational scalability in these approaches. DNN architectures have also been developed to emulate AIMD [7, 17, 60], where a large number of training samples were often used in training the model. The GPR typically requires fewer samples for accurate predictions, because of two reasons. First, GPR is a nonparametric model, and the complexity of the model, such as the number of basis in predictions, increases with the sample size, which makes it flexible to estimate the nonlinear response surface. Second, the predictive mean in GPR has a closed-form expression, and only a few parameters are required to be numerically estimated, whereas DNN typically relies on numerical optimization in a large parameter space. In both approaches, an appropriate descriptor that encodes the information of molecular geometry is important for predictions.

Combining force and energy samples with energy conservation constraints can improve the predictive accuracy of atomic forces and potential energies in AIMD simulation [2, 3, 4, 7, 8]. The approach, known as gradient-domain machine learning (GDML) [4], starts with the conservation of energy. Here, we denote $E(\mathbf{x})$ as the potential energy of the molecule configuration $\mathbf{x}$, which is the outcome of the computer simulation, denoted as $f(\mathbf{x})$ in Chapter 1 in general. We also denote $\mathbf{F}^i(\mathbf{x})$ the force vector of the i-th atom within the molecule configuration $\mathbf{x}$, another vectorized outcome from computer simulations to be predicted for $i = 1, ..., N$ in a system with N atoms. The 3D force vector of each atom $\mathbf{F}^i(\mathbf{x})$ is related to the potential energy $E(\mathbf{x})$ by

$$\mathbf{F}^i(\mathbf{x}) = -\nabla_{\mathbf{r}_i} E(\mathbf{x}), \tag{2.1}$$

where $\mathbf{x} = [\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N]$ is a $3 \times N$ matrix of atomic Cartesian coordinates for a system with N atoms, and $\mathbf{r}_i$ denotes the 3D coordinates for each atom. Given M configurations of a molecule containing N atoms, parameter estimation and predictions of the atomic forces by the GDML approach and the symmetric gradient-domain machine

learning (sGDML) [61] involve constructing and inverting a $3NM \times 3NM$ Hessian covariance matrix. The computational cost of constructing this covariance matrix scales as $\mathcal{O}(M^2N^3)$ and the cost of its inversion scales as $\mathcal{O}(M^3N^3)$, both increasing rapidly along with the number of atoms and the number of simulation runs for training the surrogate model. The large computational cost of the surrogate model prohibits predicting molecular information in larger and more complex systems.

A wide range of approximation methods for alleviating the computational cost of GP models has been proposed in recent years, including, for instance, the induced point approach [59], low-rank approximation [62], covariance tapering [63], hierarchical nearest neighbor methods [64], stochastic partial differential equation approach [65], and local Gaussian process approach [66]. Although these methods are useful for approximating GP models with observations at a low dimensional input space, none of them focuses on approximating GP models with high-dimensional gradient observations. The large computational cost prevents the direct applications of GP models with high-dimensional gradient information in large-scale systems, a problem which was recently realized in the statistics and machine learning communities [67]. Low-rank approximation and sparse approximation of the covariance were studied [68], yet the predictive accuracy can be degraded. The recent approach [67] reduces the computational complexity for GP with gradient observations with respect to the dimension of gradients, but the method requires $\mathcal{O}(M^6)$ computational operations, which is prohibitive for moderately large training runs M. Surprisingly, we found that after enforcing energy conservation and permutation symmetry of atoms onto the covariance function, the covariance matrix of atomic forces is approximately sparse. This property can be utilized to reduce computational operations substantially without sacrificing the accuracy of predictions.

2.2 Motivation

2.2.1 Invariance Constraints on Molecular Representation

Several essential requirements are important to consider when emulating potential energy and force fields, with a focus on rotational, translational, and permutational invariance. Rotational and translational invariance imply that the molecule-level information should remain unchanged following rotations or translations applied to the same molecular configuration. This consistency ensures that the measurement of the molecule's configuration remains identical, irrespective of the chosen coordinate system, as the physical configuration of the molecule itself remains unaltered. Meeting these two invariance requirements can be achieved through the construction of a representation, often referred to as a descriptor, possessing the desired properties. An example of such a descriptor is the inverse pairwise distances matrix of the atoms within the molecule, as utilized in previous work [4]. Denote $\mathbf{x} = [\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N]$ is a $3 \times N$ matrix of atomic Cartesian coordinates for a system with N atoms, and $\mathbf{r}_i$ denotes the 3D coordinates for each atom., the descriptor $\mathbf{D}(\mathbf{x})$ of $\mathbf{x}$ can be represented as:

$$\mathbf{D}(\mathbf{x})_{ij} = \begin{cases} \|\mathbf{r}_i - \mathbf{r}_j\|^{-1} & \text{for i} > \text{j}, \\ 0 & \text{o.w. ,} \end{cases} \tag{2.2}$$

where $||.||$ denotes the Euclidean distance. This representation is complete, meaning that it can accurately and uniquely describe any conceivable configuration of the system.

Another important consideration arises from the indistinguishability of identical atoms within a system. The potential energy surface (PES) of a chemical system relies solely on the charges and positions of the nuclei. Consequently, the PES exhibits symmetry under permutations of atoms with the same nuclear charge. In molecular simulations,

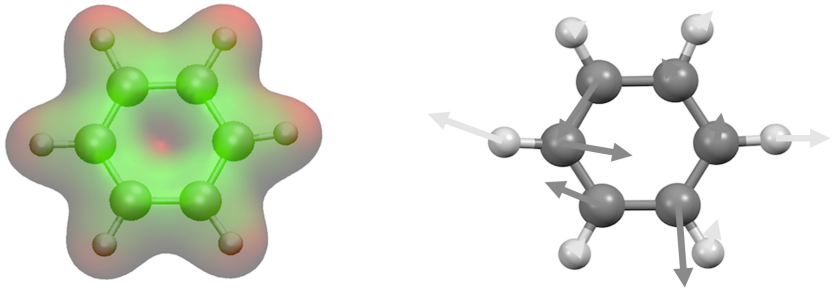

Figure 2.2: A benzene molecule with six carbon and six hydrogen atoms.

atoms may rotate or switch positions, and emulators that encode this physical symmetry information can achieve higher predictive accuracy [3, 5, 69, 70, 71]. Emulators based on machine learning methods that do not account for and treat these symmetries equivalently might yield varying predictions when atoms are permuted. Using the benzene molecule illustrated in Figure 2.2 as an example, it's important to note that all six carbon and six hydrogen atoms are physically equivalent. Consequently, there exist twelve distinct ways to arrange these twelve atoms within the same benzene molecule. This variability arises because the first atom can be any of the carbon atoms, and the arrangement can proceed either clockwise or counterclockwise.

2.2.2 GDML method based on KRR

GDML stands out as a pioneering method that incorporates the law of energy conservation in Equation (2.1) into the joint modeling of the force vector and scalar-valued potential energy within the kernel ridge regression framework. In contrast to methods that first emulate energy and then perform numerical differentiation to compute atomic forces, GDML represents a significant advancement, enhancing the accuracy of AIMD trajectories for small molecules. Moreover, it enables the development of force fields with accuracy levels akin to *ab initio* methods. The improved version of this method, sGDML

[3], takes into account permutation invariance, further elevating the accuracy threshold. In this section, we will provide a brief overview of GDML, explore its limitations, and delve into the underlying principles that form the basis for our proposed atomic force field method.

Given a dataset comprising M configurations of a molecule containing N atoms, denoted as $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M)$, The first step is to convert the 3D Cartesian coordinates of each molecule $\mathbf{x}_i$ into a corresponding descriptor $\mathbf{D}(\mathbf{x}_i)$ (as defined in Equation (2.2)). This transformation is essential for distinguishing between physically equivalent geometry in the Cartesian space. In the GDML method, the entire $3MN$ force vector of M configuration, denoted as $\mathbf{F}(\mathbf{X})$, is modeled by the following representation

$$\mathbf{F}(\mathbf{X}) = (\mathbf{K}_{Hess}(\mathbf{X}, \mathbf{X}) + \lambda \mathbf{I}_{3MN})\boldsymbol{V}, \tag{2.3}$$

where the Hessian matrix $\mathbf{K}_{Hess}(\mathbf{X}, \mathbf{X})$ is a $3MN \times 3MN$ matrix, with the (i,j)-th block matrix defined as $\nabla_{\mathbf{x}_i} K(\mathbf{D}(\mathbf{x}_i), \mathbf{D}(\mathbf{x}_j))\nabla^T_{\mathbf{x}_j}$, and $\boldsymbol{V}$ is the weight vector of dimension $3MN$ to be estimated. Various kernel functions can be used, and due to the smoothness and the response surface, the isotropic kernel function in Equation (1.3) with a Matérn kernel function with $\alpha = 5/2$ is used. Consequently, only one range parameter γ and one nugget parameter λ are required to be numerically estimated from the data. Furthermore, the input for the kernel function consists of the vectorized descriptors, in contrast to the original Cartesian coordinates, because of the translational and rotational invariance.

Direct computation through the chain rule gives

$$\nabla_{\mathbf{x}_a} K(\mathbf{D}(\mathbf{x}_a), \mathbf{D}(\mathbf{x}_b))\nabla^T_{\mathbf{x}_b} = \mathbf{J}_D(\mathbf{x}_a) \frac{\partial^2 K(\mathbf{D}(\mathbf{x}_a), \mathbf{D}(\mathbf{x}_b))}{\partial \mathbf{D}(\mathbf{x}_a) \partial \mathbf{D}(\mathbf{x}_b)} \mathbf{J}_D(\mathbf{x}_b)^T, \tag{2.4}$$

where $\mathbf{J}^T_{\mathbf{D}(\mathbf{x}_a)}$ is the Jacobian of the descriptor $\mathbf{D}(\mathbf{x}_a)$ w.r.t $\mathbf{x}_a$, and $\frac{\partial^2 K(\mathbf{D}(\mathbf{x}_a), \mathbf{D}(\mathbf{x}_b))}{\partial \mathbf{D}(\mathbf{x}_a) \partial \mathbf{D}(\mathbf{x}_b)}$ is the

Hessian matrix of $K(\mathbf{D}(\mathbf{x}_a), \mathbf{D}(\mathbf{x}_b))$ w.r.t $\mathbf{D}(\mathbf{x}_a)$ and $\mathbf{D}(\mathbf{x}_b)$. Based on the form of the descriptor matrix $\mathbf{D}(\mathbf{x}_a)$ in Equation (2.2), the gradient of (p, q)-th element of $\mathbf{D}(\mathbf{x}_a)$ w.r.t $\mathbf{r}_i$ follows

$$\frac{\partial \mathbf{D}(\mathbf{x}_a)_{pq}}{\partial \mathbf{r}_i} = \begin{cases} -\frac{\mathbf{r}_p - \mathbf{r}_q}{\|\mathbf{r}_p - \mathbf{r}_q\|^3} & p > q \text{ and } i = p, \\ \frac{\mathbf{r}_p - \mathbf{r}_q}{\|\mathbf{r}_p - \mathbf{r}_q\|^3} & p > q \text{ and } i = q, \\ 0 & \text{o.w .} \end{cases} \tag{2.5}$$

The trained force field estimator within this framework aggregates the contributions of partial derivatives, totaling $3N$, from all M training configurations to formulate the prediction. It can be expressed as:

$$\mathbf{F}(\mathbf{x}^*) = \sum_{i=1}^{M} \sum_{j=1}^{3N} \boldsymbol{V}_{ij} \frac{\partial}{\partial \mathbf{x}_j} \nabla_{\mathbf{x}^*} K(\mathbf{x}^*, \mathbf{x}_i). \tag{2.6}$$

The corresponding potential energy predictor is obtained by integrating $\mathbf{F}(\mathbf{x}^*)$ and is expressed as:

$$E(\mathbf{x}^*) = \sum_{i=1}^{M} \sum_{j=1}^{3N} \boldsymbol{V}_{ij} \frac{\partial}{\partial \mathbf{x}_j} K(\mathbf{x}^*, \mathbf{x}_i), \tag{2.7}$$

which describes how the potential energy predictor is derived from the integrated force field estimator.

The sGDML model builds upon the GDML model while taking into account the permutational symmetry of atoms. To treat all permutationally symmetric molecules as equivalent molecules, it employs a symmetric version of the kernel function defined as:

$$K_{sys}(\mathbf{x}_a, \mathbf{x}_b) = \frac{1}{S^2} \sum_{p=1}^{S} \sum_{q=1}^{S} K(\mathbf{D}(\mathbf{P}_p \mathbf{x}_a), \mathbf{D}(\mathbf{P}_q \mathbf{x}_b)). \tag{2.8}$$

Here, $\mathbf{P}_p$ represents the p-th permutation matrix within the molecular permutational group, and S is the number of permutations for the molecule. By utilizing this sym-

metric kernel function, it ensures that the measured similarity remains consistent for the same molecule configuration with different permutations. This modification significantly enhances the accuracy when modeling molecules with diverse permutational symmetries, such as benzene, toluene, and paracetamol. However, the analytical solution of $\boldsymbol{V}$ involves the inversion of matrix $\mathbf{K}_{Hess}(\mathbf{X}, \mathbf{X}) + \lambda \mathbf{I}_{3MN}$, which has dimensions of $3MN \times 3MN$. The computational cost of this matrix inversion scales as $\mathcal{O}(M^3N^3)$, limiting its application to small systems and hindering predictions in systems with a moderately large number of atoms.

2.2.3 Intuition of AFF

Given the connections between KRR and GP models discussed in Section 1.1.4, we start with a GP model for any molecule configuration $\mathbf{x}$. Let's consider treating the energy $E(\mathbf{x}) \in \mathbb{R}$ as a noisy observation that follows a GP:

$$E(\mathbf{x}) \sim \mathrm{GP}\left(\mu_E(\mathbf{x}), \sigma^2(K(\mathbf{x}, \mathbf{x}') + \lambda)\right), \tag{2.9}$$

where $K(\cdot, \cdot)$ represents the correlation function, and λ is the nugget parameter. Due to the relation between the potential energy and the force vector in Equation (2.1), the distribution of the force vector of configuration $\mathbf{x}$ follows

$$\mathbf{F}(\mathbf{x}) = -\nabla_{\mathbf{x}}\mathbf{E}(\mathbf{x}) \sim \mathrm{GP}(-\nabla_{\mathbf{x}}\mu_E(\mathbf{x}), \sigma^2(\nabla_{\mathbf{x}}K(\mathbf{x}, \mathbf{x}')\nabla_{\mathbf{x}'}^T + \lambda \mathbf{I}_{3N})). \tag{2.10}$$

This reformulation relies on the linear transformations of GP discussed in Section 1.3. If a constant mean is employed in Equation (2.9), it leads to $-\nabla_{\mathbf{x}}\mu_E(\mathbf{x}) = \mathbf{0}$.

Denote $\mathbf{R}(\mathbf{x}_a, \mathbf{x}_b)$ as the correlation matrix between the forces of two molecules, labeled as a and b, with positions denoted as $\mathbf{x}_a$ and $\mathbf{x}_b$. The element at position (i, j)

follows:

$$(\mathbf{R}(\mathbf{x}_a, \mathbf{x}_b))_{ij} = \nabla_{\mathbf{r}^{ai}} K(\mathbf{D}(\mathbf{x}_a), \mathbf{D}(\mathbf{x}_b)) \nabla^T_{\mathbf{r}^{bj}}. \tag{2.11}$$

The correlation between the i-th atom of molecule a and the jth atom of molecule b can be calculated as follows:

$$\begin{aligned}
(\mathbf{R}(\mathbf{x}_a, \mathbf{x}_b))_{ij} &= \nabla_{\mathbf{r}_{ai}} K(\mathbf{D}(\mathbf{x}_a), \mathbf{D}(\mathbf{x}_b)) \nabla^T_{\mathbf{r}_{bj}} \\
&= \sum_{pq=11}^{NN} \sum_{mn=11}^{NN} \frac{\partial^2 K}{\partial \mathbf{D}_{pq} \partial \mathbf{D}_{mn}} \frac{\partial \mathbf{D}(\mathbf{x}_a)_{pq}}{\partial \mathbf{r}_{ai}} \frac{\partial \mathbf{D}(\mathbf{x}_b)_{mn}}{\partial \mathbf{r}_{bj}} \\
&= \begin{cases}
\frac{\partial^2 K}{\partial \mathbf{D}_{ij} \partial \mathbf{D}_{ij}} \frac{\partial \mathbf{D}(\mathbf{x}_a)_{ij}}{\partial \mathbf{r}_{ai}} \frac{\partial \mathbf{D}(\mathbf{x}_b)_{ij}}{\partial \mathbf{r}_{bj}} & \text{if } i > j, \\
\frac{\partial^2 K}{\partial \mathbf{D}_{ji} \partial \mathbf{D}_{ji}} \frac{\partial \mathbf{D}(\mathbf{x}_a)_{ji}}{\partial \mathbf{r}_{ai}} \frac{\partial \mathbf{D}(\mathbf{x}_b)_{ji}}{\partial \mathbf{r}_{bj}} & \text{if } i < j, \\
\sum\limits_{\text{p or q=i; m or n=i}} \frac{\partial^2 K}{\partial \mathbf{D}_{pq} \partial \mathbf{D}_{mn}} \frac{\partial \mathbf{D}(\mathbf{x}_a)_{pq}}{\partial \mathbf{r}_{ai}} \frac{\partial \mathbf{D}(\mathbf{x}_b)_{mn}}{\partial \mathbf{r}_{bj}} & \text{if } i = j,
\end{cases}
\end{aligned} \tag{2.12}$$

where $\frac{\partial^2 K}{\partial \mathbf{D}_{pq} \partial \mathbf{D}_{mn}}$ is the simplified notation of $\frac{\partial^2 K(\mathbf{D}(\mathbf{x}_a), \mathbf{D}(\mathbf{x}_b))}{\partial D(\mathbf{x}_a)_{pq} \partial \mathbf{D}(\mathbf{x}_b)_{mn}}$.

Based on the equation (2.12), when $i = j$, the number of terms in the summation is much larger than the number of terms when $i \neq j$. This implies that the absolute correlation between different atoms is typically smaller than the correlation between identical atoms. This empirical observation usually holds true for common kernel functions such as the Gaussian kernel and Matérn kernel before enforcing the constraint of permutational symmetry.

It's worth noting that, after applying the permutational symmetric kernel function from Equation (2.8), the absolute correlation between some parts of atoms is typically smaller than the correlation between the other set of atoms. This is a consequence of the permutation operation of $\mathbf{R}(\mathbf{x}_a, \mathbf{x}_b))_{ij}$ for permutable atoms. This observation is empirically confirmed in simulated datasets.

For instance, in part (a) of Figure 2.3, which pertains to benzene, after adopting the

permutational symmetric kernel function, the correlation between "same type" atoms is substantial due to permutational symmetries, whereas the correlation of forces between atoms in different atom sets remains low. In section 2.3.1, we will provide a detailed explanation of what constitutes a "same type" set of atoms.

This intriguing phenomenon suggests a natural sparsity in the correlation matrix, which we can potentially leverage to further reduce computational complexity when emulating the force fields of molecules.

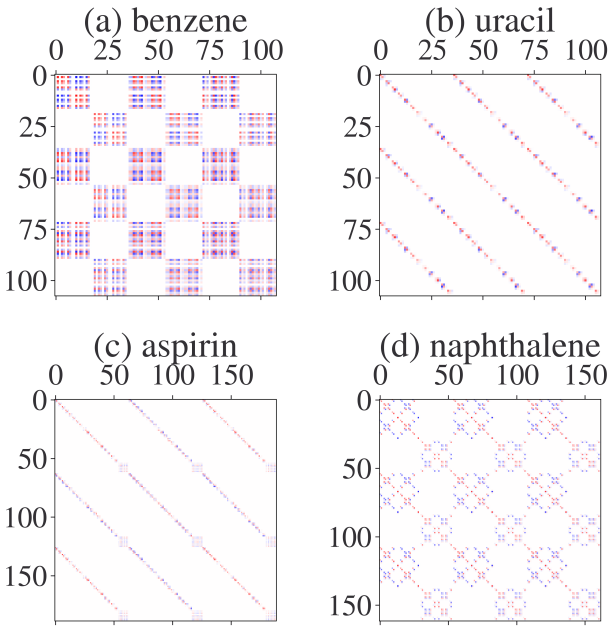

Figure 2.3: Covariance structure between atomic forces. Displayed are permutationally symmetric covariance matrices of atomic forces on three simulated configurations for (a) benzene, (b) uracil, (c) aspirin, and (d) naphthalene. Lighter colors indicate smaller absolute covariance values, and they indicate most elements in these matrices are near zero.

2.3 Atomic Force Field Method

2.3.1 Permutationally Equivalent Set

Because of the existence of different permutation orders of the atoms for the same molecule, one molecule might have several relevant physical permutation symmetries, leading to the same potential energy surface and force field [3, 72]. To follow this idea, we first define a group of atoms to be permutationally equivalent if they are interchangeable through any permutational operation. It is worth noting that the permutational symmetry here does not consider the reflection symmetry. In other words, atoms are not considered as the same PE set because they may have very different local chemical environments, and thus different forces even if they are plane symmetry. We call atoms from different PE sets permutationally distinct atoms. The AFF approach predicts the force of an atom in a molecule based on the force from its PE group of atoms rather than all atoms in this molecule. Figure 2.3 indicates that we may not need to include all atoms in a large covariance matrix for predicting atomic force to achieve computational efficiency in emulation, as many elements in the Hessian kernel matrix are near-zero.

For example, all four hydrogen atoms in methane (CH_4) form one set of PE atoms, while the carbon atom itself is another PE set, as the coordinates of all hydrogen atoms are interchangeable among all permutation symmetries. Benzene (C_6H_6), as another example, is comprised of just two sets of PE atoms–the first PE set containing the six carbon atoms and the second PE set containing the six hydrogen atoms. By contrast, all twelve atoms in a uracil molecule ($C_4H_4N_2O_2$) are permutationally distinctive, due to the unique atomic environments of each atom, which leads to twelve PE atom sets.

The PE sets of atoms can be found by minimizing the loss function through the

permutation matrix $\mathbf{P}^*$ [61, 73]

$$\mathbf{P}^* = \arg\min_{\mathbf{P}} \|\mathbf{P}\mathbf{A}_\mathbf{H}\mathbf{P}^T - \mathbf{A}_\mathbf{G}\|, \tag{2.13}$$

where $\mathbf{A}_H$ and $\mathbf{A}_G$ are adjacency matrices of two isomorphic molecules, and $\mathbf{A}_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$. By analyzing the index location from the permutation matrices of all permutation symmetries on the same type of molecule, we can partition the atoms from the same molecule into sets of PE atoms $S^i = \{\mathbf{r}_1^i, \cdots, \mathbf{r}_{l_i}^i\}$, where $\mathbf{r}_1^i, \cdots, \mathbf{r}_{l_i}^i$ are atoms belonging to the i_{th} atom set, for $i = 1, ..., L$.

As shown in Figure 2.3, the absolute correlation between the atoms in a PE set in Equation (2.8) is much larger than zero. We found that using the PE atoms significantly improves the predictive accuracy of atomic forces, compared with the approach that groups each atom as one set. This result is sensible as the forces of PE atoms are similar, and the correlation of forces from the PS kernel between PE atoms can capture the similarity. In contrast, the conventional Hessian kernel does not encode the permutational symmetries into the model. Note that the correlation of atomic forces from different PE sets is close to zero. This feature allows us to model atomic forces in each PE set separately, which substantially reduces the computational complexity.

2.3.2 Atomized Force Field Model

Consider a molecule that has N atoms grouped into L PE atom sets, each set containing l_i atoms, for $i = 1, ..., L$, we decompose the large covariance matrix to construct predictive models for each PE atom set in parallel, as illustrated in Figure 2.4. Let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_M\}$ be M configurations of this molecule that have been simulated from AIMD, and let $\mathbf{x}_j^i$ be a $3 \times l_i$ matrix that contains the i_{th} PE set's atomic coordinates in $\mathbf{x}_j$. Denote the forces of the atoms of i_{th} PE set in M training configurations by a

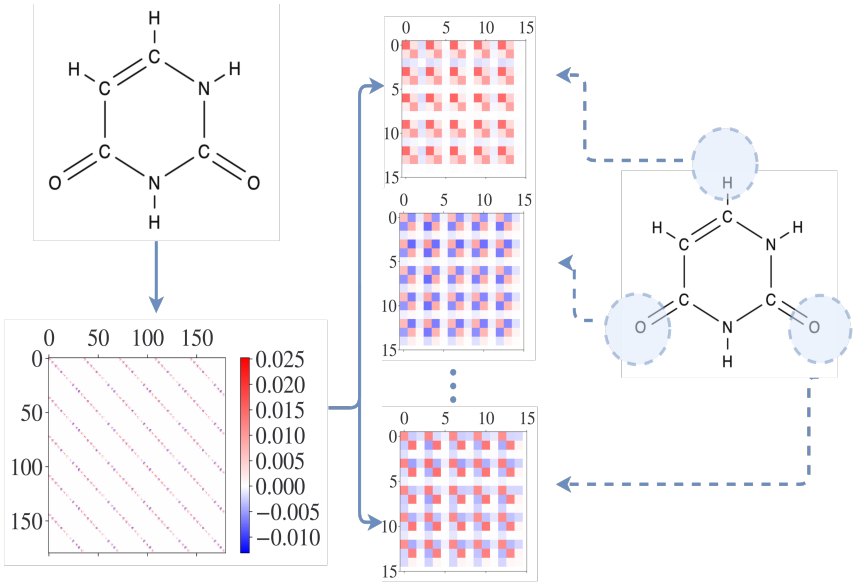

Figure 2.4: AFF force field on Uracil. This figure shows covariance matrices of atoms in the AFF method on uracil, a molecule for which each atom is its own PE set. Here there is a large correlation between atomic force on the same atom at different configurations, but a very small correlation between different atoms at the same or different configurations. The rightmost part of the figure shows the sub-covariance matrix of the atomic force of each atom in uracil across five simulated configurations.

$3Ml_i$ vector $\mathbf{F}_i$. For a new molecular configuration $\mathbf{x}^*$, the KRR estimator minimizes the loss function that penalizes both squared error fitting loss, and the complexity of the latent function simultaneous [42], leading to a weighted average of the force vectors at M training configurations:

$$\hat{\mathbf{F}}_i(\mathbf{D}(\mathbf{x}^*)) = \boldsymbol{\omega}_i^* \mathbf{F}_i, \tag{2.14}$$

where the weights follow $\boldsymbol{\omega}_i^* = \mathbf{R}_{\mathbf{x}^*}^T(\mathbf{R} + \lambda \mathbf{I}_{3Ml_i})^{-1}$ with $\mathbf{I}_{3Ml_i}$ being an identity matrix of size $3Ml_i \times 3Ml_i$. Here $\mathbf{R}$ is a $3Ml_i \times 3Ml_i$ covariance matrix with the (j,k)-th $3l_i \times 3l_i$ block term being the Hessian matrix of the kernel function $\nabla_{\mathbf{x}_j^i} K(\mathbf{D}(\mathbf{x}_j), \mathbf{D}(\mathbf{x}_k)) \nabla_{\mathbf{x}_k^i}^T$; λ is an estimated regularization parameter; $\mathbf{R}_{\mathbf{x}^*}$ is a $3Ml_i \times 3l_i$ matrix, where the jth $3l_i \times 3l_i$ block term is $\nabla_{\mathbf{x}_j^i} K(\mathbf{D}(\mathbf{x}_j), \mathbf{D}(\mathbf{x}^*)) \nabla_{\mathbf{x}^{*i}}^T$.

In addition to a point prediction on an untested run, the GP regression can provide closed-form predictive intervals, which is useful for uncertainty assessment of predictions. Thus, we can construct a GP regression model of atomic forces separately for each PE atom set. Given any M training simulation runs, the marginal distribution of the force vector $\mathbf{F}_i$ follows a multivariate normal distribution:

$$\left(\mathbf{F}_i \mid \mathbf{R}, \sigma_i^2, \lambda\right) \sim \mathcal{MN}\left(\mathbf{0}, \sigma_i^2(\mathbf{R} + \lambda \mathbf{I}_{3Ml_i})\right), \tag{2.15}$$

for $i = 1, \cdots, L$, where σ_i^2 is a variance parameter for the i-th PE set, and λ is the nugget parameter shared across all PE sets. Here the variance parameter σ_i^2 can differ across different PE sets, as the scale of the force can vary significantly for atoms in each PE atom set. The range and nugget parameters are assumed to be the same, as the smoothness of the latent function that maps atoms' positions to forces is approximately the same across different atom sets. The computational complexity of the predictive mean in a GP emulator with the same kernel and nugget parameters across atom sets is much smaller than the GP emulator with different parameters [40].

The power exponential covariance and the Matérn covariance function are widely used as the covariance function in GP models [42]. The Matérn kernel function with roughness parameter 5/2 is used as the default covariance function of a few GP emulator packages [44, 74], as well as the GDML approach for energy-conserving force field emulation [4]. This is partly because the sample path of GP with this kernel is twice differentiable, leading to relatively accurate predictions for both rough and smooth response surfaces. Here we also use the Matérn kernel function with roughness parameter 5/2:

$$K(\mathbf{D}(\mathbf{x}_a), \mathbf{D}(\mathbf{x}_b)) = \left(1 + \sqrt{5}\frac{d}{\gamma} + \frac{5d^2}{3\gamma^2}\right) \exp\left(-\sqrt{5}\frac{d}{\gamma}\right), \tag{2.16}$$

where γ is the range parameter, and d is the Euclidean distance between $\mathbf{D}(\mathbf{x}_a)$ and $\mathbf{D}(\mathbf{x}_b)$. Similar to the adjustment of the kernel function used in sGDML [3], we transform the Matérn kernel to the PS kernel function in Equation (2.8) in the AFF emulator, to capture permutational symmetries between PE atoms. The range parameter from this PS isotropic kernel function is related to the descriptors, which is a high dimensional vector of the inverse pairwise distance of a molecule generated from the simulator. Conditional on γ and λ, the maximum likelihood estimator of σ_i^2 is $\hat{\sigma}_i^2 = S_i^2/(M3l_i)$ with $S_i^2 = \mathbf{F}_i^T(\mathbf{R} + \lambda \mathbf{I}_{3Ml_i})^{-1}\mathbf{F}_i$ for the i-th PE atom set. The nugget parameter λ and the range parameter γ can be estimated by numerically optimizing the profile likelihood or by cross-validation with respect to squared error loss in predictions. When the number of training configurations is small, the marginal posterior mode may be used to avoid unstable estimation of the range and nugget parameters [41].

Conditional on the estimated parameters $\hat{\boldsymbol{\theta}}_i = [\hat{\sigma}_i^2, \hat{\gamma}, \hat{\lambda}]$, the predictive distribution of the atomic forces in the i-th PE atom set $\mathbf{F}_i(\mathbf{D}(\mathbf{x}^*))$ at any configuration $\mathbf{x}^*$ follows a multivariate normal distribution

$$\left(\mathbf{F}_i(\mathbf{D}(\mathbf{x}^*)) \mid \mathbf{F}_i, \hat{\boldsymbol{\theta}}_i\right) \sim \mathcal{MN}(\hat{\mathbf{F}}_i(\mathbf{D}(\mathbf{x}^*)), \hat{\sigma}_i^2 \mathbf{K}_i^*(\mathbf{x}^*, \mathbf{x}^*)), \tag{2.17}$$

where the predictive mean vector and predictive covariance matrix follows

$$\hat{\mathbf{F}}_i(\mathbf{D}(\mathbf{x}^*)) = \mathbf{R}_{\mathbf{x}^*}^T(\mathbf{R} + \hat{\lambda}\mathbf{I}_{3Ml_i})^{-1}\mathbf{F}_i, \tag{2.18}$$

$$\hat{\sigma}_i^2 \mathbf{K}_i^*(\mathbf{x}^*, \mathbf{x}^*) = \hat{\sigma}_i^2(\mathbf{R}^* - \mathbf{R}_{\mathbf{x}^*}^T(\mathbf{R} + \hat{\lambda}\mathbf{I}_{3Ml_i})^{-1}\mathbf{R}_{\mathbf{x}^*}), \tag{2.19}$$

with $\mathbf{R}^*$ being a $3l_i \times 3l_i$ Hessian matrix of the kernel function $\nabla_{\mathbf{x}^{*i}} K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}^*)) \nabla_{\mathbf{x}^{*i}}^T$.

2.4 Potential Energy Prediction with the AFF

Emulating energy based on integrating both simulated force vector and energy can also induce high computational costs, due to computing the inversion of a large covariance matrix of simulated force vectors and energies [2]. Here we introduce a computationally feasible approach to emulate the potential energy. For any molecule with atomic configuration $\mathbf{x}^*$, the potential energy $E(\mathbf{x}^*)$ correlates with the vector of potential energy from previously simulated molecular configurations $\mathbf{E} = (E(\mathbf{x}_1), ..., E(\mathbf{x}_M))$, and the unobserved atomic force at this molecular configuration $\mathbf{F}(\mathbf{x}^*)$. Conditional on $\mathbf{E}$ and $\mathbf{F}(\mathbf{x}^*)$, the correlation between $E(\mathbf{x}^*)$ and forces at other configurations is small. Denote the forces of all atoms in M training configurations by a $3MN$ vector $\mathbf{F}$. Thus the predictive distribution of $E(\mathbf{x}^*)$ conditional on both simulated energy and atomic force $[E(\mathbf{x}^*) \mid \mathbf{E}, \mathbf{F}]$ can be approximated by $(E(\mathbf{x}^*) \mid \mathbf{E}, \mathbf{F}(\mathbf{x}^*))$, where $\mathbf{F}(\mathbf{x}^*)$ can be estimated by the predictive distribution in the AFF model discussed in Section 2.3.2. The motivation of the method is relevant to the inducing point approximation approach [59], where given outcomes of a function at a set of well-chosen induced pseudo-inputs, the predictive distribution of the outcome at a new input is assumed to be conditionally independent of outputs in the training dataset. Here, the inducing input points of $E(\mathbf{x}^*)$ are $[\mathbf{E}, \mathbf{F}(\mathbf{x}^*)]$, due to a large correlation between these variables. Conditional on $(\mathbf{E}, \mathbf{F}(\mathbf{x}^*))$, we assume the force vector $\mathbf{F}(\mathbf{x}^*)$ at this configuration is approximately independent of other training configurations of force vectors. This simplification avoids constructing and computing the large Hessian covariance matrix of force vectors, allowing us to perform inversion of a $(3N + M) \times (3N + M)$ covariance matrix, instead of inversion of a $(3N + 1)M \times (3N + 1)M$ covariance matrix, in energy prediction. When predicting energy in many new molecular settings, matrix inversion of the sub-covariance matrix for simulated energy is shared among all predictive distributions. Thus, they are

only needed to be computed once. Details of efficient computation for predicting energy are discussed in Appendix A.1.

For any molecule with atomic coordinates $\mathbf{x}^*$, the learning objective can be represented by a combined vector of force and energy $\mathbf{EF} = (E(\mathbf{x}^*), \mathbf{E}^T, \mathbf{F}^T(\mathbf{x}^*))^T$, where $E(\mathbf{x}^*)$ is the potential energy of the molecule with atomic coordinates $\mathbf{x}^*$, and $\mathbf{F}(\mathbf{x}^*)$ is the force field vector of this molecule. Assuming a GP model for potential energy with covariance function $K(\cdot,\cdot)$ and mean function $\mu(\cdot)$, the random vector $\mathbf{EF}$ follows a multivariate normal distribution:

$$\mathbf{EF} \sim \mathcal{MN}\left(\boldsymbol{\mu}_{EF}, \boldsymbol{\Sigma}_{EF}\right), \tag{2.20}$$

where the mean vector is given as follows

$$\boldsymbol{\mu}_{EF} = \left(\mu(\mathbf{x}^*), \boldsymbol{\mu}_{\mathbf{X}}^T, -\nabla_{\mathbf{r}}\mu(\mathbf{x}^*)^T\right)^T,$$

with $\mu(\mathbf{x}^*)$ assumed to be an unknown constant m (estimated by MLE in this work), and $\boldsymbol{\mu}_{\mathbf{X}} = m\mathbf{1}_M$. The covariance matrix in Equation (2.20) is given as follows

$$\boldsymbol{\Sigma}_{EF} = \sigma^2 \left(\begin{bmatrix} K_{\mathbf{x}^*,\mathbf{x}^*} & \mathbf{K}_{\mathbf{X},\mathbf{x}^*} & -\mathbf{J}_{\mathbf{x}^*,\mathbf{x}^*}^T \\ \mathbf{K}_{\mathbf{x}^*,\mathbf{X}} & \mathbf{K}_{\mathbf{X},\mathbf{X}} & -\mathbf{J}_{\mathbf{X},\mathbf{x}^*} \\ -\mathbf{J}_{\mathbf{x}^*,\mathbf{x}^*} & -\mathbf{J}_{\mathbf{x}^*,\mathbf{X}} & \mathbf{R}_{\mathbf{x}^*,\mathbf{x}^*} \end{bmatrix} + \lambda \mathbf{I}_{M+3N+1} \right),$$

where the upper left four matrix blocks are the covariance of energy vectors $(E(\mathbf{x}^*), \mathbf{E}^T)^T$. The (i,j) element of the correlation matrix $K_{\mathbf{X},\mathbf{X}}$ is $K(\mathbf{D}(\mathbf{x}_i), \mathbf{D}(\mathbf{x}_j))$, for $i = 1, ..., M$ and $j = 1, ..., M$, and $K_{\mathbf{x}^*,\mathbf{x}^*} = 1$. The vector $\mathbf{K}_{\mathbf{x}^*,\mathbf{X}}$ is defined as:

$$\mathbf{K}_{\mathbf{x}^*,\mathbf{X}} = \mathbf{K}_{\mathbf{x}^*,\mathbf{X}}^T = (K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}_1)), ..., K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}_M)))^T, \tag{2.21}$$

which represents the correlation between the potential energy at input $\mathbf{x}^*$ and the potential energy at training inputs $\mathbf{X}$. In addition, the $3N \times 3N$ correlation matrix between forces is denoted by $\mathbf{R}(\mathbf{x}^*, \mathbf{x}^*) = \nabla_{\mathbf{x}^*} K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}^*)) \nabla_{\mathbf{x}^*}^T$. Finally, $\mathbf{J}$ denotes the correlation between energy and forces. Here $\mathbf{J}_{\mathbf{x}^*,\mathbf{X}}$ is a $3N \times M$ matrix with the jth column being $\nabla_{\mathbf{x}^*} K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}_i))$, and $\mathbf{J}_{\mathbf{x}^*,\mathbf{x}^*}$ is the correlation matrix between force and energy for molecule configuration with atom positions $\mathbf{x}^*$.

Similar to the parameter estimation of AFF model discussed in Section 2.3.2, the mean and variance parameters can be estimated by the MLE of the simulated (training) energy vector below

$$\hat{m} = (\mathbf{1}_M^T (\mathbf{K}_{\mathbf{X},\mathbf{X}} + \lambda \mathbf{I}_M)^{-1} \mathbf{1}_M)^{-1} \mathbf{1}_M^T (\mathbf{K}_{\mathbf{X},\mathbf{X}} + \lambda \mathbf{I}_M)^{-1} \mathbf{E},$$
$$\hat{\sigma}^2 = \frac{(\mathbf{E} - \mathbf{1}\hat{m})^T (\mathbf{K}_{\mathbf{X},\mathbf{X}} + \lambda \mathbf{I}_M)^{-1} (\mathbf{E} - \mathbf{1}\hat{m})}{M}.$$

The range parameter γ and the nugget parameter λ in the kernel function can be estimated through numerical optimization by cross-validation or MLE.

Based on the previous discussion, assuming that the given $[\mathbf{E}, \mathbf{F}(\mathbf{x}^*)]$, $E(\mathbf{x}^*)$ is approximately independent of the rest of force vectors, then we have

$$\left(E(\mathbf{x}^*) \mid \mathbf{E}, \mathbf{F}, \hat{m}, \hat{\sigma}^2, \hat{\gamma}, \hat{\lambda} \right) \dot{\sim} \mathcal{N}\left(\hat{E}(\mathbf{x}^*), \hat{\sigma}^2 K_E^*(\mathbf{x}^*, \mathbf{x}^*) \right), \tag{2.22}$$

where $\dot{\sim}$ denotes the approximation of the predictive distribution, and the predictive mean is a weighted average of training energy $\mathbf{E}$ and training force $\mathbf{F}$:

$$\hat{E}(\mathbf{x}^*) = \boldsymbol{\omega}_E^* \mathbf{E} + \boldsymbol{\omega}_F^* \mathbf{F}. \tag{2.23}$$

Closed-form expressions of $\boldsymbol{\omega}_E^*$, $\boldsymbol{\omega}_F^*$ and $\mathbf{K}_E^*(\mathbf{x}^*, \mathbf{x}^*)$ are derived in Appendix A.2.

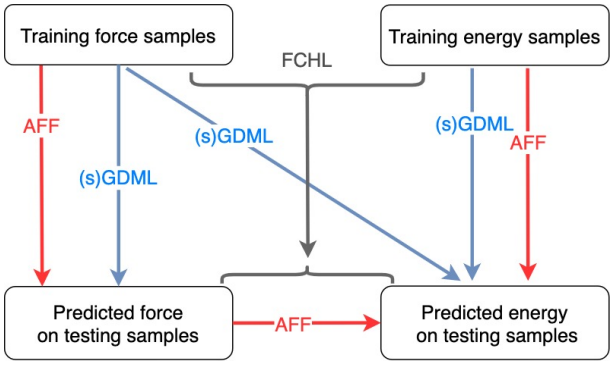

Figure 2.5: Schematic representation of different approaches in predicting atomic force and potential energy of molecules. GDML and sGDML methods predict the force of molecule at a new configuration based on forces on simulated configurations. The predictive force and energy were used to estimate the energy of this molecule. The FCHL method estimates the energy and atomic force by a joint model fitted using both the simulated force and energy samples. The AFF method partitions the atoms into PE atoms set and the atomic force of atoms of a new configuration is predicted based on the simulated force of atoms in the same PE set. The energy of the molecule at this configuration is predicted based on the predicted atomic force and energy from simulated samples.

In practice, the energy on the testing set is estimated in batches using the conditional distribution in Equation (2.22). The advantage of our method is that we exploit the estimable information from the force vector, but avoid computing the inverse of the gigantic kernel matrix on $\mathbf{F}$, which substantially simplifies the computation.

The comparison between the AFF and GDML models for predicting the molecular energy is illustrated in Figure 2.5. For GDML, as well as for both sGDML and FCHL, all simulated energy and force are used, but the inversion of a large covariance matrix is computationally expensive. Here, conditional on the simulated energy and force of a new molecular configuration, we assume the potential energy of a new molecule is independent of the forces of other molecular configurations simulated before. Since AFF does not need to handle the $3MN \times 3MN$ covariance matrix of simulated force vectors, it is more scalable for predicting the molecular level information of larger systems.

2.5 Numerical Results

We begin by comparing full Gaussian process regression and sparse Gaussian process regression for simple energy emulation using the global descriptor defined in sGDML and the AFF model. Subsequently, we evaluate the performance of the AFF approach by analyzing the required training time and learning curves on a variety of molecules, including benzene, uracil, and naphthalene from the MD17 dataset, and aspirin, alpha-glucose, and hexadecane from our simulated dataset (see Appendix A.3 for simulation details). We compare the predictive error and required training time from AFF with some of the most commonly used KRR-based models, such as the GDML and sGDML approaches for force and energy predictions. We also conducted a performance comparison between AFF and SOAP-GAP, which is based on the sparse approximation of covariance in a GP model. All comparisons are implemented under the same training and testing set. In addition, we provide the uncertainty assessment of predictions from our model through the proportion of held-out outcomes covered in the 95% predictive interval and the average length of the predictive interval (see Table 2.2 for details on the prediction accuracy, required training time, and uncertainty assessment of the AFF predictions). The ratio of the average length of the predictive interval to the range of testing forces L^{norm}, and the difference between the 95% confidence level and the proportion of the held-out samples contained in the predictive interval $\Delta p_{CI}^{0.95}$ on the held-out dataset are shown in Figure 2.9. An efficient method should have a small predictive error, small training cost, short average length of the predictive interval, and around 95% of the held-out test data covered by the 95% predictive interval.

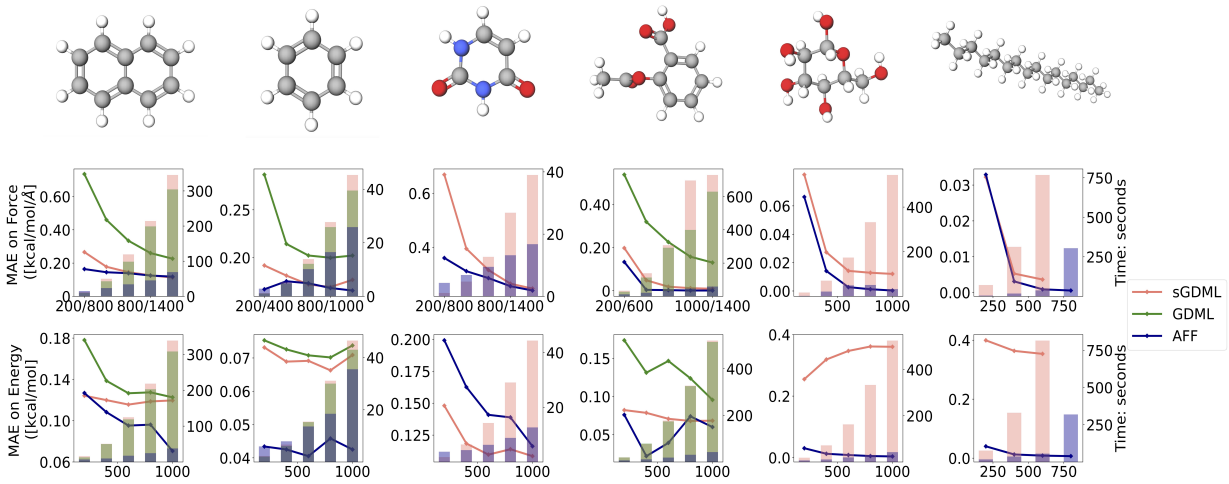

Figure 2.6: The learning curves for force and energy on naphthalene, benzene, and uracil from the MD17 dataset, and aspirin, alpha-glucose, and hexadecane from our simulated dataset (ordered from left to right). Learning curves are presented for the GDML and sGDML methods as well as the AFF method under the same training, validation, and testing set. The training sample size (x-axis) is tweaked for the AFF method on force prediction. The AFF uses a larger training set for predicting forces for the first four molecules, and the computational time(shown as the blue bars) is still much lower compared with the GDML and sGDML approaches. The top row contains the learning curve [in terms of mean absolute error (MAE)] and training time for the out-of-sample force prediction. The bottom row contains the learning curve and training time for the out-of-sample energy prediction. GDML and sGDML approaches are equivalent on uracil, alpha-glucose, and hexadecane, as all atoms are permutationally distinct. Thus only two learning curves are shown for those molecules.

2.5.1 Comparison between Full GPR and Sparse GPR

Constructing a sparse model on Gaussian process regression [11, 75, 76] is a common approach to address the computational cost associated with full Gaussian process models. Unlike the local descriptors used in the GAP model, both the GDML and our AFF method utilize global descriptors as inputs. To highlight the practical significance of the AFF model in reducing complexity through the natural sparsity of the correlation matrix, we will initially compare the performance of energy emulation between the full Gaussian process regression model using only energy information and the sparse GPR (SGP) model employing a similar sparse representation as used in the GAP approach, all under the same settings.

In this section, both the full GPR and SGP models employ the same descriptor $\mathbf{D}(\mathbf{x})$, as demonstrated in the AFF model, and utilize the same Matérn kernel function with a roughness parameter of 5/2 to define the kernel function. In the full GPR setting, the energy $\mathbf{E}$ is assumed to follow a multivariate normal distribution, given by:

$$\mathbf{E} \mid \mathbf{X}, \theta, \sigma^2, \gamma, \lambda \sim \mathcal{MN}(\boldsymbol{\mu}(\mathbf{X}), \sigma^2(\mathbf{R} + \lambda \mathbf{I}_M)), \tag{2.24}$$

where $\mathbf{R}$ is the correlation matrix among the data with $(i,j)_{th}$ element $\mathbf{R}_{ij} = K(\mathbf{D}(\mathbf{x}_i), \mathbf{D}(\mathbf{x}_j))$, λ is the nugget parameter of the random noise in $\mathbf{E}$, and $\boldsymbol{\mu}(\mathbf{X}) = [\theta, \cdots, \theta]^T$ with θ be the mean parameter. Conditional on γ and λ, the maximum likelihood estimator of θ is $\hat{\theta} = (\mathbf{H}^T(\mathbf{R} + \lambda \mathbf{I}_M)^{-1}\mathbf{H})^{-1}\mathbf{H}^T(\mathbf{R} + \lambda \mathbf{I}_M)^{-1}\mathbf{E}$, where $\mathbf{H} = [1, \cdots, 1]^T$, and the maximum likelihood estimator of σ^2 is $\hat{\sigma}^2 = S^2/M$ with $S^2 = (\mathbf{E} - \mathbf{H}\hat{\theta})^T(\mathbf{R} + \lambda \mathbf{I}_M)^{-1}(\mathbf{E} - \mathbf{H}\hat{\theta})$. The $\hat{\gamma}$ and $\hat{\lambda}$ are estimated numerically following the parameter estimation procedure in Section 1.1.1.

Conditional on the estimated parameter $\hat{\theta}$, $\hat{\sigma}^2$ $\hat{\gamma}$ and $\hat{\lambda}$, the predictive distribution E^*

of an unknown test configuration $\mathbf{x}^*$ for the potential energy follows

$$E^* \mid \mathbf{x}^*, \mathbf{E}, \hat{\theta}, \hat{\sigma}^2, \hat{\gamma}, \hat{\lambda} \sim \mathcal{N}(\mu^*, \hat{\sigma}^2 K^*), \tag{2.25}$$

$$\mu^* = \hat{\theta} + \mathbf{R}_{*M}(\mathbf{R} + \hat{\lambda}\boldsymbol{I}_M)^{-1}\mathbf{E}, \tag{2.26}$$

$$K^* = K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}^*)) - \mathbf{R}_{*M}(\mathbf{R} + \hat{\lambda}\boldsymbol{I}_M)^{-1}\mathbf{R}_{*M}^T, \tag{2.27}$$

where $\mathbf{R}_{*M}$ is the $1 \times M$ correlation matrix for $\mathbf{x}^*$ and data $\mathbf{X}$ with i-th element be $K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}_j))$. The Equation (3.14) is the defining equation for the full Gaussian process regression.

In the sparse Gaussian process emulation, we denote M^* as the number of representative points, and $\mathbf{X}^r$ represents the set of representative samples. The selection of the representative set is carried out using the leverage-score CUR [77] method, which is similar to the approach used for selecting representative environments in GAP [6]. Using the low-rank approximation of correlation matrices

$$\mathbf{R} \approx \mathbf{R}_{MM^*}\mathbf{R}_{M^*M^*}^{-1}\mathbf{R}_{MM^*}^T, \tag{2.28}$$

$$\mathbf{R}_{*M^*} \approx \mathbf{R}_{*M^*}\mathbf{R}_{M^*M^*}^{-1}\mathbf{R}_{MM^*}^T, \tag{2.29}$$

where $\mathbf{R}_{MM^*}$ is cross correlation matrix between $\mathbf{X}^r$ and $\mathbf{X}$.

Using the Woodbury matrix identity, the predictive mean of this SGP is given by

$$\mu^* \approx \hat{\theta} + \mathbf{R}_{*M^*}(\hat{\lambda}\mathbf{R}_{M^*M^*} + \mathbf{R}_{MM^*}^T\mathbf{R}_{MM^*})^{-1}\mathbf{R}_{MM^*}^T(\mathbf{E} - \hat{\theta}\mathbf{H}), \tag{2.30}$$

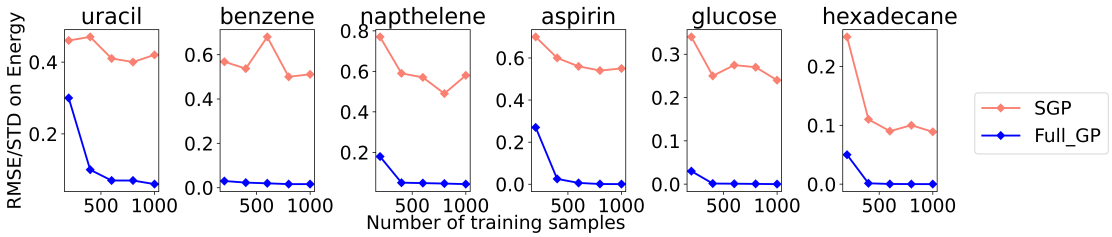

Figure 2.7: The learning curves (RMSE/STD) of Full GP (blue) and SGP (red) for energy on naphthalene, benzene, uracil from the MD17 dataset, and aspirin, alpha-glucose, hexadecane from our simulated dataset (ordered from left to right). The descriptor and kernel function are the same between Full GP and SGP methods. The number of representative samples used in SGP equals 200.

.

The comparison between full GP and SGP is presented in Figure 2.7. When using a global descriptor to encode molecule configurations, the figure suggests that the sparse GP based on our global descriptor doesn't perform as well in terms of accuracy compared to the simple single-energy-based GP model. This is because it's challenging to find an ideal representative set that accurately represents the entire training set using the global descriptor.

2.5.2 Comparison with GDML and sGDML Methods

Previous studies have shown that the GDML and sGDML have relatively small errors, compared with other approaches [2, 3]. Indeed, according to Figure 2.6, the predictive error is relatively small for both approaches. However, both GDML and sGDML have a large computational cost, mainly due to the inversion of $3NM \times 3NM$ covariance matrix of force vectors at all training configurations. Because of the reduced computational order on force prediction by partitioning the atoms into PE atom sets, the AFF model has a smaller predictive error of force prediction (blue curves) when using similar or even less training time (blue histograms) compared to GDML and sGDML approaches. The improved accuracy of force predictions by the AFF model is even more noticeable in the

additional simulation of aspirin, alpha-glucose, and hexadecane as shown in Figure 2.6. For some small molecules in the MD17 dataset, the AFF method could achieve better accuracy with less computational cost by using more training samples. For the larger molecules such as glucose and hexadecane, the proposed AFF method shows no loss of accuracy compared to the sGDML predictions at the same number of training samples.

The sGDML approach typically has a smaller predictive error compared with GDML for molecules with at least two PE atom sets, such as benzene, aspirin, and naphthalene molecules, consistent with the result reported in the previous study [3]. This is because sGDML approach encodes the PS kernel to properly represent the large correlation of forces between atoms in the PE atom set. Note that here the reduced computational cost in AFF allows us to train our models with more observations than the GDML and sGDML approaches for predicting the force with an even smaller computational budget. The number of training observations required in training the AFF model, however, is still very small (from a few hundred to a thousand).

In comparison, neural network (NN) approaches typically need a larger set of training observations (ranging from 10^4 to 10^5) to achieve similar or better predictive performance [7, 8, 78, 79]. Several recent NN methods are worth exploring [80, 81], as they seem to require fewer samples than conventional NN approaches. On the other hand, only two parameters (range and nugget parameters) in the GP model need to be numerically optimized, whereas a large number of parameters may need to be numerically optimized in NN approaches.

Given the same number of observations, the error in predicting the potential energy by the AFF model is typically smaller than the sGDML and GDML approaches, as shown in the second rows of Figure 2.6. For some molecules, such as naphthalene and benzene in the MD17 data set, and alpha-glucose and hexadecane in our simulated data set, the AFF model has much smaller predictive error than sGDML approach. This is

because our approach incorporates both force and energy vectors in energy prediction when making predictions on energy. Applying the sGDML with the hybrid loss function, the predictive error of it is about the same as the AFF model for alpha-glucose and hexadecane. Jointly modeling force and energy was recently studied in [2], which could have a large computational cost. Here the approximated approach introduced in Section 2.4 allows us to keep the computational complexity of predicting the energy the same as predicting the force, while maintaining relatively high predictive accuracy as that in [2]. For larger molecules, such as alpha-glucose, aspirin, and hexadecane (with 21 - 51 atoms) in our simulated dataset, the computation reduction is huge (see the blue histograms in Figure 2.6). For these examples, AFF achieves higher accuracy in predicting atomic force, despite costing less than 10% of the training time of the sGDML approach, as given in Table 2.2. Furthermore, the sGDML method requires a larger memory size to store the covariance of the simulated force vectors. Since we only need to store the covariance of force vectors in each PE atom set, the memory requirement is often much smaller.

Among all the molecules we compared, the AFF model has a larger predictive error for the uracil, using the same number of training inputs (third panel in the second row in Figure 2.6). Since the AFF model estimates the energy based on the emulated atomic force, the accuracy of energy prediction would be reduced when the emulated force is not accurate. As shown in Figure 2.6, the estimated force by AFF for uracil is not accurate when the number of the training sample is small. This problem can be solved by using a moderately large sample size (≈ 1000) to achieve similarly accurate predictions as the sGDML model. The predictive energy vector by the AFF model along the AIMD trajectories is graphed in Figure 2.8 along with the held-out energy in the simulation. Also plotted are the predictive atomic forces and the truth at two held-out configurations. Based on $M = 800$ simulated forces and energies, predictions of potential energies and forces by the AFF model are accurate.

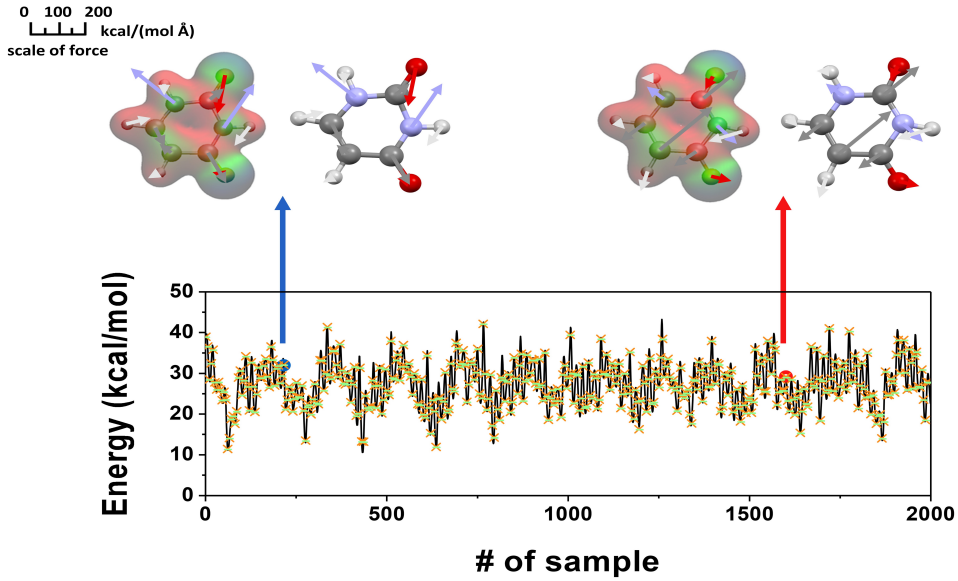

Figure 2.8: Energy of uracil obtained from the AFF method along the AIMD trajectories, where the AIMD energies are displayed in the black curve and predictions of held-out energies from the AFF model are graphed in yellow crosses. The green bars are the 95% predictive intervals of the AFF energies. As the length of the intervals is very small, the intervals almost overlap with the AIMD energies and they are nearly invisible. Depicted above the panel of energy in the upper half of the figure is a comparison of AIMD atomic forces and predicted atomic forces by the AFF model on two randomly selected molecules. Within each of the two pairs shown, the same molecule is illustrated twice with the depiction on the left displaying AIMD atomic forces and the depiction on the right displaying the atomic forces by the AFF model. $M = 800$ simulated forces and energies were used to train the AFF model for predictions.

2.5.3 Comparison with Gaussian Approximation Potential Method

The Gaussian approximation potential (GAP) [6] framework is another widely used method to reproduce the potential energy surface, based on sparse approximation of the covariance in a GP model. In GAP, the total energy of a system is decomposed into atomic energies and then generates the ML-based interatomic potentials using local atomic neighborhood descriptors such as SOAP vector [5], instead of global descriptors used in GDML and AFF methods. We are also interested in comparing the performance of the AFF with a naturally sparse covariance matrix to that of GAP, which relies on a sparse approximation of the covariance matrix.

Following the steps from [77], we use the leverage-score CUR for the sparse set selection. Table 2.1 compares the accuracy of predicted energy between AFF and SOAP-GAP (using QUIP code [82]) on three molecules from the MD17 dataset. The predictive error of the AFF method is around one order of magnitude smaller than the SOAP GAP method for all scenarios.

Compared with the SOAP-GAP method, the AFF method does not enforce additional sparse approximation, as the sparsity comes naturally from the kernel function of atomic forces that satisfies energy conservation and permutation symmetry of atoms, leading to a faster learning rate given a small set of observations

Unlike the local descriptor used in the Gaussian Approximation Potential model, the AFF method uses a global descriptor to characterize correlation across all atoms. For emulating a system of particle interactions with a large number of particles, approximation based on local information would be needed to reduce the computational speed. Though a few recent works have shown local descriptors are efficient for emulating interactions between a large number of small molecules (e.g. water molecule) [7, 8, 58], emulating interactions for larger molecules shown here are harder, due to complex molecule structures and chemical bonds. It is interesting to extend the AFF approach to systems of interactions of moderately large molecules. Additional modeling steps and approximation steps would be needed to accelerate the AFF approach for emulating systems with a large number of particles.

2.5.4 Numerical Performance of AFF

Table 2.2 gives the predictive error of force vectors and energy, the percentage of forces covered in the predictive interval, the average length of the predictive intervals of forces and computational costs in emulation from different methods. First, the predictive

RMSE/STD Performance				
Molecule	GAP Energy	AFF Energy	GAP Force	AFF force
Naphthalene	0.54	0.024	0.56	0.022
Benzene	0.47	0.043	0.49	0.015
Uracil	0.326	0.05	0.365	0.05

Table 2.1: The second to last columns show the root mean square error (RMSE) over the standard deviation of test samples on estimated energy and force. We use 2000 samples to train the SOAP-GAP method and the size of the induced-point set is 100. As a comparison, the AFF model is trained by only 200 samples.

Performance of AFF					
Molecule	Energy [kcal/mol]	Force [kcal/mol/Å]	Training Time [s]	P_{CI} (95%)	L_{CI} (95%)
Naphthalene	0.07 (0.12)	0.11 (0.11)	68 (345)	98.5%	1.36
Benzene	0.04 (0.07)	0.173 (0.176)	23 (45)	85%	0.7
Uracil	0.10 (0.10)	0.239 (0.249)	16 (43)	97.8%	2.37
Alpha-glucose	0.09 (0.36)	0.0003 (0.012)	32 (543)	100%	0.03
Hexadecane	0.008 (0.35)	0.0008 (0.003)	37 (767)	99%	0.05
Aspirin	0.06 (0.09)	0.0028 (0.009)	32 (629)	99.8%	0.08

Table 2.2: The second and third columns show the MAE on estimated energy and force. The fourth column is the training time of the model at shown force accuracy, which is provided in seconds. The numbers in parentheses are the sGDML results, and they are tested under the same held-out test set. The specific criteria employed are the following: $P_{CI}(95\%) = \frac{1}{3NM^*}\sum_{i=1}^{M^*}\sum_{j=1}^{3N} 1_{\{}\mathbf{F}(\mathbf{x}_i^*)_j\} \in CI_{ij}(95\%)$, $L_{CI}(95\%) = \frac{1}{3NM^*}\sum_{i=1}^{M^*}\sum_{j=1}^{3N} \text{length}\{CI_{ij}(95\%)\}$, where M^* is the number of test samples, $\mathbf{F}(\mathbf{x}_i^*)_j$ is the j_{th} element from the force vector prediction of the output of the i_{th} held-out molecule; $CI_{ij}(95\%)$ is the 95% predictive credible interval from the multivariate normal distribution in (2.17); and length$\{CI_{ij}(95\%)\}$ is the length of the 95% predictive credible interval. The number of training samples used in the AFF and sGDML (in parentheses) method is naphthalene 1600 (1000), benzene 1200 (1000), uracil 1600 (1000), alpha-glucose 1000 (1000), aspirin 1200 (1000), and hexadecane 600 (600).

error of AFF methods for both forces and energy is typically not larger than the sGDML approach. For some molecules such as alpha-glucose and hexadecane, the predictive error of the AFF model seems to be one order of magnitude smaller, based on the same held-out test set. It is worth noting that it takes the AFF model less computational costs (ranging from 1/2 to 1/30 of costs compared to sGDML) to achieve a similar or

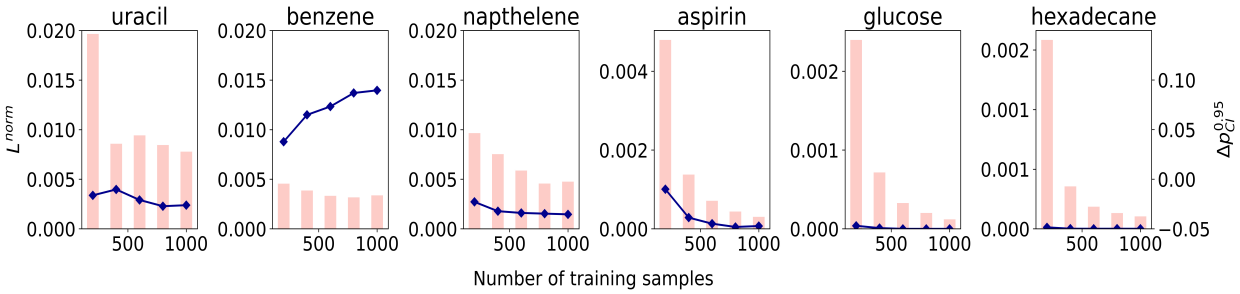

Figure 2.9: The bar charts show the proportion of average AFF predicted atomic force confidence interval $L_{CI}(95\%)$ over the range from testing samples changing with the number of training samples. The blue line charts show the difference Δp between confidence level 0.95 and actual coverage of AFF predicted atomic force confidence level. The proportion L^{norm} is given by $\rho = L_{CI}(95\%)/\text{range}(\mathbf{F}(\mathbf{x}^*))$, where $L_{CI}(95\%) = \frac{1}{3NM^*}\sum_{i=1}^{M^*}\sum_{j=1}^{3N} length\{CI_{ij}(95\%)$, and $\text{range}(\mathbf{F}(\mathbf{x}^*)) = \max(\mathbf{F}(\mathbf{x}^*)) - \min(\mathbf{F}(\mathbf{x}^*))$. The $\Delta p_{CI}^{0.95}$ is given by $\Delta p_{CI}^{0.95} = 0.95 - P_{CI}(95\%)$, where $P_{CI}(95\%) = \frac{1}{3NM^*}\sum_{i=1}^{M^*}\sum_{j=1}^{3N} 1\{\mathbf{F}(\mathbf{x}_i^*)_j\} \in CI_{ij}(95\%)$.

higher level of predictive accuracy. These results indicate the AFF model is more efficient in emulating atomic forces and energy in AIMD simulation. Furthermore, around 95% (or higher percentage) of the held-out atomic forces are covered by relatively short 95% predictive intervals from the AFF approach, indicating that the AFF model provides a reliable way to quantify the uncertainty in predictions.

Furthermore, it is worth mentioning that the reduction of computational cost by the AFF model is more pronounced on molecules with more PE atoms' sets, such as alpha-glucose, aspirin, and hexadecane. For molecules with fewer PE sets such as benzene (where we can only partition the atoms into two PE sets for each configuration), the computational reduction will be smaller. Thus, our approach may be useful for reducing the computational cost of interactions between a large number of molecules, as most PE sets may only contain one atom.

Finally, uncertainty assessments of predictions of the AFF approach are shown in Figure 2.9. Compared with the range of observations, the average length of 95% (pink

bar) is much shorter, indicating a small uncertainty associated with predictions. The difference between the number of held-out test samples covering the 95% interval and the nominal 95% range (blue curves) is small, meaning that the uncertainty is accurately quantified. The internal assessment of the uncertainty of the AFF model can be used to identify the input region with large uncertainty, and sequentially design simulation runs for uncertainty reduction or Bayesian optimization [83, 84]. With a fast and accurate emulator of the force vectors of the particles, one can also utilize experimental data from optical or electronic microscopy to inversely estimate dynamical properties of the system [85, 86], and to identify new features by statistical tests to improve simulation models [87].

Chapter 3

High Dimensional Optimization on Inverse Force-Fields Design

A fundamental goal of materials design is to optimize specific materials properties while accounting for practical constraints rooted in chemistry and physics, enabling the synthesis of new materials. Due to the high dimensionality involved in molecular geometry including the atom positions, types of atoms, and chemical bonds, it is challenging to optimize the entire input space. Even if optimizing the positions of N atoms and exploring variations from a fixed initial structure within a pre-specified number of grids c, the number of unexplored configurations c^N becomes astronomically large with the increase of c and N.

Many years ago, researchers embarked on a quest to build consistent and predictive quantum-chemical models aimed at uncovering valuable structure-function relationships in various realms of materials chemistry [88]. The physics and chemistry of these molecules are governed by quantum mechanics, which can be solved via the Schrodinger equation. In general, the challenge of molecular design involves nonlinear optimization [89]. Alterations in molecular structure, and consequently, changes in the matrix elements

of the corresponding Hamiltonian, result in variations in wave functions, energy eigenvalues, and derived properties. The contributions from these quantities are inherently nonlinear.

In recent years, the rapid development of statistical and machine learning surrogate models provided an alternative way of learning the mapping from molecule structure to molecular properties, while maintaining a high level of accuracy and greatly reducing the complexity order compared to simulation methods such as AIMD. From the simulated set of samples, the surrogate models are able to accurately predict the molecular properties given molecule structure information. The surrogate models facilitate the exploration of the mapping from molecular properties to molecular configurations, thereby providing a set of configurations for molecules that attain the desired properties.

Facilitating the inverse mapping from potential energy to the molecular configuration is crucial in the field of computational chemistry such as energy minimization. However, the abundance of degrees of freedom on atomic position for describing potential energy surfaces makes the prediction of a molecular structure satisfying certain energy a challenging task. This chapter focuses on the inverse mapping problem, aiming to construct molecule structures that meet predefined potential energy conditions. This problem is relevant to finding the optimized structure with a local or global potential energy minimum. Instead of employing conventional optimization techniques that often require gradient evaluations or a large number of iterations, we explore the capacity to resolve the geometry optimization problem through a learned inverse mapping by leveraging the AFF emulator introduced in Chapter 2 and using a Bayesian optimization (BO) approach [84, 90].

3.1 Background and Literature Review

The potential energy function, denoted as $E(\mathbf{r}^N)$, serves as the foundational determinant of the thermodynamic and kinetic characteristics of a classical atomic system comprising N atoms. The input of this function has $3N$ dimensions for a system of N atoms. Energy optimization is of paramount importance because the lowest-energy structural arrangement of a molecule fundamentally governs a majority of its properties. As a result of energy minimization, a detailed molecular structure is obtained, including geometric information such as Cartesian coordinates.

Within the landscape of the potential energy function, various significant points exist, commonly referred to as stationary points including local minimizers and global minimize. Local minimizers represent configurations where the potential energy reaches a local minimum within their immediate surroundings, offering insights into stable states. Global minimizer, on the other hand, is particularly significant as they correspond to the lowest energy state across the entire landscape, signifying the most stable configuration. These points are characterized by the property that the gradient of the potential energy landscape is zero for each atom's coordinate:

$$\nabla_{\mathbf{r}_i} E(\mathbf{r}^N) = \mathbf{0}, \quad i = 1, 2, \ldots, N. \tag{3.1}$$

This condition implies that the atomic force vectors are all zero when the potential energy is minimized, in accordance with the law of energy conservation, and these configurations represent mechanically stable states of the atoms within the system.

In general, energy minimization commences from a non-equilibrium molecular geometry. It leverages mathematical optimization techniques, often employing numerical methods such as the Newton-Raphson method [91] and the conjugate gradient method

[92], to iteratively adjust the atomic positions. This iterative process reduces the forces acting on the atoms (corresponding to the gradients of potential energy) until these forces become negligibly small.

Quantum mechanics can be employed to precisely calculate the energy and force field of a molecule. Among these methods, the most accurate representation of a molecule's potential energy is achieved through AIMD. However, AIMD is not always feasible in practice because of the large associated computational cost. In Chapter 2, we introduced the AFF emulator, which demonstrates the remarkable capability to predict energy and force fields with accuracy comparable to AIMD but at a significantly reduced computational cost. This approach enables us to focus on optimizing the machine learning learned function, enhancing the minimization process to discover local or global minima more effectively.

In recent years, various methods have emerged to tackle the challenging problem of molecule design. One approach, as demonstrated in [9], utilizes a generative model to generate molecule conformation given the molecular graph learned from a large pair of different molecules, each paired with a reference conformation obtained by optimizing the molecular geometry with density function theory. In this approach, molecules are represented as an undirected, complete graph, which is a common representation in the context of message-passing neural networks [17]. By employing a conditional variational graph autoencoder to capture the conformation distribution given the molecular graph, one is able to generate a set of plausible conformations.

The G-SchNet model introduced in [10] is designed for the generation of 3D point sets with rotational invariance. It builds upon the foundation of SchNet, a forward deep learning-based model mapping from molecule structures to molecular properties as proposed in [7], which utilizes continuous-filter convolutional layers to model molecular interactions. Unlike some methods focusing on generating molecular graphs, G-SchNet

directly generates atomic types and positions. In another study [93], a generative adversarial network architecture is proposed to learn the distribution of Euclidean distance matrices and generate these matrices for molecular structures.

However, all those generative models rely on a large number of simulated structures, and none of them have yet been used to discover the inverse mapping from potential energy surface to 3D molecule structure. As discussed before, an accurate representation of a molecule's potential energy achieved through AIMD is a computation-consuming procedure. Mapping from molecule structure to potential energy surface or force fields can be considered an expensive, black-box function. Bayesian optimization [84, 90] is a popular method for finding the global minimum of expensive functions within a bounded set of parameters without the need to compute the gradient. It has been extensively used in optimizing time-consuming engineering simulations and fitting machine learning models on large datasets. For example, A BO framework named ChemBO was proposed in [94] to optimize the arrangement of atoms and bonds in organic molecules for desired molecular properties. In [95], the authors demonstrated that by reformulating the search procedure as a constrained Bayesian optimization problem, the validity of generated molecules from a variational autoencoder is significantly improved. Additionally, in [96], Bayesian optimization was shown to be significantly more efficient than random search and grid search in solving large-scale hyperparameter search problems for machine learning methods such as KRR in computational chemistry.

The key element of BO is its predictive component, which relies on Gaussian process regression. It is natural to draw a connection between the inverse mapping problem of potential energy surface and force fields with our AFF method.

Estimating the inverse mapping from molecular properties to molecular structure presents two primary difficulties for Bayesian optimization. First, Bayesian optimization is typically applied to optimizing functions with input dimensions that are not exces-

sively large (usually less than 20). In contrast, the input for molecular PES and force field simulations involves the molecular structure, which encompasses the positions of all atoms, resulting in a $3N$ dimensional space, where N represents the system size. Second, our model training data consists of a limited number of samples, and the configurations of sampled molecules only cover a small portion of the vast PES landscape. This means that a successful method for inverse mapping from PES and force fields must possess the ability to explore the broader, unsampled regions within the constraints of physical validity. In essence, the molecules suggested by this method should, at a minimum, be chemically valid and able to be simulated by the underlying physics-based simulator.

Our GP-based AFF emulator demonstrates the capability to construct the forward PES and force-field mapping using a relatively small set of samples while providing valid uncertainty quantification. In the subsequent sections, we will commence by introducing BO. Following that, we will illustrate how we leverage the AFF method and BO to address the inverse mapping of the PES and force fields in the high-dimensional molecule structure space. Through this inverse mapping process, we will demonstrate how it enhances the energy minimization process.

3.2 Bayesian Optimization

The Bayesian optimization is an approach for optimizing objective functions that are resource-intensive to evaluate:

$$\min_{\mathbf{x}\in\mathcal{A}} f(\mathbf{x}), \tag{3.2}$$

where $f(\mathbf{x})$ is a function that lacks ideal properties such as convexity or concavity, and $\mathcal{A}$ defines the design space of interest. In global optimization, $\mathcal{A}$ is often a compact subset of $\mathbb{R}^d$, typically assumed as a hyper-rectangle $\mathbf{x} \in \mathbb{R}^d : a_i \leq x_i \leq b_i$, or locates within

a d-dimensional simplex, denoted as $\mathbf{x} \in \mathbb{R}^d : \sum_i x_i = 1$. For some real applications, the input space $\mathcal{A}$ is not a hyper-rectangle, where obtaining an efficient initial design may be achieved by simply first obtaining a 'space-filling' design, such as the Latin-hypercube design [22] in a hyper-rectangle that contains the $\mathcal{A}$ and then only use the sample within the input space $\mathcal{A}$. The observed or output value y from $f(\mathbf{x})$ may contain noise or noise-free. The typical structure of the BO algorithm consists of two main components: a surrogate model with internal uncertainty assessment, typically Gaussian process regression, for modeling the objective function, and an acquisition function that guides the selection of the next evaluation location.

Suppose we have an initial dataset of M observation pairs $(\mathbf{x}_i, y(\mathbf{x}_i))$. For simplicity, let's assume these observations are noise-free. Following the GPR framework outlined in the introduction, we treat the vector $(y(\mathbf{x}_1), \cdots, y(\mathbf{x}_M))$ as a random draw from a multivariate normal distribution, as shown in Equation (1.1). Given this initial dataset and a selected correlation function, our next step is to infer the value of $y(\mathbf{x}^*)$ at a new input $\mathbf{x}^*$. By constructing the joint distribution of $(y(\mathbf{x}_1), \cdots, y(\mathbf{x}_M))^T$ and $y(\mathbf{x}^*)$, we can derive the posterior probability distribution

$$y(\mathbf{x}^*)|y(\mathbf{x}_1), \cdots, y(\mathbf{x}_M) \sim \mathcal{N}(\mu^*(\mathbf{x}^*), \sigma^2 K^*(\mathbf{x}^*, \mathbf{x}^*)), \tag{3.3}$$

where $\mu^*(\mathbf{x}^*)$ represents the posterior mean function, and $\sigma^2 K^*(\mathbf{x}^*, \mathbf{x}^*)$ is the posterior variance. The parameters associated with this posterior probability distribution can be estimated following the standard procedure outlined in Section 1.1.1. Once the estimated parameters are plugged into the posterior probability distribution, we obtain the predictive mean and predictive variance at the new input point $\mathbf{x}^*$.

The subsequent step involves selecting the input value to be sampled in the next round. We require an acquisition function to guide us in exploring the input space to

minimize the objective function. We first provide a brief overview of several commonly used acquisition functions, such as probability of improvement, expected improvement, and upper confidence bound.

3.2.1 Acquisition Functions

Probability of Improvement

The Probability of Improvement (PI) [97] is quite straightforward, as its name implies. Let f_n^* denote the minimum value of the objective function $f(\cdot)$ observed thus far. PI assesses the function $f(\cdot)$ at the point most likely to yield an improvement over this value. This corresponds to the following utility function associated with evaluating $f(\cdot)$ at a given point x:

$$u_{PI}(x) = \begin{cases} 0 & \text{if } f(x) > f_n^* \\ 1 & \text{if } f(x) \leq f_n^* \end{cases} \tag{3.4}$$

It simply means that PI aims to find the location most likely to yield a lower minimum value compared to the lowest value observed so far. Given observations $(x_i, f(x_i))$ with $i = 1, 2, \cdots, n$, let $\mu_n(x)$ and $\sigma_n^2(x)$ represent the mean and variance from the posterior distribution $f(x)|f(x_1), \cdots, f(x_n)$. This PI acquisition function can be explicitly expressed as:

$$a_{PI}(x) = \mathbb{E}[u_{PI}(x)|(f(x_1), \cdots, f(x_n))] = \Phi(f_n^*; \mu_n(x), \sigma_n^2(x)). \tag{3.5}$$

Here $\Phi(x; \mu_n(x), \sigma_n^2(x))$ represents the cumulative distribution of a normal distribution with mean $\mu_n(x)$ and variance $\sigma_n^2(x)$. The point x with the highest probability of improvement will be selected. One of the main limitations of this PI acquisition function is that it only considers the locations where the objective can be reduced, without quan-

tifying how much improvement can be achieved. This leads to another commonly used acquisition function known as Expected improvement.

Expected Improvement

Expected improvement (EI) [98, 99] is one of the most commonly used acquisition functions in BO. It evaluates objection function $f(\cdot)$ at the point that, in expectation, improves upon f_n^* the most. The utility function of EI is

$$u_{EI}(x) = \max(0, f_n^* - f(x)). \tag{3.6}$$

It means the reward we can receive is the improvement $f_n^* - f(x)$ if the $f(x)$ is less than the observed minima, and do not receive a reward otherwise. The EI acquisition function is expressed as

$$a_{EI}(x) = \mathbb{E}[u_{EI}(x)|f(x_1), \cdots, f(x_n)] = \Delta_n(x)\Phi(\frac{\Delta_n(x)}{\sigma_n(x)}) + \sigma_n(x)\varphi(\frac{\Delta_n(x)}{\sigma_n(x)}), \tag{3.7}$$

where $\Delta_n(x) = f_n^* - \mu_n(x)$, and $\varphi(x; \mu_n(x), \sigma_n^2(x))$ represents the probability distribution of a normal distribution with mean $\mu_n(x)$ and variance $\sigma_n^2(x)$. Unlike the objective function $f(\cdot)$, which may be computationally expensive to evaluate, the acquisition function $a_{EI}(x)$ is typically inexpensive to compute. Its first and second-order derivatives are easy to evaluate, making it amenable to optimization using methods like the quasi-Newton L-BFGS-B algorithm [100].

Upper Confidence Bound

The upper confidence bound, commonly referred to as GP-UCB, was first introduced in [101]. The UCB is often used for maximizing a certain property. In this context,

GP-UCB combines a weighted sum of mean predictions and uncertainties:

$$a_{UCB}(x, \beta) = \mu_n(x) + \sqrt{\beta}\sigma_n(x), \tag{3.8}$$

where $\sqrt{\beta}$ is the tradeoff parameter controlling the exploitation and the exploration. A larger value of $\sqrt{\beta}$ indicates that the optimization becomes more exploratory, favoring the evaluation of locations where the posterior uncertainty is largest. Conversely, a smaller value suggests that the optimization becomes more exploitative, focusing on locations with a higher predicted mean value. This can be explained as selecting the point for the next sampling such that it provides a reasonable upper bound on $f(x)|f(x_1), \cdots, f(x_n)$.

3.2.2 Examples of Bayesian Optimization

There are several popular packages available for performing baseline Bayesian optimization (BO) in low-dimensional spaces. Notable examples include GPyOpt [102] and bayesian-optimization [103], both of which are `Python` tools designed for optimizing black-box functions using GPs. In `R`, DiceOptim [74], can be used for optimizing expensive-to-evaluate deterministic functions. Here, we will use two simulated functions as examples with `Python` package GPyOpt and bayesian-optimization to illustrate the typical procedure of Bayesian optimization.

One-dimensional Function

Suppose we aim to maximize a one-dimensional simulated function over the interval $[3, 7.5]$, which is defined as:

$$f(x) = \sin(x) + \sin(\frac{10x}{3}). \tag{3.9}$$

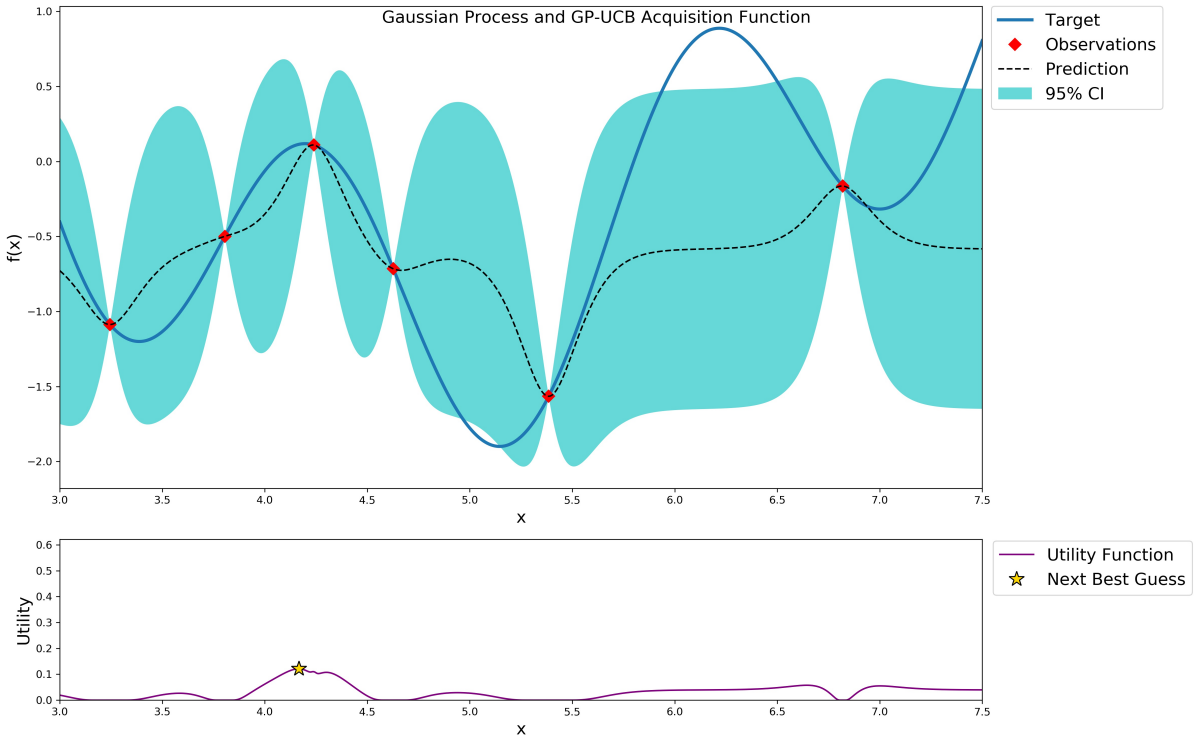

Figure 3.1: Predictive distribution and acquisition function for an unobserved sample based on the first five initial observations. The blue curve represents the true function, the red points denote the observed data, the black dashed curve shows the predicted mean function, and the green-shaded region corresponds to the 95% confidence interval.

Using the GP-UCB acquisition function with a tradeoff parameter $\sqrt{\beta} = 5$ and given the initial six observations randomly selected from the interval, we obtain the predictive distribution and acquisition function values for an unobserved sample within the range $x \in [3, 7.5]$, as shown in Figure 3.1. In this example, we utilize the `Python` package bayesian-optimization to compute these quantities. Based on the acquisition function plot, the next best guess for the maximum occurs at $x = 4.24$.

After evaluating the function defined in Equation (3.9) at the suggested location $x = 4.24$, the observation set is expanded, leading to an update in the predictive distribution and acquisition function value. This update is illustrated in Figure 3.2. As more observations are acquired, the predictive curve aligns more closely with the real function

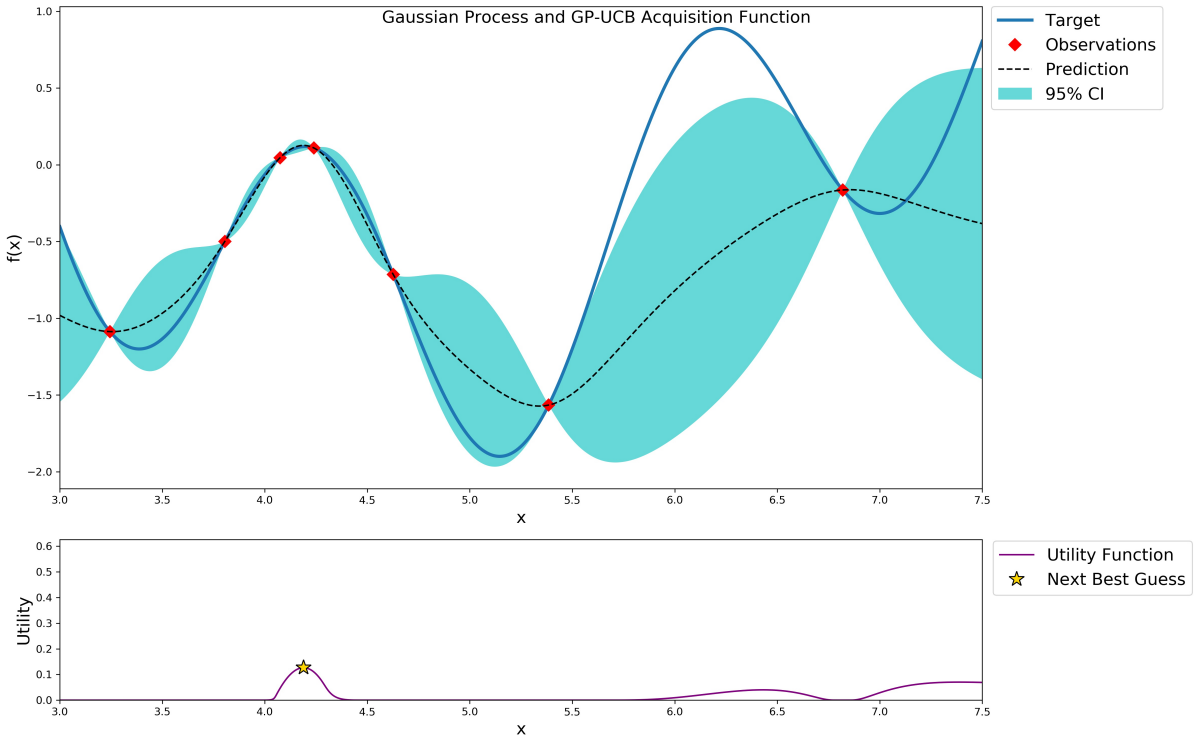

Figure 3.2: Predictive distribution and acquisition function for an unobserved sample given the first six observations and one additional sample evaluated at $x = 4.24$. The blue curve represents the true function, the red points denote the observed data, the black dashed curve shows the predicted mean function, and the green-shaded region corresponds to the 95% confidence interval.

curve, resulting in a more accurate prediction interval.

The function will be evaluated iteratively at a location with the best acquisition value, and the observation set will be expanded until we reach the predefined evaluation limit. After eight additional evaluations, given the existing fourteen observations, the updated predictive distribution and acquisition function are plotted in Figure 3.3. The suggested location for maximum exploration is $x = 6.22$, which is very close to the global maximizer located around $x = 6.217$.

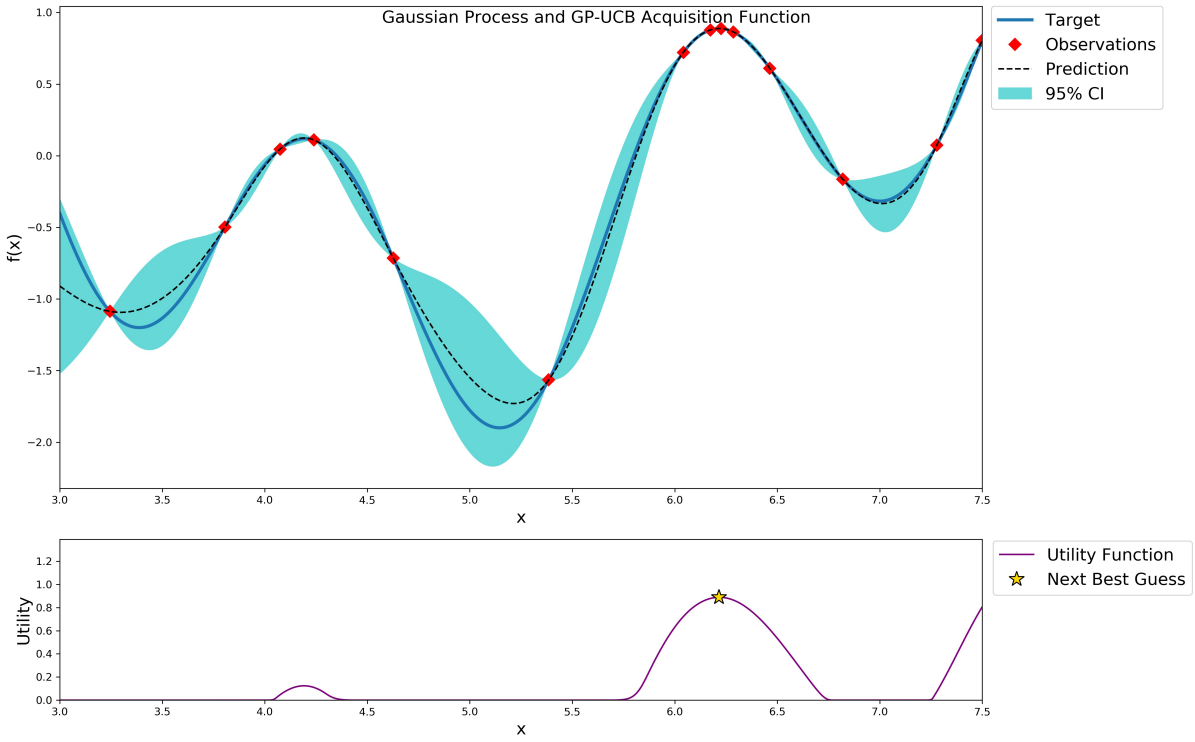

Figure 3.3: Predictive distribution and acquisition function for an unobserved sample given the initial six observations and additional eight samples evaluated beforehand. The blue curve represents the true function, the red points denote the observed data, the black dashed curve shows the predicted mean function, and the green-shaded region corresponds to the 95% confidence interval.

Two-dimensional Ackley Function

The Ackley function [104] is defined as

$$f(x_1, x_2) = -20e^{-0.2\sqrt{0.5*(x_1^2+x_2^2)}} - e^{0.5(\cos(2\pi x_1)+\cos(2\pi x_2))} + e + 20. \tag{3.10}$$

As shown in Figure 3.4, the Ackley function is a non-convex function characterized by multiple local minima and a global minimum at the point $(0, 0)$. It is commonly used to test the performance of optimization algorithms. In this example, we employ Bayesian optimization to minimize the Ackley function within a constrained space, where $x_1 \in$

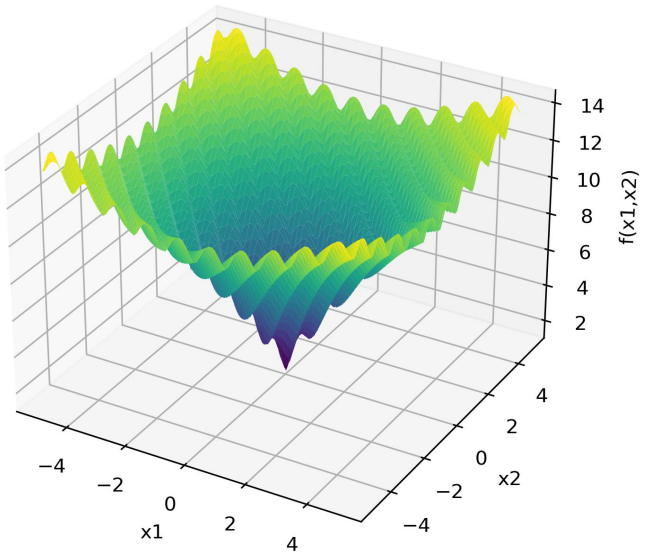

Figure 3.4: Two-dimensional Ackley function.

$[-5, 5]$, and $x_2 \in [-5, 5]$, utilizing the `Python` package GPyOpt.

We begin with ten initial observations, represented as colored dots in the first two plots, and for this example, we opt to use the EI acquisition function. With these observations, we can determine the posterior mean and posterior standard deviation at any unsampled location $\mathbf{x}^*$. The posterior standard deviation provides valuable information about the degree of uncertainty regarding the prediction at $\mathbf{x}^*$. The subsequent step involves selecting the points to be sampled in the next round. As illustrated in the third plot of Figure 3.5, the location of the colored dot is the location of the maximum value of the EI acquisition function, which will be selected to be evaluated in the next round.

After updating the observed dataset, the posterior distribution is also updated, and this iterative process continues until we reach a predefined threshold or time limit of evaluation. Figure 3.6 presents the distances between consecutive selected $\mathbf{x}_i$ values and the values of the best-selected samples over the 20 iterations. In this example with mul-

tiple minima, we find that BO can effectively explore a broad input space and eventually converge to the global minimum when the input dimension is small. The pseudo-code for BO is outlined in Algorithm 1.

Although BO works fine for problems with a small dimension, our primary objective is to minimize potential energy with respect to atoms' positions, and the position vector often has a high dimension. This endeavor presents a challenge due to the considerably higher input dimensionality compared to the previously illustrated BO examples. Furthermore, to find the molecular structures with any desired target energy value or atomic force value, we require a suitable loss function. This function, similar to the acquisition function in BO, serves to guide the inverse exploration process. Additionally, the molecules we seek to identify must adhere to chemical constraints, which include maintaining reasonable chemical bond lengths and bond angles. For molecules outside the reasonable range of bond lengths and angles, the simulation will not converge to a finite value. The conventional BO approach, designed for lower-dimensional input spaces is not efficient to solve problems with high-dimensional inputs. As we showed later, direct minimization of the energy is hard, as we do not have enough data to predict energy around minimum values. Instead, we will introduce a novel methodology to minimize the atomic force, whereas we often have abundant data for the entire range to train the AFF emulator. When the atomic forces are minimized around zero, the molecule is at the equilibrium state with minimum energy.

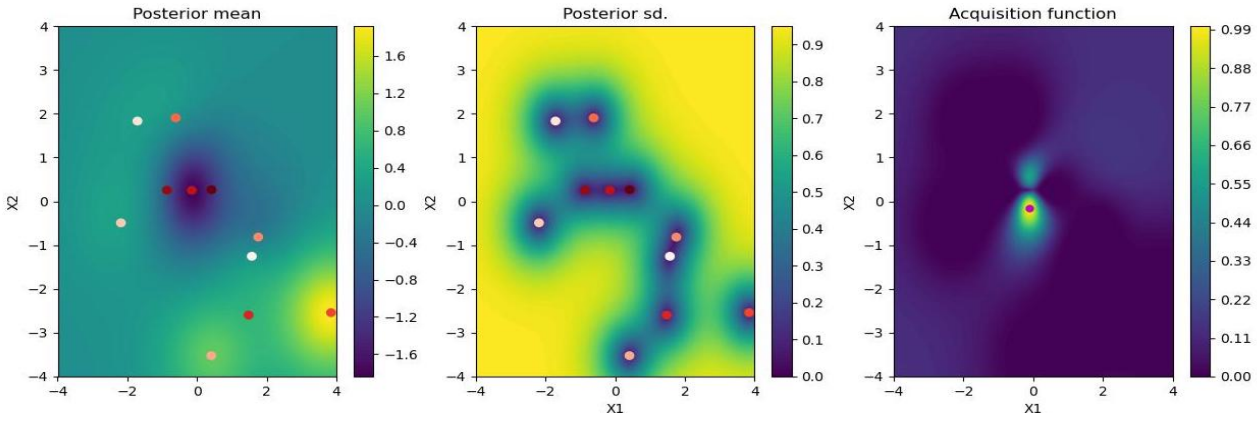

Figure 3.5: The plots illustrate the posterior distribution $y(\mathbf{x}^*)|(y(\mathbf{x}_i), \cdots, y(\mathbf{x}_n))$ and the value of the acquisition function (expected improvement) in the Ackley function minimization example. In the first two plots, the colored dots represent ten initial observations. The heatmap indicates the values of the posterior mean, posterior standard deviation, and the acquisition function. The dot in the third plot denotes the selected location for the next sampling, corresponding to the maximum acquisition function value. This figure is generated using GPyOpt

Algorithm 1 Pseudo-code for Bayesian optimization

1: Place a Gaussian process prior to $f(x)$

2: Observe $f(\cdot)$ at n_0 points according to an initial space-filling experimental design on designed space $\mathcal{A}$. Set n = n_0

3: **while** $n \leq N$ **do**

4: Update the posterior probability distribution using all available data

5: Let x_n be a maximizer of the acquisition function over x, where the acquisition function is computed using the current posterior distribution

6: compute $f(x_n)$ at x_n

7: increment n

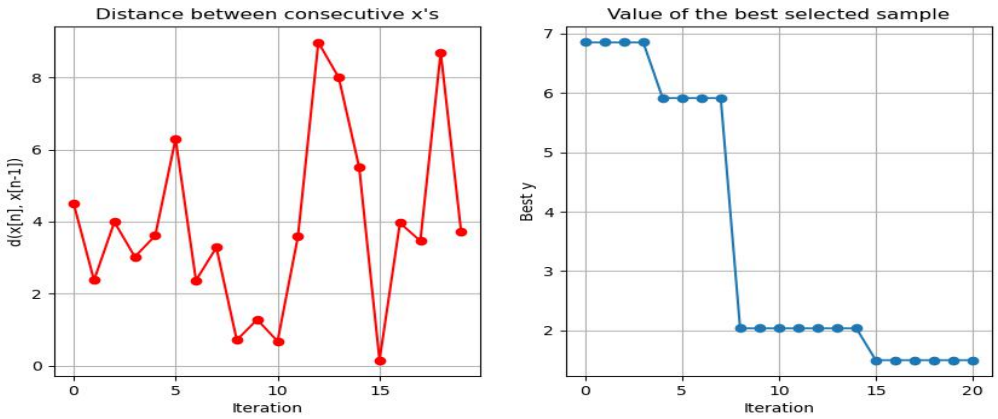

Figure 3.6: The left plot illustrates the distances between consecutive selected $\mathbf{x}_i$ values during Bayesian Optimization iterations. The right plot displays the values of the best-selected samples throughout the iterations. In this example, the predefined evaluation limit is set to 20. This plot is generated by GPyOpt

3.3 Learning the Inverse Energy Mapping

Let us first consider the strategy for constructing an inverse mapping from the molecule's potential energy to its molecular structure. Given a desired potential energy value E_T, with the subscript T meaning the target value, our goal is to provide a set of valid molecule structures that satisfy the condition $E(\mathbf{x}^*) = E_T$. The desired property value such as the target potential energy value E_T is predefined by the user. Firstly, we define a loss function without considering the predictive uncertainty:

$$\mathcal{L}(\mathbf{x}^*) = \|E_T - \hat{E}(\mathbf{x}^*)\|_2^2, \tag{3.11}$$

where $\hat{E}(\mathbf{x}^*)$ represents the predicted value at $\mathbf{x}^*$ obtained from the emulator. This transforms the inverse mapping problem into an optimization problem concerning the loss function. In this context, the loss function plays the role of the acquisition function in BO. Minimizing the loss function provides us with guidance in finding the direction

towards the structure with the desired properties.

To illustrate the steps involved in this inverse mapping process, we will employ the GPR as the surrogate model with the same descriptor and kernel function utilized in the AFF method. Let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_M\}$ be M training configurations, and $\mathbf{E} = (E_1, E_2, \cdots, E_M)^T$ be a energy vector of those configurations. We use the same descriptor $\mathbf{D}(\cdot)$ and the same Matérn kernel function with roughness parameter $5/2$ as shown in the AFF method. Let $K(\mathbf{D}(\mathbf{x}_a), \mathbf{D}(\mathbf{x}_b)) = \left(1 + \sqrt{5}\frac{d}{\gamma} + \frac{5d^2}{3\gamma^2}\right) \exp\left(-\sqrt{5}\frac{d}{\gamma}\right)$ define the correlation between $\mathbf{x}_a$ and $\mathbf{x}_b$, where γ is the range parameter, and d is the Euclidean distance between the descriptors $\mathbf{D}(\mathbf{x}_a)$ and $\mathbf{D}(\mathbf{x}_b)$. The energy $\mathbf{E}$ is assumed to follow a multivariate normal distribution given by

$$\mathbf{E} \mid \theta, \gamma, \sigma^2, \lambda \sim \mathcal{MN}(\boldsymbol{\mu}, \sigma^2(\mathbf{R} + \lambda \mathbf{I}_M)), \tag{3.12}$$

where $\mathbf{R}$ is the correlation matrix among the data with $(i, j)_{th}$ element $R_{ij} = K(\mathbf{D}(\mathbf{x}_i), \mathbf{D}(\mathbf{x}_j))$, λ is the nugget parameter of the random noise in $\mathbf{E}$, and $\boldsymbol{\mu} = [\theta, \cdots, \theta]^T$ with θ being the mean parameter.

Conditional on γ and λ, the maximum likelihood estimator of θ is $\hat{\theta} = (\mathbf{H}^T(\mathbf{R} + \lambda \mathbf{I}_M)^{-1}\mathbf{H})^{-1}\mathbf{H}^T(\mathbf{R} + \lambda \mathbf{I}_M)^{-1}\mathbf{E}$, where $\mathbf{H} = [1, \cdots, 1]^T$, and the MLE of σ^2 is $\hat{\sigma}^2 = S^2/M$ with $S^2 = (\mathbf{E} - \mathbf{H}\hat{\theta})^T(\mathbf{R} + \lambda \mathbf{I}_M)^{-1}(\mathbf{E} - \mathbf{H}\hat{\theta})$. The nugget parameter λ and the range parameter γ can be estimated through MLE by numerically optimizing the profile likelihood or through cross-validation with respect to squared error loss in predictions. Conditional on the estimated parameter $\hat{\theta}$, $\hat{\gamma}$, $\hat{\sigma}^2$, and $\hat{\lambda}$, the predictive distribution of potential energy $E^*(\mathbf{x}^*)$ at a test input $\mathbf{x}^*$ follows

$$E^*(\mathbf{x}^*) \mid \mathbf{E}, \hat{\theta}, \hat{\gamma}, \hat{\sigma}^2, \hat{\lambda} \sim \mathcal{N}\left(\hat{E}^*(\mathbf{x}^*), \hat{\sigma}^2 K^*(\mathbf{x}^*, \mathbf{x}^*)\right), \tag{3.13}$$

with

$$\hat{E}^*(\mathbf{x}^*) = \hat{\theta} + \mathbf{R}_{*M}(\mathbf{R} + \hat{\lambda}\boldsymbol{I}_M)^{-1}(\mathbf{E} - \hat{\theta}\mathbf{H}), \tag{3.14}$$

and

$$K^*(\mathbf{x}^*, \mathbf{x}^*) = K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}^*)) - \mathbf{R}_{*M}(\mathbf{R} + \hat{\lambda}\boldsymbol{I}_M)^{-1}\mathbf{R}_{*M}^T, \tag{3.15}$$

where $\mathbf{R}_{*M}$ is the $1 \times M$ correlation matrix for $\mathbf{x}^*$ and input configurations $\mathbf{X}$ with the i_{th} element being $K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}_i))$.

The predicted value $\hat{E}(\mathbf{x}^*)$ from predictive mean function in nature is a weighted sum of the kernel function $K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}_i))$ on all training samples. The analytical form of the gradient of $K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}_i))$ w.r.t $\mathbf{x}^*$ can be easily calculated through chain rule.

The primary concern of using the loss function in Equation (3.11) lies in the mismatch of the region of the training sample and target value, which limits the predictive accuracy for the region from the training sample space. To illustrate this point, consider the case where a target energy value of $\hat{\theta}$ is set. Due to the nature of the predicted value as a weighted sum of kernel functions, the minimization of the initial loss function might eventually favor a structure situated far from all training molecule configurations. Consequently, the weighted kernel function values $K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}_i))$ all become zero, often resulting in a structurally invalid configuration with implausible bond lengths and angles. In this context, "invalid structure" refers to a chemically unrealistic arrangement where the bond lengths and bond angles deviate significantly from typical values. As plotted in Figure 3.7, after optimizing this initial loss function, the obtained structure with bond lengths and angles significantly deviating from expected ranges, is chemically unrealistic.

Fortunately, we can utilize the uncertain information provided by the GP model. The predictive variance from the forward GP model could help us avoid the over-exploration of potentially invalid regions. Inspired by the GP-UCB acquisition function, the loss

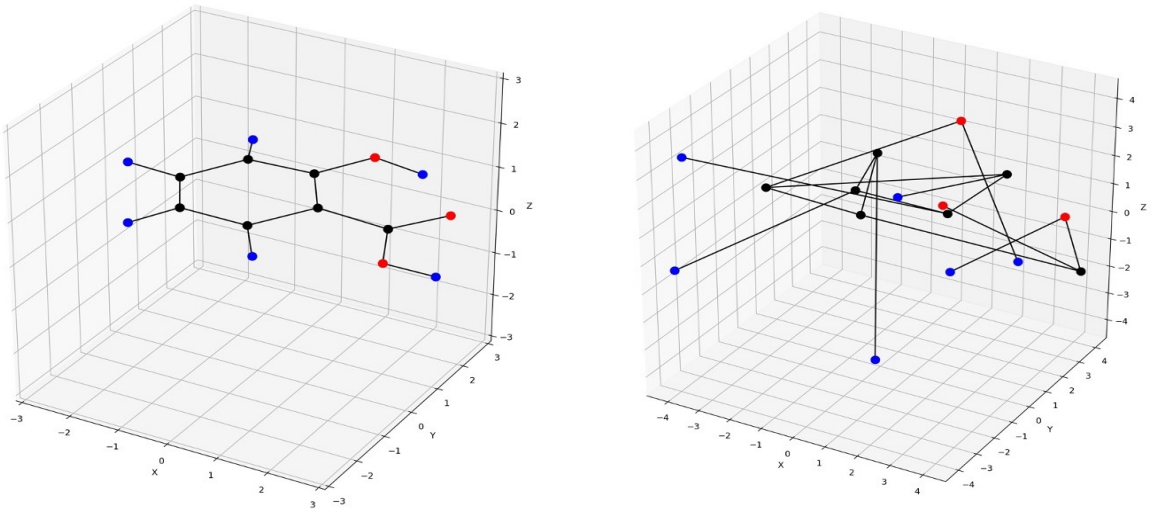

Figure 3.7: The black, blue, and red dots in the figure represent carbon, hydrogen, and oxygen atoms, respectively. The initial salicylic acid structure is shown on the left, while the structure on the right, obtained through the optimization of the initial loss function, is evidently chemically unrealistic.

function is defined as

$$\mathcal{L}(\mathbf{x}^*) = \|E_T(\mathbf{x}^*) - \hat{E}(\mathbf{x}^*)\|_2^2 + \beta\hat{\sigma}^2\|K^*(\mathbf{x}^*, \mathbf{x}^*)\|^2. \tag{3.16}$$

Here, $\hat{\sigma}^2 K^*(\mathbf{x}^*, \mathbf{x}^*)$ represents the predictive variance of energy at $\mathbf{x}^*$, and β serves as the trade-off parameter governing the balance between exploration and exploitation. In practice, we employ the maximum likelihood estimator $\hat{\sigma}^2$ in the evaluation of the loss function. The trade-off parameter β is determined via a logarithmic grid search. We usually initiate the selection of the trade-off parameter β with a smaller value and increase it if the simulator is unable to perform the calculation on the proposed structure. Unlike the GP-UCB acquisition function, this approach tends to prioritize exploitation in the vicinity when a large value of β is set, and it leans more towards exploration with a smaller β. The loss function in Equation (3.16) aims to guide the search towards regions that are closer to the target while ensuring that these regions are not too distant from

the observed space. This approach prevents the exploration from becoming trapped in invalid regions.

The procedure for inversely optimizing potential energy is outlined in Algorithm 2. Due to the high dimensionality of the input space, which is of size $3N$, conducting a grid search exploration strategy to minimize the loss function is unrealistic, despite the loss function's computational simplicity. Existing BO packages cannot handle optimization tasks in such a high-dimensional space. Nevertheless, with the existing AFF framework, it is feasible to readily compute the gradient vectors for both predictive mean and predictive variance terms, using the specified descriptor and kernel function. We undertake this optimization within the AFF model, which leverages PyTorch's automatic differentiation functionality [105]. In practice, we employ the Adam optimizer [106] with empirically selected parameters to optimize the loss function.

To aid the algorithm in escaping local optima and finding better solutions, we introduce random noise to the most recently proposed configuration. The resulting configuration with added random noise can then serve as the starting point for the next iteration of exploration. Finally, this approach will yield a proposed configuration $\mathbf{x}^*$. It is worth noting that the proposed configuration is not unique and may vary depending on the initial structure and the loss minimization process following the stochastic gradient descent algorithm.

Algorithm 2 Pseudo-code for inverse optimization of potential energy

Input: Pairs of molecules' configuration $\mathbf{x}_i$ and potential energy $E(\mathbf{x}_i)$, where $i = 1, \cdots, M$. User-specified target energy value E_T

1: Fit a forward energy GPR model on all training samples
2: Select an initial molecular structure $\mathbf{x}^*$, and apply the adam optimizer to minimize the loss $\mathcal{L}(\mathbf{x}^*)$
3: Calculates the real energy of $\mathbf{x}^*$: $E(\mathbf{x}^*)$ from the simulator.
4: **while** $\|E(\mathbf{x}^*) - E_T(\mathbf{x}^*)\|_2^2 \geq c$ **do**
5: Adds $\mathbf{x}^*$ into the training set and updates the posterior distribution
6: $\mathbf{x}^* = \mathbf{x}^* + \boldsymbol{\epsilon}$, and select $\mathbf{x}^*$ as the initial molecular structure
7: Apply the adam optimizer to minimize the loss $\mathcal{L}(\mathbf{x}^*)$
8: Calculates the real energy $E(\mathbf{x}^*)$ of $\mathbf{x}^*$ from the simulator

Return: the proposed configuration $\mathbf{x}^*$

One of the practical application goals behind constructing this inverse potential energy mapping is to discover an alternative approach to energy minimization that reduces the need for extensive evaluations from the computationally expensive AIMD simulator. However, in practice, we have found that inversely maximizing the potential energy is easy whereas minimizing the energy directly is hard. In Figure 3.8, we present the actual performance of the inverse energy mapping on aspirin and salicylic acid. The initial forward energy GP model was trained using 400 simulations randomly selected from a small trajectory of the simulation runs. The results reveal that we can easily identify a valid structure with potential energy close to the predefined target energy value, as long as the value of energy falls within the training energy range or is above the minimum energy value obtained from the training samples. The heatmaps show that this method is capable of exploring regions far from the initial input space in this direction within

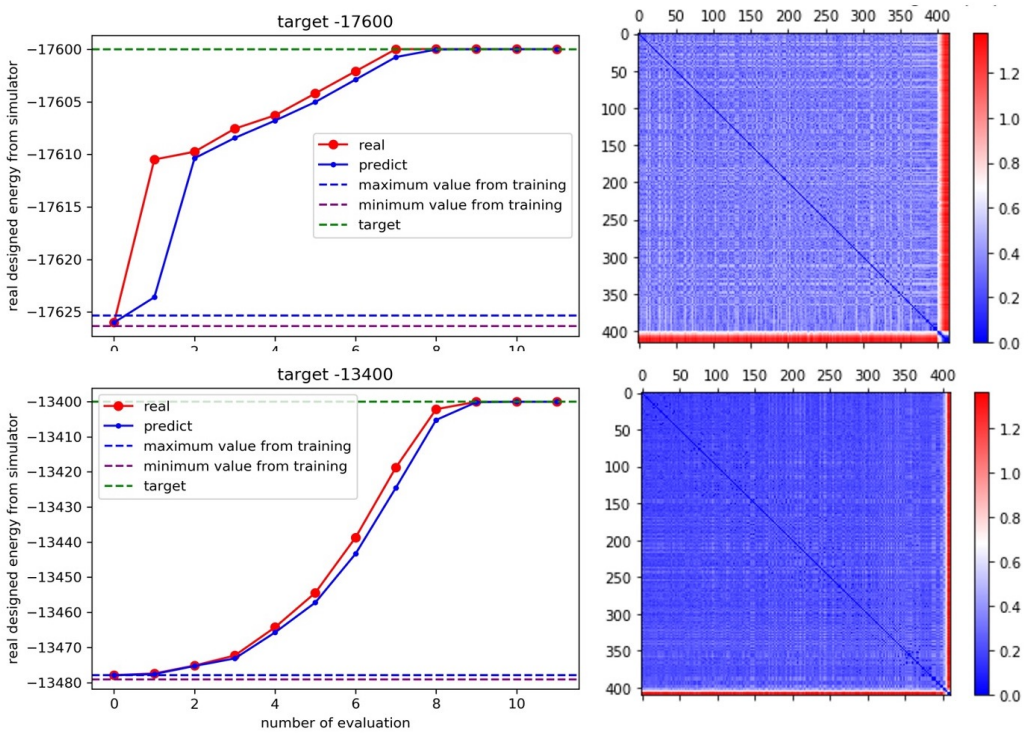

Figure 3.8: The inverse energy mapping for aspirin (top) and salicylic acid (bottom). The two figures on the left depict the proposed energy (eV) versus the number of evaluations. The green dashed line represents the predefined target energy value, while the purple and blue dashed lines indicate the minimum and maximum energy values observed among the initial 400 training samples. Blue solid points represent the predicted value from the GP energy emulator, while red solid points represent the real energy values evaluated by the simulator. The two heatmaps on the right show the Euclidean distances between the training and proposed structures. Red indicates different structures, while blue indicates similar structures. The target energy value represents an arbitrary, user-specified energy value that we aim to achieve.

just a few evaluations.

Directly minimizing the energy is challenging due to the lack of structure with small energy in the training data set. The simulated energy from the proposed structure always falls above the lower bound of the training set, even when the predicted mean from the previous posterior distribution is less than this lower bound. This observation aligns with the laws of physics, as it is inherently challenging to locate structures with very low energy values. Unlikely the potential energy from the randomly selected configurations always

lies on one side of the energy axis, while our ideal target energy value is on the other side, atomic forces are more evenly spaced and also bring the important information. To solve this energy minimization problem, we can learn the inverse mapping from atomic positions to atomic force vectors.

3.4 Minimization of Atomic Forces

The primary challenge encountered with energy minimization through inverse energy mapping is that the training trajectories selected from the simulation samples consistently fall on the higher-energy side of the threshold we aim to reach. However, in contrast to the potential energy, most of the atomic forces exhibit variations on both sides of the target value, which is zero, as shown in Figure 3.9. Now, the inverse mapping also operates within a high-dimensional space, with the objective of locating structures where all atomic forces closely approximate zero. Fortunately, our previously developed AFF emulator inherits the physical connection between energy and force vectors, and it demonstrates state-of-the-art accuracy at the level of AIMD while reducing computational complexity. Therefore, it serves as a reliable forward model that we can utilize in inverse force mapping.

Following the notations in the AFF force field model, consider a molecule that has N atoms grouped into L PE atom sets, each set containing l_i atoms, for $i = 1, ..., L$, and denote $\mathbf{F}_i$ with dimension $3Ml_i$ as the forces of the atoms of i_{th} PE set in M training configurations. Let $\mathbf{F}_i^T(\mathbf{x}^*)$ be the target force vector of $\mathbf{x}^*$ on the atoms of i_{th} PE set. Given the training set, we can build an AFF model of l prior distributions in parallel, numerically fitting the range parameter γ and the nugget parameter λ. The predictive distribution of the atomic forces in the i-th PE atom set $\mathbf{F}_i(\mathbf{D}(\mathbf{x}^*))$ at any configuration

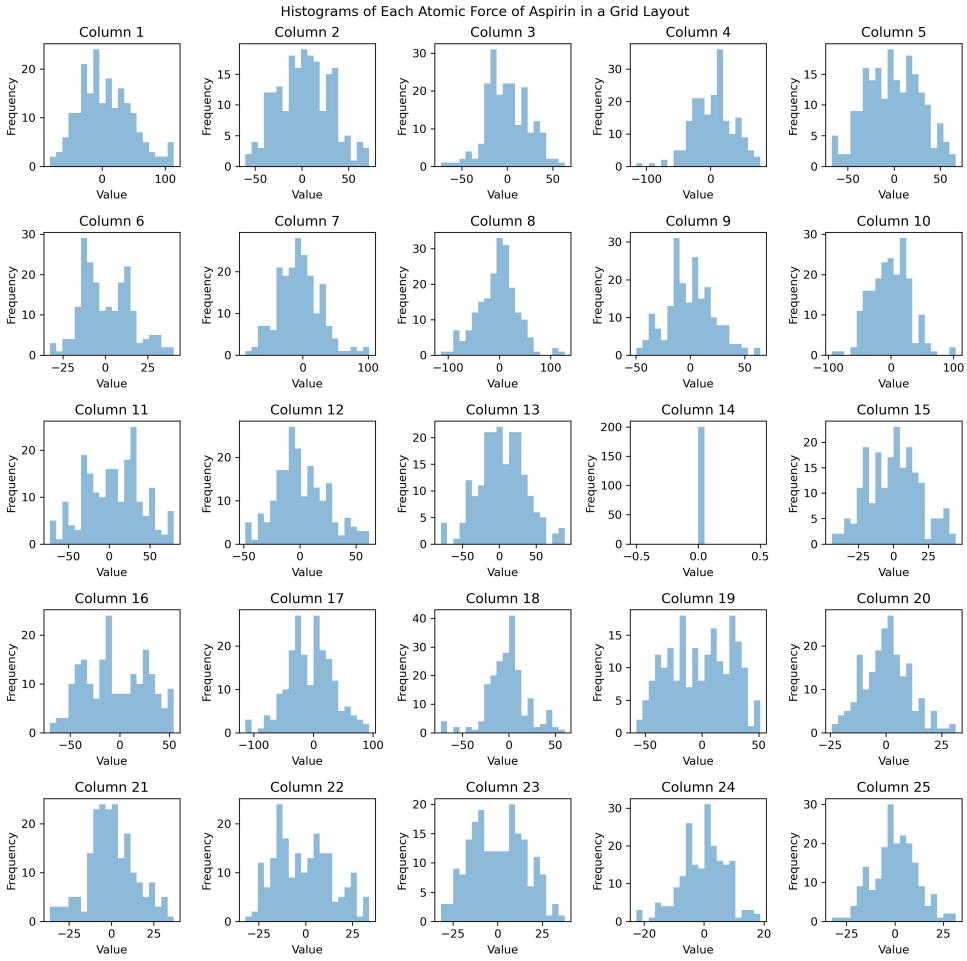

Figure 3.9: The histograms represent the atomic forces on the first 25 dimensions observed in a randomly selected segment of the aspirin AIMD simulation. Each individual subplot displays the distribution of atomic forces along one specific dimension.

$\mathbf{x}^*$ follows a multivariate normal distribution

$$\left(\mathbf{F}_i(\mathbf{D}(\mathbf{x}^*)) \mid \mathbf{F}_i, \hat{\boldsymbol{\theta}}_i\right) \sim \mathcal{MN}(\hat{\mathbf{F}}_i(\mathbf{x}^*), \hat{\sigma}_i^2 \mathbf{K}_i^*(\mathbf{x}^*, \mathbf{x}^*)), i = 1, \cdots, L, \tag{3.17}$$

where the predictive mean vector and predictive covariance matrix follows

$$\hat{\mathbf{F}}_i(\mathbf{x}^*) = \mathbf{R}_{\mathbf{x}^*}^T(\mathbf{R} + \hat{\lambda}\mathbf{I}_{3Ml_i})^{-1}\mathbf{F}_i, \tag{3.18}$$

$$\hat{\sigma}_i^2\mathbf{K}_i^*(\mathbf{x}^*, \mathbf{x}^*) = \hat{\sigma}_i^2(\mathbf{R}^* - \mathbf{R}_{\mathbf{x}^*}^T(\mathbf{R} + \hat{\lambda}\mathbf{I}_{3Ml_i})^{-1}\mathbf{R}_{\mathbf{x}^*}), \tag{3.19}$$

with $\mathbf{R}^*$ being a $3l_i \times 3l_i$ Hessian matrix of the kernel function $\nabla_{\mathbf{x}^{*i}} K(\mathbf{D}(\mathbf{x}^*), \mathbf{D}(\mathbf{x}^*))\nabla_{\mathbf{x}^{*i}}^T$.

Similar to the inverse energy map, the loss function is defined as:

$$\mathcal{L}(\mathbf{x}^*) = \sum_{i=1}^{L} \|\mathbf{F}_i^T(\mathbf{x}^*) - \hat{\mathbf{F}}_i(\mathbf{x}^*)\|_2^2 + \beta \sum_{i=1}^{L} \hat{\sigma}_i^2 \mathrm{Tr}(\mathbf{K}_i^*(\mathbf{x}^*, \mathbf{x}^*)), \tag{3.20}$$

where $\hat{\sigma}_i^2$ is the MLE estimator of σ_i^2 and the $\mathbf{F}_i^T(\mathbf{x}^*)$ is desired force vector, which is intentionally set to a zero-force vector in order to optimize the energy structure. The second term of the loss function can be interpreted as a summation of the predictive variance of all atomic forces. The gradient of this $\nabla_{\mathbf{x}^*}\mathcal{L}(\mathbf{x}^*)$ also can be analytically solved through:

$$\nabla_{\mathbf{x}^*}\mathcal{L}(\mathbf{x}^*) = 2\sum_{i=1}^{L}(\mathbf{F}_i^T(\mathbf{x}^*) - \nabla_{\mathbf{x}^*}\hat{\mathbf{F}}_i(\mathbf{x}^*)) + 2\beta \sum_{i=1}^{L} \hat{\sigma}_i^2 \sum_{j=1}^{3l_i} \nabla_{\mathbf{x}^*}[\mathbf{K}_i^*(\mathbf{x}^*, \mathbf{x}^*)]_{jj}, \tag{3.21}$$

where $[\mathbf{K}_i^*(\mathbf{x}^*, \mathbf{x}^*)]_{jj}$ denote the j-th diagonal element of matrix $\mathbf{K}_i^*(\mathbf{x}^*, \mathbf{x}^*)$.

Algorithm 3 Pseudo-code for Inverse Force Design

Input: target force vector $\mathbf{F}^T$, molecule's configuration $\mathbf{x}_i$ and force vectors $\mathbf{F}(\mathbf{x}_i)$, where $i = 1, \cdots, M$.

1: Partition the PE atoms and force vectors follow the AFF method and fit a forward AFF force field model on all training samples.
2: Select an initial molecular structure $\mathbf{x}^*$, and apply the adam optimizer to minimize the loss $\mathcal{L}(\mathbf{x}^*)$.
3: Calculates the simulated force vector of $\mathbf{x}^*$: $\mathbf{F}(\mathbf{x}^*)$ from the simulator.
4: **while** $\|\mathbf{F}(\mathbf{x}^*) - \mathbf{F}_T\|_2^2 \geq c$ **do**
5: Adds $\mathbf{x}^*$ into the training set and updates the posterior distribution.
6: $\mathbf{x}^* = \mathbf{x}^* + \epsilon$, and select $\mathbf{x}^*$ as the initial molecular structure.
7: Apply the adam optimizer to minimize the loss $\mathcal{L}(\mathbf{x}^*)$.
8: Calculates the simulated force $\mathbf{F}(\mathbf{x}^*)$ of $\mathbf{x}^*$ from the simulator

Return: the proposed configuration $\mathbf{x}^*$.

Algorithm 3 presents the pseudo-code for the proposed inverse force design method. It's worth noting that during the iterative procedure in practice, the estimated range parameter γ and nugget parameter λ can be held constant, and there's no need to retrain the model every time new data points are added to the training set. This won't impact the model's prediction performance, and these parameters only need to be retrained after batches of iterations. A similar approach was employed in [107]. By keeping the range and nugget parameters fixed, we can significantly reduce computational costs when updating the posterior distribution. The primary computational load arises from the matrix inversion of $(\mathbf{R} + \hat{\lambda}\mathbf{I}_{3Ml_i})^{-1}$, and we can apply a similar block matrix inversion technique as described in Appendix A.1 to optimize this computation.

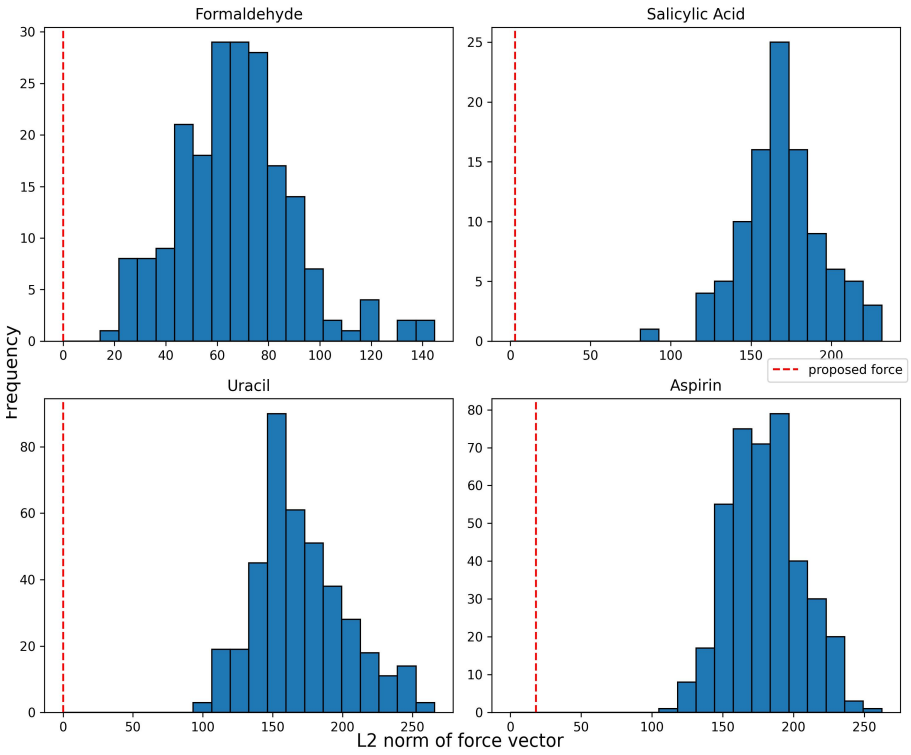

Figure 3.10: The histograms represent the L2-norm value of the force vectors observed in initial training simulations on formaldehyde, salicylic acid, uracil, and aspirin. The red dashed line shows the L2-norm value of force from the proposed structures within 10 evaluations. The initial training sample sizes are 200 (formaldehyde), 100 (salicylic acid), 400 (uracil) and 400 (aspirin).

3.5 Numerical Results

We applied the proposed inverse force design approach to various simulated molecule structures, including formaldehyde, uracil, aspirin, and salicylic acid. Our goal was to identify valid structures with the desired target of zero atomic force vectors and subsequently locate local or global energy-minimized structures. To collect training configurations, we initiated simulations with an initial structure and gathered the first several hundred configurations at fixed intervals along the AIMD trajectories. These configurations obtained from the simulator typically exhibited relatively high energy values and large L2 norm values for atomic force vectors.

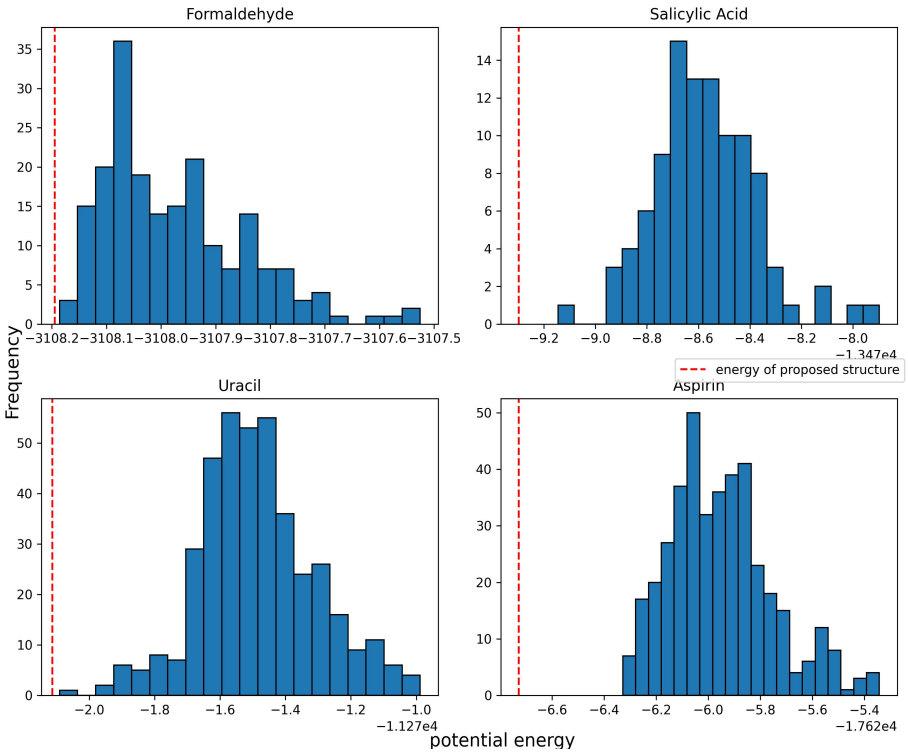

Figure 3.11: The histograms represent the potential energy value (negative in ev) of the force vectors observed in initial training simulations on formaldehyde, salicylic acid, uracil, and aspirin. The red dashed line shows the potential energy value from the proposed structures within 10 evaluations. The initial training sample sizes are 200 (formaldehyde), 100 (salicylic Acid), 400 (uracil) and 400 (aspirin).

As shown in Figure 3.10, the estimated inverse force mapping effectively pinpointed valid saddle structures with force vectors close to zero. These structures were identified starting from the initial configurations, which had high L2 norm force values and were randomly selected from the initial training set. This outcome shows the effectiveness of the proposed approach in mapping the force field back to the input molecule's 3D structures.

We are also interested in the potential energy values of the proposed structures. From Figure 3.11, we can observe that the energy of the proposed structures with force vectors close to zero falls below the lower threshold of the potential energy observed in the simulated trajectories. This achievement was not attainable using the previous

inverse energy mapping method. This demonstrates the feasibility of energy optimization through inverse force mapping, as the gradients provide equally vital information.

It's important to note that the AFF model, trained with only a few hundred simulations, can identify structures with force vectors close to zero in just two evaluations for simpler molecules like formaldehyde (H2CO). For these simpler structures, the AFF model demonstrates impressive learning capabilities, achieving state-of-the-art accuracy in predicting atomic force vectors from 3D structural inputs. Therefore, inverse mapping of the force vector can be accomplished in just one or two evaluations. However, for molecules like aspirin, the L2 norm of the force vector for the proposed structure after 10 evaluations remains somewhat distant from zero. This is due to the lower accuracy of the AFF model trained on 400 simulations compared to other molecules. It may require more evaluations or a more accurate forward model to optimize the force vector towards zero, but the potential energy of the displayed proposed structures already significantly exceeds the lower bound set by the training potential energy data.

We also illustrate the proposed molecular structure during the inverse force design process using salicylic acid as an example. Figure 3.12 presents 3D configurations of salicylic acid along with atomic force vectors during the inverse zero force vector design process. In practice, we can initiate this process from any structure in the training set. For comparative purposes, we start with the molecular configuration characterized by the lowest L2 norm value of the force vector. Notably, the initial structure exhibits significant atomic forces. Nevertheless, as each iteration progresses, the values of the atomic force vectors in the proposed structure consistently decrease. The capability to discover zero-force structures improves with additional evaluations. After just five evaluations, we are able to propose a 3D salicylic acid structure with atomic forces that are close to zero.

Finally, we are interested in assessing the similarity between the minimized structures obtained using our method and those from classical conventional geometry optimization

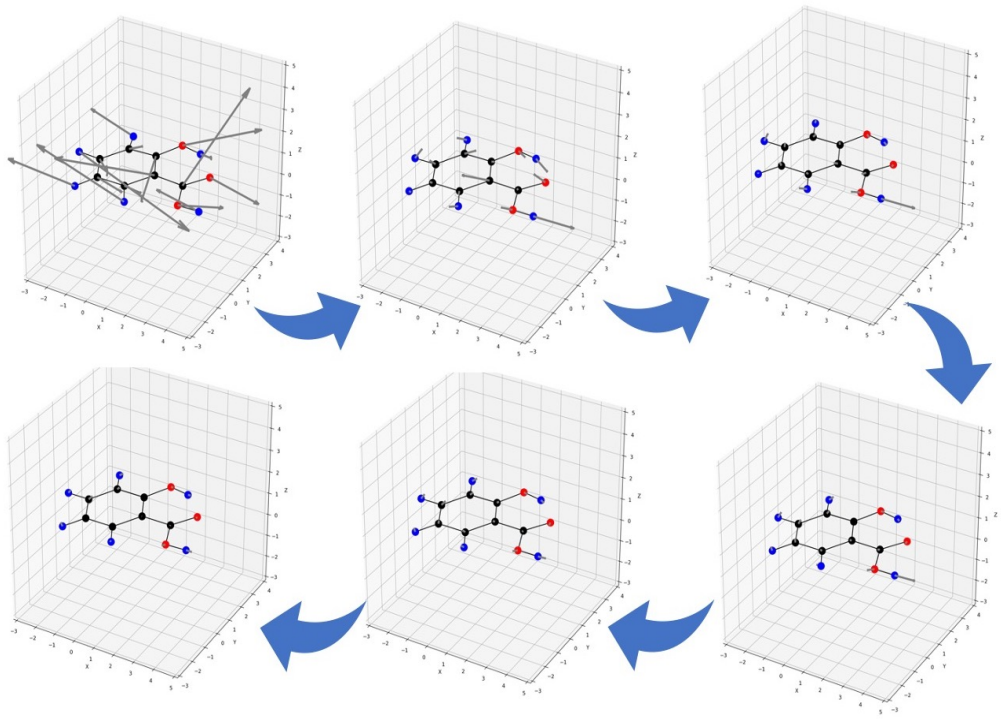

Figure 3.12: The atomic force graph of salicylic acid during the inverse force design is shown here. In this visualization, carbon, oxygen, and hydrogen atoms are denoted by black, red, and blue solid dots, respectively. The grey arrows represent the atomic force vectors acting on each atom. The length of each arrow corresponds to the magnitude of the force vector, and all arrows are drawn to the same scale for comparison. The top-left configuration represents the initial structure, while the bottom-left configuration depicts the proposed structure after five evaluations. All the force vector values are computed using the AIMD simulator.

simulations. We continue to use salicylic acid as an example, as its atoms are permutationally distinguishable, and the order of atoms can be uniquely identified. To highlight the structural differences between two 3D molecule structures, we plot the absolute difference between their pairwise distance matrices, which is not affected by rotation and translation. Specifically, the color value of the difference between structures $\mathbf{x}_a$ and $\mathbf{x}_b$ at the (i,j)-th element is given by $|D_{ij}(\mathbf{x}_a) - D_{ij}(\mathbf{x}_b)|$, where $D_{ij}(\mathbf{x}_a)$ represents the Euclidean distance between the i-th and j-th atoms of structure $\mathbf{x}_a$.We kept the initial structure the same for both the calculation of the converged energy-minimized structures in the simulator and our inverse force design approach. The first eleven heatmaps illustrate

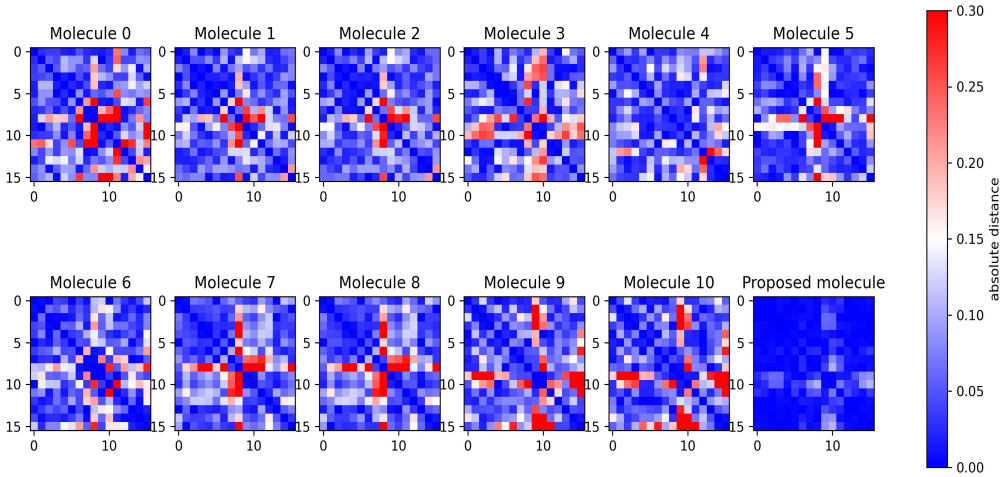

Figure 3.13: The pairwise distance matrix differences between salicylic acid molecule configurations and the converged optimized configuration from the simulator are shown in the heatmaps. Each 16×16 heatmap shows the absolute difference between the pairwise distance matrices of two molecule configurations. The first eleven heatmaps represent Molecule 0 to Molecule 10, which correspond to randomly selected configurations from the training set. The final heatmap illustrates the difference between our proposed structure and the converged structure calculated by the simulator.

significant differences between the training structures and the converged structure calculated by the simulator. However, the last heatmap reveals a remarkable structural similarity between our proposed structure and the converged one, affirming the practical utility of our inverse force design approach for geometry optimization.

The proposed inverse energy and force design method has demonstrated its efficacy in establishing a reliable backward mapping from the potential energy and atomic force field to 3D molecular structures. Leveraging the foundation laid by the efficient and precise forward AFF emulator, which offers the valuable capability of quantifying prediction uncertainties, this method effectively navigates the solution space towards target energy or force values while intelligently avoiding potentially invalid regions. Moreover, our

exploration extends to the domain of energy minimization, presenting a novel alternative approach to performing energy minimization, distinct from conventional methods that rely on solving physical equations.

We acknowledge that while the inverse energy and force design method exhibits promising capabilities, it builds upon the forward AFF emulator. While the AFF emulator has significantly reduced computational costs and broadened its applicability to larger systems compared to previous methods like sGDML, it can be expensive when the number of simulations is large. This constraint hinders its application to even larger systems comprising hundreds or thousands of atoms. A few future to address this challenge is discussed in Chapter 5.

Chapter 4

Python Package PyRobustGaSP and AFF

In this chapter, we first introduce a `Python` version package called PyRobustGaSP, which can be easily installed from [46]. This package is a `Python` version of the RobustGaSP package in R [44] and `MATLAB` [45], provides a versatile toolset for conducting fast GP emulation on large datasets, achieving robust parameter estimation, and prediction of GaSP emulators. Additionally, we also demonstrate how to utilize the AFF package to predict molecule potential energy and atomic force fields with uncertainty quantification using our proposed AFF method, as discussed in Chapter 2.

Parallel partial Gaussian stochastic process (PP-GP), introduced in [40], is a popular technique for accelerating the emulation of computer models with vectorized outputs [108, 109, 110] and forecasting nonlinear dynamical systems [111]. Moreover, the robust parameter estimation approaches through marginal posterior mode can make estimation of the model parameters in GPs and PP-GPs more stable [34, 41]. Recognizing the widespread adoption of `Python` in the realms of data science and machine learning, it becomes imperative to make these fast and resilient GP emulation methods accessible

to a broader audience. Our `Python` package, PyRobustGaSP, seamlessly mirrors the complete functionality of RobustGaSP. This means that researchers who are more inclined towards `Python` can harness the power of this package without the need to switch between programming languages.

In the subsequent sections, we will provide a concise overview of PyRobustGaSP and a few illustrative examples.

4.1 Main Functions of PyRobustGaSP

To construct a GP model using PyRobustGaSP, the initial step involves creating a task using the function `PyRobustGaSP.create_task()`. This function serves to generate a data structure of a custom-type task, which allows users to specify the mean function, correlation function, prior distribution for the parameters and to include a noise term or not. Much like the RobustGaSP package in R, these settings are pre-specified in the default setting. The `PyRobustGaSP.create_task()` function is a very important function in PyRobustGaSP, since it delineates all essential components required for model fitting. The only mandatory arguments for this function are `design`, which is an array-like $M \times p$ design matrix $[\mathbf{x}_1^T, .., \mathbf{x}_M^T]^T$, and `response`, which is the M-dimensional output vector $\mathbf{y}$ for M simulation runs for scalar-valued output. In PP-GP, the entire output matrix has dimension $M \times k$ for M vectorized outputs, each having k coordinates.

The default setting in the argument `trend` is a constant function, i.e., $\mathbf{h}(\mathbf{x}) = \mathbf{1}_M$. If assume the mean function in the GP model is zero, we can set the argument `zero_mean="Yes"`. By default, the GP model is defined to be noise-free, which means the nugget parameter is zero. A noise term can be added by specifying the argument `nugget_est=True`. The default setting for the correlation function is the Matérn correlation is $\alpha = \frac{5}{2}$, and other correlation function choices can be specified by using the

argument `kernel_type`. For additional details on those settings, we refer to [44].

Once a task has been created, the subsequent step involves training the PP-GP model of a vectorized output using `train_ppgasp()` or the Robust-GP model of a scalar-valued output using `train_rgasp()`, utilizing the previously obtained task data structure as input. The return data structure encompasses all crucial estimated parameters, including the mean, variance, noise, and range parameters to perform prediction. Upon obtaining the fitted model from either `train_ppgasp()` or `train_rgasp()`, the final step is to compute the predictive distribution of the GP model previously established. This can be accomplished through `predict_ppgasp()` or `predict_rgasp()`. The resulting dictionary includes predictive means, the 95% predictive credible intervals, and the predictive standard deviations at each test point, providing comprehensive insight into the model's predictions.

4.2 Robust Parameter Estimation and Examples

Parameter estimation of covariance parameters such as range and nugget parameters are critical for training a GP model. It's widely recognized that maximum likelihood estimation of these parameters is unstable under certain conditions [32, 33, 112]. For instance, instability may arise due to the Cholesky decomposition of covariance matrices, which can become close to singular. Another scenario is when the covariance matrix is estimated to be nearly diagonal. The objective of robust estimation is to estimate parameters in a way that ensures the following situations do not occur:

$$\text{(i)} \quad \hat{\mathbf{R}} = \mathbf{1}_n \mathbf{1}_n^T$$

$$\text{(ii)} \quad \hat{\mathbf{R}} = \mathbf{I}_n,$$

where $\hat{\mathbf{R}}$ is the estimated correlation matrix. The RobustGaSP and PyRobustGaSP implement robust parameter estimation without compromising the predictive accuracy of the emulator, through utilizing the marginal posterior mode with robust parameter estimation [34, 41].

4.2.1 Example of a One-Dimensional Input Function

The code below provides an example of fitting a robust GaSP using the `train_rgasp()` function from PyRobustGaSP. We utilize a simulated one-dimensional function to generate the input and output datasets, defined as:

$$y(x) = \sin(x) + \sin(\frac{10}{3}x). \tag{4.1}$$

```
from PyRobustGaSP import PyRobustGaSP
from src.functions import *
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import qmc

# Create a PyRobustGaSP model instance
P_rgasp = PyRobustGaSP()
# Generate the training sample
sampler = qmc.LatinHypercube(d=1)
sample_input = 10 * sampler.random(n=20)

# objective function
def simulated_f(x):
 return np.sin(x) + np.sin((10.0 / 3.0) * x)

```

```
sample_output = simulated_f(sample_input)
```

The RobustGaSP model is trained on a randomly selected subset of 20 observations, and subsequently, predictions are made for 1000 test points uniformly selected from the interval $[0, 10]$.

```
# Create a task for model training
task = P_rgasp.create_task(sample_input, sample_output)

# Fit a rgasp model using created task
model = P_rgasp.train_rgasp(task)

# Get testing input and output
testing_input = np.arange(0,10,1/100).reshape(-1,1)
testing_output=simulated_f(testing_input)

# Get the rgasp predict object
testing_predict = P_rgasp.predict_rgasp(model,
                  testing_input)
```

The predictive means of those testing inputs can be accessed through the key `mean`, while the upper bound and lower bound of 95% predictive credible interval can be obtained through the `lower95` and `upper95` keys, respectively. The corresponding plot generated by the following code is shown in the upper part of Figure 4.1. In the plot, the yellow curve represents the actual output values derived from Equation (4.1), while the blue curve depicts the predicted mean values for evenly spaced testing inputs using the trained GP model. The light blue shaded region signifies the 95% prediction interval. The lower part of Figure 4.1 is created using RobustGaSP in R. It's worth noting that the results generated by PyRobustGaSP and RobustGaSP are exactly the same.

```
# Display the plot
```

```
fig, ax = plt.subplots()
# Plot the first line on the axes
ax.plot(testing_input,testing_predict['mean'], label='predicted',
        color = 'blue')

# Plot the second line on the axes
ax.plot(testing_input, testing_output, color = 'yellow', label='real')
ax.fill_between(testing_input[:,0], testing_predict['upper95'],
                testing_predict['lower95'], alpha=0.2,color = 'gray')
# Set the labels and title of the plot
ax.set_xlabel('input')
ax.set_ylabel('output')
# Add a legend to the plot
ax.legend()
plt.show()
```

4.2.2 Example of Multiple-Dimensional Input Function

Branin Function

The Branin Function is a commonly used two-dimensional test function in computer experiments. It is defined as follows:

$$f(x_1, x_2) = (x_2 - bx_1^2 + cx_1 - 6)^2 + 10(1-t)\cos(x_1) + 10. \tag{4.2}$$

Here, the parameters are defined as $b = \frac{5.1}{4\pi^2}$, $c = \frac{5}{\pi}$, and $t = \frac{1}{8\pi}$. The Branin Function is typically evaluated within the square region defined by $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$.

For the purpose of demonstration, we aim to construct a GP emulator for the Branin Function. To do this, we first generate 40 Latin hypercube samples within the specified square region using the following code:

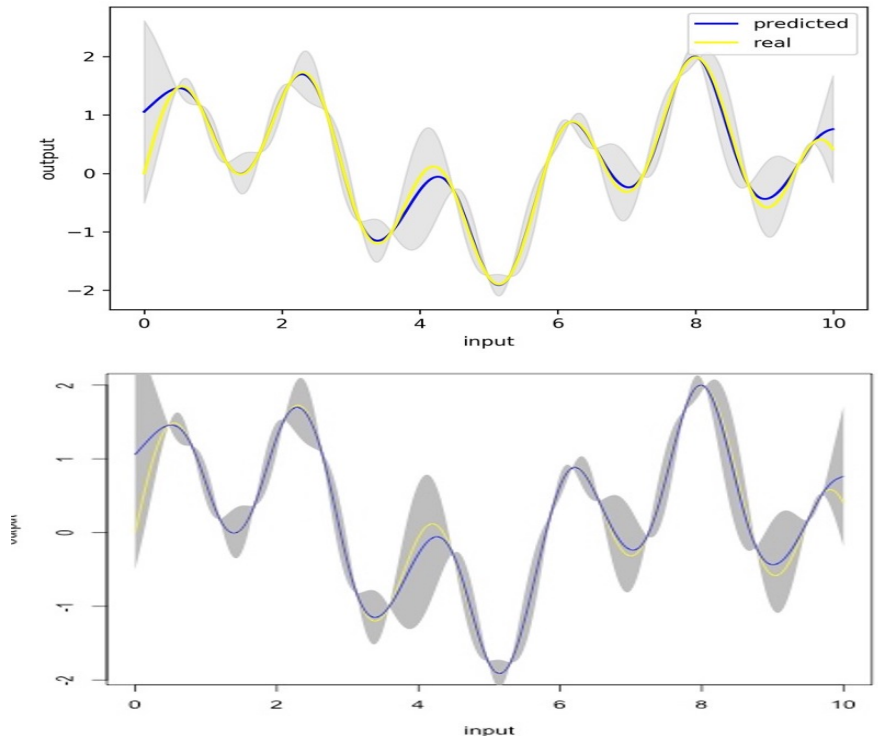

Figure 4.1: The predicted function value, actual function value, and their 95% prediction interval are shown in the graph. The predicted function value is represented by the blue solid line, the actual function value by the yellow solid line, and the shaded area corresponds to the 95% prediction interval. The upper figure is generated using PyRobustGaSP in `Python`, while the lower figure is created using RobustGaSP in `R`.

```
from PyRobustGaSP import PyRobustGaSP
from src.functions import *
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import qmc

# simulated branin function
def branin(x):
    x1 = x[0]
    x2 = x[1]
    t = 1/(8*np.pi)
    ans = 1*(x2-5.1/(np.pi**2*4)*x1**2+5/np.pi*x1-6)**2
```

```
    +10*(1-t)*np.cos(x1)+10
    return ans

def rescale(arr,x1_range,x2_range):
    # rescale the samples from  [0,1]*[0,1] to x1_range * x2_range
    arr[:,0] = x1_range[0] + arr[:,0] * (x1_range[1] - x1_range[0])
    arr[:,1] = x2_range[0] + arr[:,1] * (x2_range[1] - x2_range[0])
    return arr

# Create a PyRobustGaSP model instance
P_rgasp = PyRobustGaSP()

# Generate the training sample
sampler = qmc.LatinHypercube(d=2)
sample_input = sampler.random(n=40)
x1_range = (-5, 10)
x2_range = (0, 15)
sample_input = rescale(sample_input,x1_range,x2_range)
num_obs=sample_input.shape[0]
sample_output= np.zeros((num_obs,1))
for i in range(num_obs):
    sample_output[i,0]=branin(sample_input[i,:])
```

Next, we construct a GP emulator using these 40 samples under the default settings. We then evaluate its performance by calculating metrics such as the ratio of root mean square error to the standard deviation of the testing output. This evaluation is conducted on 400 random input points within the parameter space of the Branin function, as shown in the following code:

```
# Create a task for model training
task = P_rgasp.create_task(sample_input, sample_output)
```

```
# Fit a rgasp model using created task
model = P_rgasp.train_rgasp(task)

# Get testing input and output
dim_inputs=sample_input.shape[1]
num_testing_input = 400
testing_input = np.random.uniform(size =num_testing_input*
                                  dim_inputs).reshape(
   num_testing_input,-1)
testing_input = rescale(testing_input,x1_range,x2_range)
testing_output = np.zeros(num_testing_input)
for i in range(num_testing_input):
    testing_output[i]=branin(testing_input[i,:])

# Get the rgasp predict object
testing_predict = P_rgasp.predict_rgasp(model,
                  testing_input)

m_rmse=np.sqrt(np.mean( (testing_predict['mean']-testing_output)**2))#/
   np.std(testing_output[:,0])
print('RMSE/std is ', m_rmse/np.std(testing_output))
>> RMSE/std is  0.02755

prop_m = np.sum((testing_predict['lower95']<=testing_output) & (
   testing_predict['upper95']>=testing_output))/(testing_output.shape
   [0])
print("The Proportion of the test output covered by 95% CI is ",prop_m)
>> The Proportion of the test output covered by 95% CI is  0.965
```

Ackley Function with Noise

The following example presented here involves fitting the two-dimensional simulated Ackley function as shown in Equation (3.10), and illustrated in Figure 3.4, with the addition of random noise. We constructed the GP emulator with a nugget term based on 80 Latin hypercube design samples on $[-5, 5] \times [-5, 5]$ through the following code.

```
from PyRobustGaSP import PyRobustGaSP
from src.functions import *
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import qmc

# simulated ackley function
def ackley(x):
    ans = -20*np.exp(-0.2*np.sqrt(0.5*(x[0]**2 + x[1]**2)))
    -np.exp(0.5*(np.cos(np.pi*2*x[1]+np.pi*2*x[0])))+np.exp(1)+20
    return ans

# rescale the input space
def rescale(arr,x1_range,x2_range):
    # rescale the samples from  [0,1]*[0,1] to x1_range * x2_range
    arr[:,0] = x1_range[0] + arr[:,0] * (x1_range[1] - x1_range[0])
    arr[:,1] = x2_range[0] + arr[:,1] * (x2_range[1] - x2_range[0])
    return arr

# Create a PyRobustGaSP model instance
P_rgasp = PyRobustGaSP()

# Generate the training sample
sampler = qmc.LatinHypercube(d=2)
```

```
sample_input = sampler.random(n=80)
x1_range = (-5, 5)
x2_range = (-5, 5)
sample_input = rescale(sample_input,x1_range,x2_range)
num_obs=sample_input.shape[0]
sample_output= np.zeros((num_obs,1))
for i in range(num_obs):
    sample_output[i,0]=ackley(sample_input[i,:])

noise = np.random.normal(0, 1e-3, sample_output.shape)
sample_output = sample_output+noise

# Create a task for model training
task = P_rgasp.create_task(sample_input, sample_output, nugget_est=True
   )
# Fit a rgasp model using created task
model = P_rgasp.train_rgasp(task)
```

We then assess the performance of the trained model on 200 uniformly selected testing inputs, and the resulting plot is displayed in Figure 4.2. In this plot, the x-axis represents the predicted means generated by the trained model, while the y-axis represents the real output values from the Ackley function.

```
# Get testing input and output
dim_inputs=sample_input.shape[1]
num_testing_input = 200
testing_input = np.random.uniform(size =num_testing_input*
                                  dim_inputs).reshape(
   num_testing_input,-1)
testing_input = rescale(testing_input,x1_range,x2_range)
testing_output = np.zeros((num_testing_input,1))
```

```
for i in range(num_testing_input):
    testing_output[i,0]=ackley(testing_input[i,:])

# Get the rgasp predict object
testing_predict = P_rgasp.predict_rgasp(model,
                  testing_input)
ratio_rmse_std=np.sqrt(np.mean( (testing_predict['mean']-testing_output
   [:,0])**2))/np.std(testing_output)
print('RMSE/STD is ', ratio_rmse_std)
>> RMSE/STD is  0.06525564678324436

# Display the plot
fig, ax = plt.subplots()
# Plot the first line on the axes
ax.scatter(testing_predict['mean'],testing_output, label='Predicted',
           color = 'blue')
ax.set_xlabel('prediction')
ax.set_ylabel('real output')
ref_line_x = np.array([-8, -20])
ref_line_y = np.array([-8, -20])
ax.plot(ref_line_x, ref_line_y, linestyle='--', color='r')
plt.show()
```

An Example for Emulation of Potential Energy from Atomic Positions

In the following example, we will demonstrate how to perform energy emulation using the GPR model introduced in Section 2.5.1. The GPR model is trained on a dataset consisting of 400 alkane molecule simulations with transformed descriptors and potential energy values. We reserve 200 simulations as the testing dataset. The isotropic correlation function is applied, and the nugget parameter needs to be estimated in this

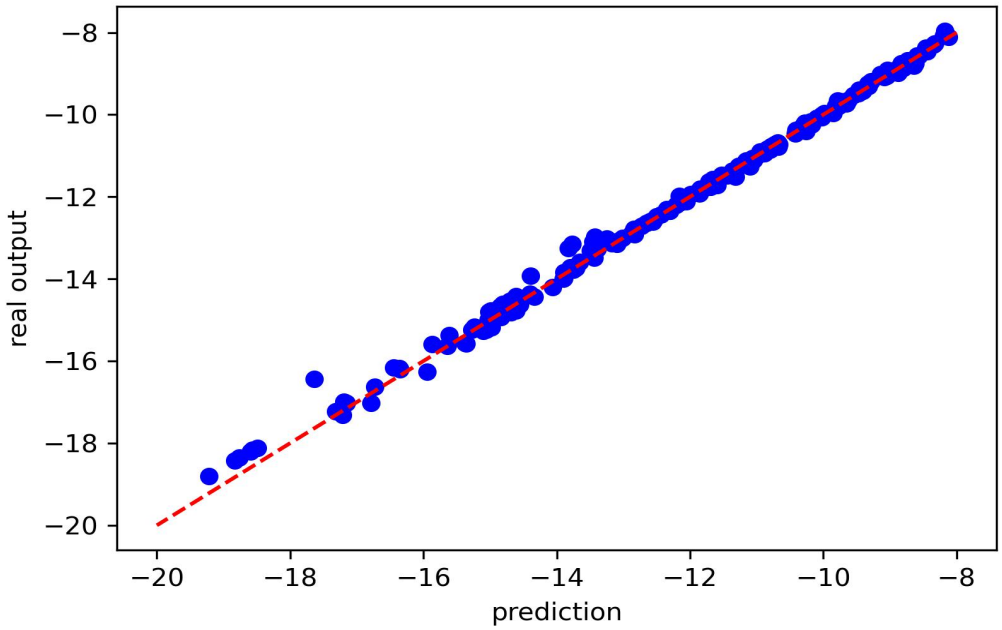

Figure 4.2: The predicted function value and actual function value of Ackley function.

example.

The printed output reveals that the ratio between the RMSE and the standard deviation of the testing energy is approximately 0.005. Furthermore, the 95% prediction interval effectively covers 100% of the real energy values, indicating the accuracy and reliability of the GPR model's predictions.

```
104 # Create a task for model training
105 task_energy = P_rgasp.create_task(X_train, Y_train, isotropic = True,
       nugget_est = True)
106 # Fit a rgasp model using created task
107 model = P_rgasp.train_rgasp(task_energy)
108 # Get the rgasp predict object
109 Y_predict = P_rgasp.predict_rgasp(model,
110                   testing_input)
111
```

```
# Show the performance: RMSE/STD
ratio_rmse_std = np.sqrt(np.sum((Y_predict['mean']- Y_test)**2)/n_test)
   /np.std(Y_test)
print('RMSE/STD is ', ratio_rmse_std)
>> RMSE/STD is  0.005083512072227473
# Show the proportion of samples coverd by 95% predictive interval
prop_m = np.sum((Y_predict['lower95']<=Y_test) &
                (Y_predict['upper95']>=Y_test))/(Y_test.shape[0])
print("The Proportion of the test output covered by 95% CI is ",
      round(prop_m,4)*100,'%')
>> The Proportion of the test output covered by 95% CI is  100.0 %
```

4.3 Emulation of Expensive Simulations with Massive Outputs

PP-GP is designed to tackle the challenge of emulating computer models that generate massive outputs. For instance, consider a simulator that produces electron density data over a straightforward 3D grid with 100 grid points on each axis of coordinates, resulting in a total of 10^6 data points. PP-GP is ideally suited for efficiently constructing fast emulators capable of handling such extensive electron density outputs.

In the example below, we aim to simulate electron densities of benzene molecules across a 3D grid with 50 grid points along each axis of coordinates. We use high-dimensional coefficients on this grid to represent the potential energy function as input data. In this emulation problem, both the input coefficients and the output electron densities for each observation have dimensions of 125,000. The following example demonstrates how to fit a PP-GP model to the given dataset using PyRobustGaSP. Firstly, we load the PyRobustGaSP module, followed by importing the input coefficients and output

electron densities for both the training and testing set. The training set comprises 50 simulated observations, while the testing set encompasses a separate set of 50 observations reserved for validation.

```
from PyRobustGaSP import PyRobustGaSP
from src.functions import *
import numpy as np

# Create a PyRobustGaSP model instance
P_rgasp = PyRobustGaSP()
# Load the training input and output
X_train = np.genfromtxt("./src/dataset/X_train.txt", skip_header=1)
    [:,1:]
Y_train =np.genfromtxt("./src/dataset/Y_train.txt", skip_header=1)
    [:,1:]
# Load the testing dataset
X_test =np.genfromtxt("./src/dataset/X_test.txt", skip_header=1)[:,1:]
Y_test = np.genfromtxt("./src/dataset/Y_test.txt", skip_header=1)[:,1:]
```

Subsequently, we proceed to create the PP-GP mode task using `create_task()`. Here, we have specified the utilization of an isotropic correlation function. Following this, we proceed with fitting a PP-GP model employing `train_ppgasp()` and proceed to compute the predictive distribution for the testing input via `predict_ppgasp()`.

```

# Create a task for model fit
task = P_rgasp.create_task(X_train, Y_train,isotropic=True)
# Fit a PP-GP model
model = P_rgasp.train_ppgasp(task)
# Get the prediction on testing input
testing_predict = P_rgasp.predict_ppgasp(model, X_test)
```

Similar to the prediction object from `predict_rgasp()` function, the predictive means of those testing inputs also can be accessed through the key `mean`, while the upper bound and lower bound of 95% predictive credible interval can be obtained through the `lower95` and `upper95` keys, respectively.

```
# Show the performance: RMSE/STD
m_rmse=np.sqrt(np.mean( (testing_predict['mean']-
                         Y_test)**2))/np.std(Y_test)
print('RMSE/STD is ', m_rmse)
>> RMSE/STD is  0.001024979836884022
# Show the proportion of samples coverd by 95% predictive interval
prop_m = np.sum((testing_predict['lower95']<=Y_test) &
                (testing_predict['upper95']>=Y_test))/(Y_test.shape[0]*
   Y_test.shape[1])
print("The Proportion of the test output covered by 95% CI is ", round(
   prop_m,4))
>> The Proportion of the test output covered by 95% CI is  0.9332
```

The graphs in Figure 4.3 compare the held-out electron density with the predicted electron density for a single benzene molecule. Figure 4.4 displays the uncertainty quantification, measured by the length of the 95% prediction interval for the same molecule.

4.4 An example of the AFF emulator

In this section, we provide an example to demonstrate how to generate AFF force fields using the code available in [113]. The dataset format employed in this package aligns with the MD17 dataset format [3]. To generate the AFF force fields, we need to first apply the function `aff.create_task()` to create a task, which specifies the training set, validation set, and some other settings such as the nugget parameter, whether to add uncertainty quantification and so on. Then, the function `aff.train()` returns a

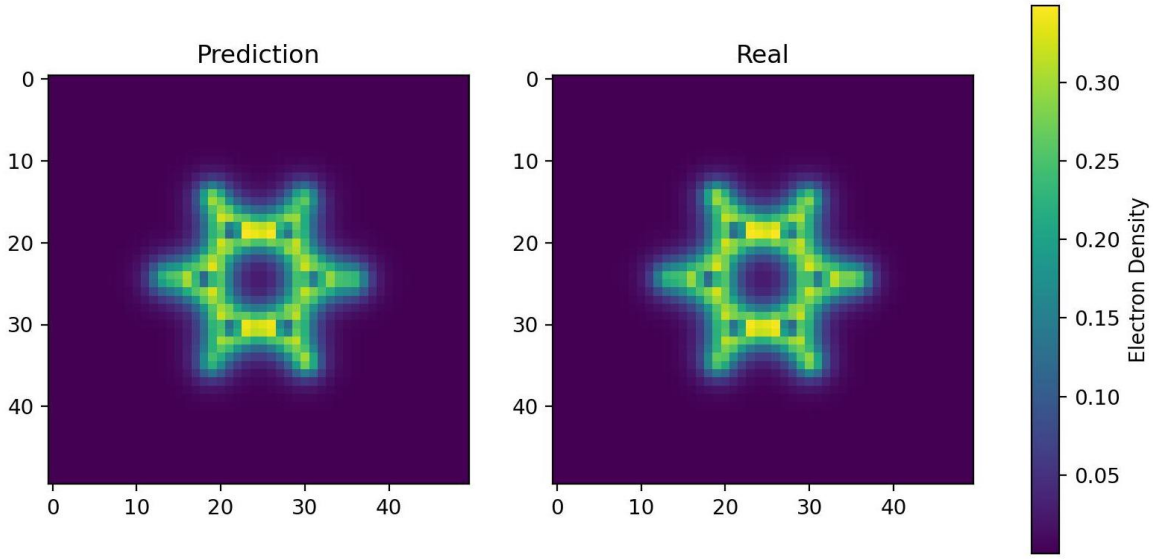

Figure 4.3: The comparison between PP-GP predicted electron density and simulated electron density on benzene molecule.

model fitted using the provided dataset. Predictions for new datasets are achieved using `aff.predict()`.

```
122 import numpy as np
123 from utils import aff
124
125 # load the dataset contains the geometry information and the force-
    fields
126 dataset=np.load('./dataset/uracil_dft.npz')
127
128 AFF_train=aff.AFFTrain()
129 n_train=100
130
131 # create the task file contains the training, validation and testing
    dataset
132 task=AFF_train.create_task(train_dataset=dataset,
```

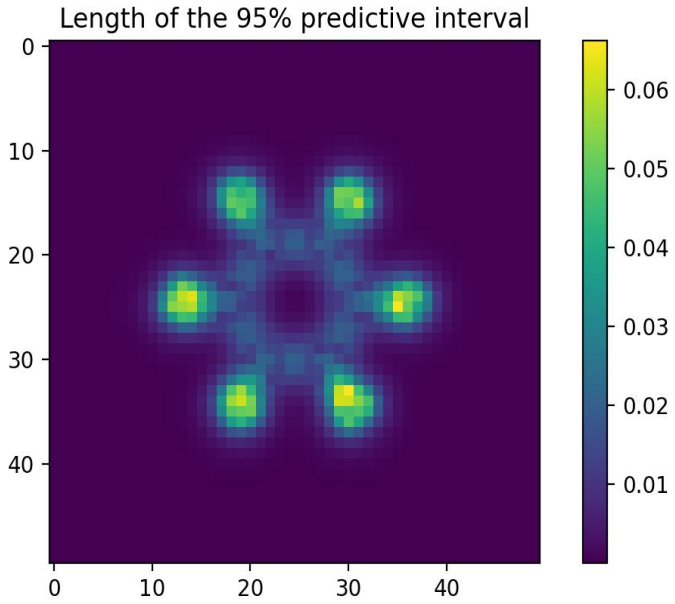

Figure 4.4: Length of the 95% predictive interval.

```
                              n_train = n_train ,
                              valid_dataset=dataset,
                              n_valid=50,
                              n_test=50,
                              lam = 1e-15,
                              uncertainty=False)

# train the model based on the training dataset
trained_model = AFF_train.train(task,sig_candid_F = np.arange(10,20,10)
    )

# predict the force-field using the trained_model
prediction=AFF_train.predict(task = task,
                               trained_model = trained_model,
                               R_test = task['R_test'][[0,1],:,:])
# force field prediction
```

```
148 predicted_force = prediction['predicted_force']
```

Chapter 5

Concluding remarks and future work

This thesis has considered the Gaussian process surrogate model and Bayesian optimization for learning the mapping between high-dimensional atomic positions and the potential energy surface, as well as the high-dimensional atomic force vectors of molecules.

Chapter 2 introduced the AFF emulator, a computational tool designed for efficient and accurate modeling of potential energy surfaces and molecular force fields in *ab initio* simulations, with the added benefit of providing valid uncertain quantification. While the theoretical framework of the gradient-based KRR and GP models such as GDML, sGDML, and FCHL, were already established, the challenge posed by the huge computational cost limited the applicability of these methods in emulating systems with a larger size of molecules. The AFF emulator was shown to overcome this computational challenge without compromising its accuracy. The efficient emulation of forces was hinged upon the fact that the similarity of atomic forces between permutationally equivalent atoms is high, whereas the correlation is small across different permutationally equivalent atom sets. By partitioning the atoms of a molecule into different atom sets, the AFF model can capture a large correlation of forces between PE atoms, thereby providing accurate predictions of atomic forces of the molecule at a new configuration with

less computational cost. Second, a new approach was developed to reduce the computational complexity for emulating the potential energy, compared to a joint model of energy and atomic forces of simulated configurations. Numerical results have shown predictions by the AFF emulator are more accurate than alternative approaches, given the same computational budget.

Chapter 3 introduced the concepts of inverse energy and atomic force design, both aimed at constructing the inverse mapping from potential energy surfaces and atomic force fields to 3D molecule structures. These approaches were developed to efficiently address high-dimensional optimization problems. Leveraging the accuracy and efficiency of the forward AFF emulator, the chapter demonstrated the good performance of inverse energy and force field design in discovering the equilibrium structure with minimum potential energy. These methods provide an innovative alternative for conducting geometry optimization through machine learning-based forward models, reducing the need for computationally expensive simulator evaluations.

Chapter 4 introduced the `Python` package called PyRobustGaSP, which provides a powerful and robust solution for Gaussian stochastic process emulation on large datasets. The chapter includes several example codes to illustrate its functionality. By offering the full capabilities of PP-GP and Robust GaSP to `Python` users, PyRobustGaSP extends these methods to a wider audience of researchers and practitioners familiar with the `Python` ecosystem.

5.1 Future Work in Predicting Large Systems with Active Subspace Methods

The primary challenge in applying machine learning methods to simulation problems and inverse molecule or material design in the fields of chemistry and material science remains the efficient handling of large systems while maintaining accuracy. For instance, most proteins consist of thousands of molecules, resulting in an extremely high-dimensional structural information space. Navigating or optimizing complex functions or forward models in such a high-dimensional space is a daunting task. One approach to overcoming the challenge of high dimensionality is dimensionality reduction, which aims to reduce the dimension of structural information without compromising valuable insights.

Many multivariate functions exhibit primary variations along a few specific directions in the input space. In cases where these directions align with the original coordinate axes, dimensionality reduction can be achieved through sensitivity analysis, as discussed in [114]. This approach involves identifying the variables that exert the most influence on model predictions, allowing for the construction of response surfaces that focus on the most influential factors.

However, these methods tend to perform poorly when the directions of variability do not align with the natural coordinates of the input space. In scenarios involving molecule structure information, the coordinate system is often randomly selected, making traditional sensitivity analysis methods less suitable. To address this challenge, [115] introduced the active subspace method. This technique first identifies the directions of the most significant variability using gradient evaluations and subsequently leverages these directions to construct a response surface within a lower-dimensional subspace of the original high-dimensional inputs.

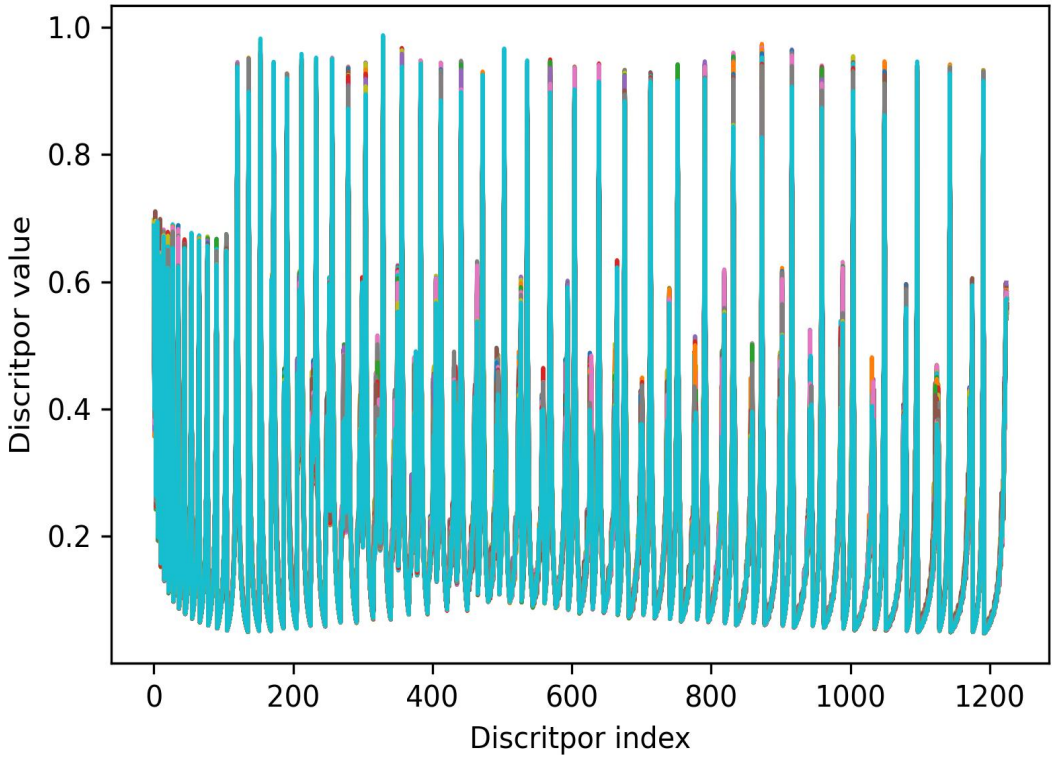

Figure 5.1: The descriptor values (y-axis) are plotted against the descriptor index (x-axis) for 200 alkane configurations. Each color represents an individual configuration.

The gradient information of the PES, force fields, is naturally accessible in the PES emulation problem. It inspires us to reduce the input dimension with the active subspace method. In the following paragraph, we will briefly show how to reduce the input dimension on the PES emulation problem.

Consider the potential energy function f of a system with N atoms with M continuous inputs:

$$f = f\left(\mathbf{D}(\mathbf{x})\right), \tag{5.1}$$

where $\mathbf{x} = (x_1, x_2, \cdots, x_{3N})$, and we still use the same vectorized inverse pairwise distance matrix $\mathbf{D}(\mathbf{x})$ as a descriptor to encode the input original Cartesian coordinate. We use

l_D to denote the dimensionality of $\mathbf{D}(\mathbf{x})$, and in fact $l_D = N(N-1)/2$. Denote the force vector, which is the gradient of f, by the column vector $\nabla_{\mathbf{x}} f(\mathbf{x}) = (\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_{3N}})^T$. Unlike the normal active subspace method working the original input $\mathbf{x}$, it is more suitable to treat descriptor $\mathbf{D}(\mathbf{x})$ as input with the full-dimensional space. This is due to the rotational and translational invariance discussed in Chapter 2. As shown in Figure 5.1, the descriptor values of different configurations of the same molecule share a similar pattern, making it easier to find a valid subspace. The gradient information is the key to finding the valid active subspace.

Given the fact that we have the observed $\nabla_{\mathbf{x}} f(\mathbf{x})$ and not $\nabla_{\mathbf{D}(\mathbf{x})} f(\mathbf{x})$. For simplicity, we use $\nabla_{\mathbf{D}} f$ to represent $\nabla_{\mathbf{D}(\mathbf{x})} f(\mathbf{x})$. According to the chain rule, $\nabla_{\mathbf{D}} f$ can be approximated through

$$\nabla_{\mathbf{D}} f \approx \nabla_{\mathbf{x}} f(\mathbf{x}) \left(\nabla_{\mathbf{x}} D(\mathbf{x})\right)^{\dagger}, \tag{5.2}$$

where $\left(\nabla_{\mathbf{x}} D(\mathbf{x})\right)^{\dagger}$ is the pseudoinverse of matrix $\nabla_{\mathbf{x}} D(\mathbf{x})$.

Define the $l_D \times l_D$ matrix $\mathbf{C}$ by

$$\mathbf{C} = \mathbb{E}[(\nabla_{\mathbf{D}} f)(\nabla_{\mathbf{D}} f)^T]. \tag{5.3}$$

The matrix $\mathbf{C}$ can be interpreted as the uncentered covariance of the gradient vector. The matrix $\mathbf{C}$ is symmetric and positive semidefinite and it can be decomposed into real eigenvalues as follows

$$\mathbf{C} = \mathbf{W}\boldsymbol{T}\mathbf{W}^T, \tag{5.4}$$

where $\boldsymbol{T}$ is a diagonal matrix with elements $(\tau_1, \cdots, \tau_{l_D})$ in descending order. The eigenvectors $\mathbf{W}$ define a rotation of $\mathbb{R}^{l_D}$ and, consequently, in the domain of function f. The components of the rotated coordinate system contain a set with greater value variation

and another set with smaller average variation, as shown below.

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{T}_1 & \\ & \boldsymbol{T}_2 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \mathbf{W}_1 & \mathbf{W}_2, \end{bmatrix} \tag{5.5}$$

where $\boldsymbol{T}_1 = diag(\tau_1, \cdots, \tau_k)$ with $k < l_D$, and $\mathbf{W}_1$ is an $l_D \times k$ matrix. The rotated coordinates $\mathbf{y} \in \mathbb{R}^k$ and $\mathbf{z} \in \mathbb{R}^{l_D - k}$ are defined as

$$\mathbf{y} = \mathbf{W}_1^T \mathbf{D}(\mathbf{x}), \quad \mathbf{z} = \mathbf{W}_2^T \mathbf{D}(\mathbf{x}). \tag{5.6}$$

According to the Lemma 2.2 in [115], the mean-squared gradients of f with respect to the coordinates $\mathbf{y}$ and $\mathbf{z}$ satisfy

$$\mathbb{E}[(\nabla_{\mathbf{y}} f)^T (\nabla_{\mathbf{y}} f)] = \tau_1 + \cdots + \tau_k \tag{5.7}$$

$$\mathbb{E}[(\nabla_{\mathbf{z}} f)^T (\nabla_{\mathbf{z}} f)] = \tau_{k+1} + \cdots + \tau_{l_D}. \tag{5.8}$$

The active subspace is actually motivated by the equations above, which indicate that the function f exhibits greater average variation along the directions defined by the columns of $\mathbf{W}_1$ than along the other directions defined by columns of $\mathbf{W}_2$. In practice, with samples of the gradient observations, the matrix $\mathbf{C}$ is approximated through

$$\mathbf{C} \approx \bar{\mathbf{C}} = \frac{1}{M} \sum_{j=1}^{M} (\nabla_{\mathbf{D}} f(\mathbf{x}_j))(\nabla_{\mathbf{D}} f(\mathbf{x}_j))^T, \tag{5.9}$$

where $\nabla_{\mathbf{D}} f(\mathbf{x}_j)$ can be approximated by Equation (5.2).

As shown in Figure 5.2, in our simulated alkane example, the sum of the first one hundred eigenvalues in matrix $\bar{\mathbf{C}}$ already accounts for 99.999% of the sum of all eigenvalues. Compared to the original input with a dimension of 1225, the emulator, based

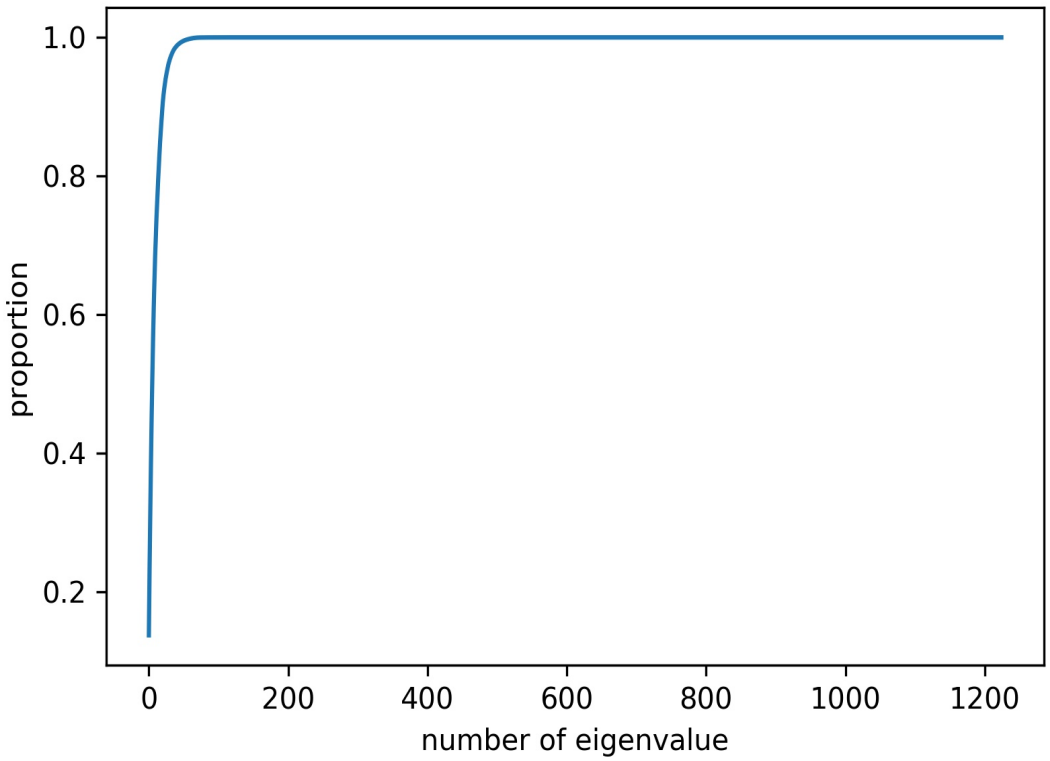

Figure 5.2: The proportion of first i-th eigenvalue over the summation of all eigenvalues (y-axis) are plotted against the number of eigenvalues (x-axis) for alkane simulated configurations.

on the active subspace $\mathbf{y}$, reduces the original dimension by 91.8% without sacrificing accuracy. The graph in Figure 5.3 provides a comparison of potential energy predictions between two GP models. One model utilizes the full descriptor space with a dimension of 1225, while the other uses a reduced subspace with a dimension of 50, both trained on the same set of 400 alkane simulations and tested on the same hold-out set of 200 simulations. The results demonstrate that both models achieve a similar level of prediction accuracy. These promising results indicate that, in large-scale simulation calculations with gradient observations, it is possible to significantly reduce the dimension of large systems using the above approach. This is a significant step towards the goal of inverse molecule or

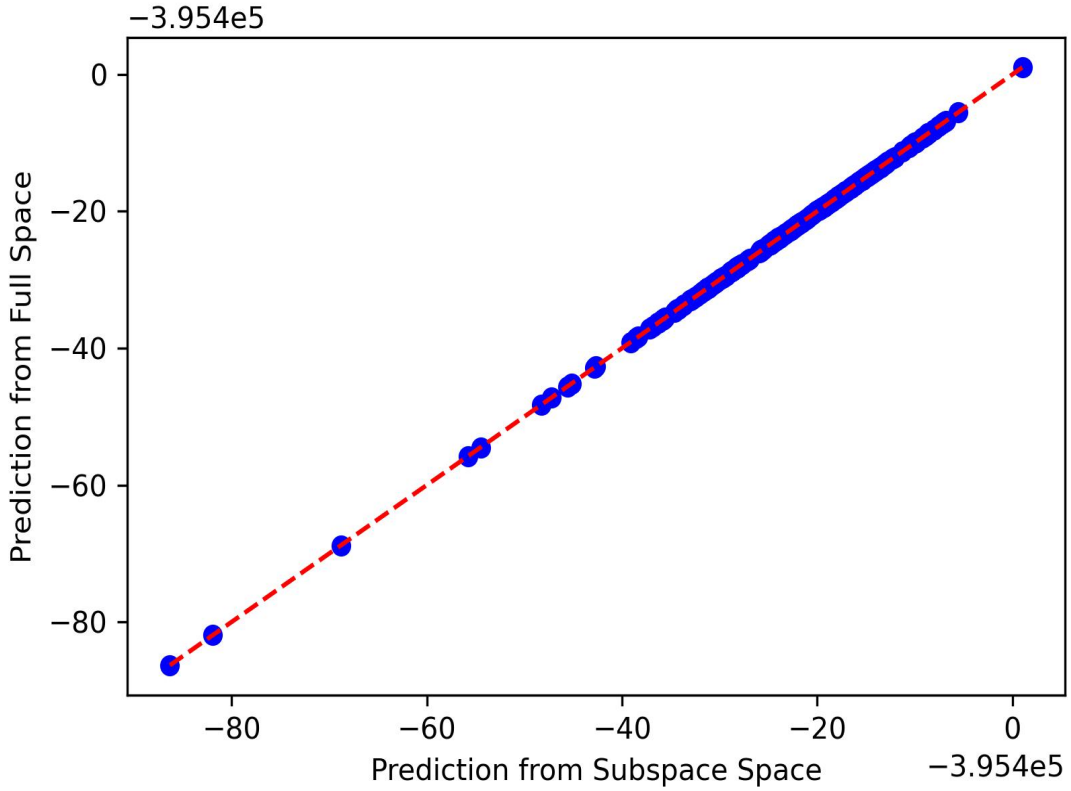

Figure 5.3: The comparison of potential energy predictions between the GP model using full space (dimension 1225) and reduced active subspace (dimension 50) for alkane simulated configurations. The y-axis displays prediction values from the model with the full descriptor space, while the x-axis presents prediction values from the model using the reduced subspace.

system design, where a stable and computationally efficient approach is needed for future advancements.

5.2 Future Work on Computation Reduction

Based on our discussion in Chapter 2, Gaussian processes have emerged as a valuable tool for emulating calculations due to their high accuracy, especially in the context of molecular dynamics simulations. However, one major limitation of GP inference is

its computational infeasibility for large systems. One potential avenue for applying GP emulation to large systems is to explore scalable GP inference techniques, such as the Vecchia GP approximations introduced in [116, 117, 118]. Initially designed for spatial inputs, this approximation method has recently been extended to nonspatial inputs. Vecchia's approximation hinges on an exact decomposition of the joint density $p(\mathbf{y}) = \prod_{i=1}^{M} P(y_i \mid y_1, \cdots, y_{i=1})$ into a product of univariate conditional densities. It approximates this joint density as follows:

$$\hat{p}(\mathbf{y}) = \prod_{i=1}^{M} p\left(y_i \mid \mathbf{y}_{c(i)}\right). \tag{5.10}$$

Here, $c(i) \subset 1, \cdots, i-1$ represents a conditioning index of size $|c(i)| = min(m, i-1)$, where m is a relatively small conditioning set size $m \ll M$. It's worth noting that the computation of each $p(y_i \mid \mathbf{y}_{c(i)})$ involves the computation complexity $\mathcal{O}(m^3)$ not $\mathcal{O}(M^3)$. Vecchia's approximation boasts several advantageous properties, including the fact that the implied joint distribution $\hat{p}(\mathbf{y})$ remains multivariate Gaussian. Additionally, the Cholesky factorization of the correlation matrix inversion becomes highly sparse. It's essential to highlight that, unlike local GP approximations, the Vecchia approach provides a global approximation to the underlying model. The accuracy of this approximation depends on the choice of variable ordering and conditioning sets. For future work in the context of molecule structure information, it may be worthwhile to develop approximate algorithms tailored to this setting. Such efforts could significantly reduce the computational cost of applying GPs to MD calculations, particularly for quantities like force vectors.

Appendix A

Appendix Title

A.1 Fast Predictions of Potential Energies in Batches

This section discusses the efficient way to calculate the inversion of the covariance matrix in the AFF model on energy prediction. We achieve the reduced computational cost by predicting the molecules' energies in batches rather than the energy for one configuration each time. Let b be the batch size, and $\mathbf{x}^{b*} = [\mathbf{x}_1^*, \cdots, \mathbf{x}_b^*]$ be b configurations of a molecular structure. For predicting the energy of a new configuration of this molecular structure, we need to calculate the inversion of $\mathbf{\Sigma}_{sub}$, where $\mathbf{\Sigma}_{sub} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$, with $\mathbf{A} = \mathbf{K}_{\mathbf{X},\mathbf{X}} + \lambda \mathbf{I}_M$ being a $M \times M$ matrix, $\mathbf{B} = -\mathbf{J}_{\mathbf{X},\mathbf{x}^{b*}}$ being a $M \times 3Nb$ matrix, $\mathbf{C} = -\mathbf{J}_{\mathbf{x}^{b*},\mathbf{X}}$ being a $3Nb \times M$ matrix, and $\mathbf{D} = \mathbf{R}_{\mathbf{x}^{b*},\mathbf{x}^{b*}} + \lambda \mathbf{I}_{3Nb}$ being a $3Nb \times 3Nb$ matrix. Note that the sub-covariance matrix of training energy samples $\mathbf{A}$ is the same among all different batches in prediction. Thus we need to invert $\mathbf{A}$ once and by applying

the block matrix inversion for $\mathbf{\Sigma}_{sub}$, we have:

$$\mathbf{\Sigma}_{sub}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\mathbf{D}^{*-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{D}^{*-1} \\ -\mathbf{D}^{*-1}\mathbf{C}\mathbf{A}^{-1} & \mathbf{D}^{*-1} \end{bmatrix}, \tag{A.1}$$

where $\mathbf{D}^* = \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$. Accordingly, for each batch samples, we just need to do a matrix inversion on a $3Nb \times 3Nb$ matrix $\mathbf{D}^*$, which is much faster than inverting the entire covariance matrix $\mathbf{\Sigma}_{sub}$ for each batch sample.

A.2 Predictive Distribution of Potential Energy

To simplify the notation, we would use the batch size $b = 1$ in this section. Conditional on simulated energies in the training dataset $\mathbf{E}$, and the atomic force $\mathbf{F}(\mathbf{x}^*)$ at the new configuration $\mathbf{x}^*$, and the estimated parameters $\hat{\boldsymbol{\theta}} = [\hat{m}, \hat{\sigma}^2, \hat{\gamma}, \hat{\lambda}]$, the conditional distribution of the energy at this configuration follows

$$E(\mathbf{x}^*) \mid \mathbf{E}, \mathbf{F}(\mathbf{x}^*), \hat{\boldsymbol{\theta}} \sim \mathcal{MN}\left(\mu_E^*(\mathbf{x}^*), \hat{\sigma}^2 K_E^{**}(\mathbf{x}^*, \mathbf{x}^*)\right), \tag{A.2}$$

where the conditional mean follows

$$\mu_E^*(\mathbf{x}^*) = \hat{m} + \begin{bmatrix} \mathbf{K}_{\mathbf{x}^*,\mathbf{X}} & -\mathbf{J}_{\mathbf{x}^*,\mathbf{x}^*}^T \end{bmatrix} \mathbf{\Sigma}_{sub}^{-1} \begin{bmatrix} \mathbf{E} - \hat{m}\mathbf{1}_M \\ \mathbf{F}(\mathbf{x}^*) \end{bmatrix},$$

with $\mathbf{\Sigma}_{sub} = \begin{bmatrix} \mathbf{K}_{\mathbf{X},\mathbf{X}} & -\mathbf{J}_{\mathbf{X},\mathbf{x}^*} \\ -\mathbf{J}_{\mathbf{x}^*,\mathbf{X}} & \mathbf{R}_{\mathbf{x}^*,\mathbf{x}^*} \end{bmatrix} + \lambda \mathbf{I}_{3N+M}$, and the conditional variance follows

$$\mathbf{K}_E^{**}(\mathbf{x}^*, \mathbf{x}^*) = K_{\mathbf{x}^*,\mathbf{x}^*} - \begin{bmatrix} \mathbf{K}_{\mathbf{x}^*,\mathbf{X}} & -\mathbf{J}_{\mathbf{x}^*,\mathbf{x}^*}^T \end{bmatrix} \mathbf{\Sigma}_{sub}^{-1} \begin{bmatrix} \mathbf{K}_{\mathbf{x}^*,\mathbf{X}}^T \\ -\mathbf{J}_{\mathbf{x}^*,\mathbf{x}^*} \end{bmatrix}.$$

Note that we do not observe $\mathbf{F}(\mathbf{x}^*)$ and thus Equation (A.2) cannot be directly used for predicting energy at the new configuration $\mathbf{x}^*$. Given the energy of training set $\mathbf{E}$ and the atomic force of training set $\mathbf{F}$, we use the total expectation to integrate the unobserved force vector by its predictive distribution:

$$\begin{aligned} \hat{E}(\mathbf{x}^*) &= \mathbb{E}[E(\mathbf{x}^*) \mid \mathbf{E}, \mathbf{F}] \\ &= \mathbb{E}[\mathbb{E}[E(\mathbf{x}^*) \mid \mathbf{E}, \mathbf{F}, \mathbf{F}(\mathbf{x}^*)]] \\ &\doteq \mathbb{E}[\mathbb{E}[E(\mathbf{x}^*) \mid \mathbf{E}, \mathbf{F}(\mathbf{x}^*)] \mid \mathbf{F}] \\ &= \mathbb{E}[\mu_E^*(\mathbf{x}^*) \mid \mathbf{E}, \mathbf{F}], \end{aligned}$$

where $\doteq$ denotes the approximation of $\mathbb{E}[E(\mathbf{x}^*) \mid \mathbf{E}, \mathbf{F}, \mathbf{F}(\mathbf{x}^*)]$ by $\mathbb{E}[E(\mathbf{x}^*)|\mathbf{E}, \mathbf{F}(\mathbf{x}^*)]$, which is equivalent to assume that given $(\mathbf{E}, \mathbf{F}(\mathbf{x}^*))$, $E(\mathbf{x}^*)$ is independent of the rest of force vectors in simulated configurations. Plugging the predictive mean $\hat{\mathbf{F}}(\mathbf{x}^*)$ from the above equation to replace $\mathbf{F}(\mathbf{x}^*)$ in $\mu_E^*(\mathbf{x}^*)$, we approximate the predictive mean of energy for $\mathbb{E}[\hat{E}(\mathbf{x}^*) \mid \mathbf{E}, \mathbf{F}]$ by $\hat{E}(\mathbf{x}^*)$ with the following expression:

$$\hat{E}(\mathbf{x}^*) = \hat{m} + \begin{bmatrix} \mathbf{K}_{\mathbf{x}^*,\mathbf{X}} & -\mathbf{J}_{\mathbf{x}^*,\mathbf{x}^*}^T \end{bmatrix} \mathbf{\Sigma}_{sub}^{-1} \begin{bmatrix} \mathbf{E} - \hat{m}\mathbf{1}_M \\ \hat{\mathbf{F}}(\mathbf{x}^*) \end{bmatrix}.$$

The predictive variance in Equation (2.22) can be computed by properties of multivariate

normal distributions:

$$K_E^*(\mathbf{x}^*, \mathbf{x}^*) = K_{\mathbf{x}^*,\mathbf{x}^*} - \begin{bmatrix} \mathbf{K}_{\mathbf{x}^*,\mathbf{X}} & -\mathbf{J}_{\mathbf{x}^*,\mathbf{x}^*}^T \end{bmatrix} \boldsymbol{\Sigma}_{sub}^{-1} \begin{bmatrix} \mathbf{K}_{\mathbf{x}^*,\mathbf{X}}^T \\ -\mathbf{J}_{\mathbf{x}^*,\mathbf{x}^*} \end{bmatrix}.$$

Let $\mathbf{W}_1$ and $\mathbf{W}_2$ be the first $1 \times M$ block matrix and the latter $1 \times 3N$ block matrix of $\begin{bmatrix} \mathbf{K}_{\mathbf{x}^*,\mathbf{X}} & -\mathbf{J}_{\mathbf{x}^*,\mathbf{x}^*}^T \end{bmatrix} \boldsymbol{\Sigma}_{sub}^{-1}$, respectively, and $\hat{\mathbf{F}}(\mathbf{x}^*) = \boldsymbol{\omega}_F \mathbf{F}$, where $\boldsymbol{\omega}_F$ follows from Equation (2.14). We can also write the $\hat{E}(\mathbf{x}^*)$ as the weighted average value of $\mathbf{E}$ and $\hat{\mathbf{F}}(\mathbf{x}^*)$:

$$\hat{E}(\mathbf{x}^*) = \boldsymbol{\omega}_E^* \mathbf{E} + \boldsymbol{\omega}_F^* \mathbf{F}, \tag{A.3}$$

where

$$\boldsymbol{\omega}_E^* = (1 - \mathbf{W}_1 \mathbf{1}_M)(\mathbf{1}_M^T \mathbf{K}_{\mathbf{X},\mathbf{X}}^{-1} \mathbf{1}_M)^{-1} \mathbf{1}_M^T K_{\mathbf{X},\mathbf{X}}^{-1} + \mathbf{W}_1,$$

and $\boldsymbol{\omega}_F^* = \mathbf{W}_2 \boldsymbol{\omega}_F$.

A.3 Simulation Details

In Chapter 2, in addition to molecules available from the MD17 dataset, the force and energy of additional molecules with more atoms and complicated structures are generated in this work. AIMD simulation is performed via Q-Chem to generate highly accurate molecular force and energy to benchmark AFF and other machine learning force fields. In this work, all AIMD simulations are carried in NVT ensemble with timestep of 1 fs at room temperature (300 K). All the calculations were performed at the level of Perdew-Burke-Ernzerhof(PBE)/6-31G(d,p). vdW interactions are taken into account by using TS-vdW method. The Nosé-Hoover thermostat is used to control the temperature. In Chapter 3, all simulations, including the evaluation of energy and force fields, were

conducted using the same settings as demonstrated above through Q-Chem.

Bibliography

[1] F. Brockherde, L. Vogt, L. Li, M. E. Tuckerman, K. Burke, and K.-R. Müller, *Bypassing the kohn-sham equations with machine learning*, *Nature communications* **8** (2017), no. 1 872.

[2] A. S. Christensen, L. A. Bratholm, F. A. Faber, and O. Anatole von Lilienfeld, *FCHL revisited: Faster and more accurate quantum machine learning*, *The Journal of Chemical Physics* **152** (2020), no. 4 044107.

[3] S. Chmiela, H. E. Sauceda, K.-R. Müller, and A. Tkatchenko, *Towards exact molecular dynamics simulations with machine-learned force fields*, *Nature communications* **9** (2018), no. 1 1–10.

[4] 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*, *Science advances* **3** (2017), no. 5 e1603015.

[5] A. P. Bartók, R. Kondor, and G. Csányi, *On representing chemical environments*, *Physical Review B* **87** (2013), no. 18 184115.

[6] A. P. Bartók and G. Csányi, *Gaussian approximation potentials: A brief tutorial introduction*, *International Journal of Quantum Chemistry* **115** (2015), no. 16 1051–1057.

[7] 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** (2018), no. 24 241722.

[8] L. Zhang, J. Han, H. Wang, R. Car, and E. Weinan, *Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics*, *Physical review letters* **120** (2018), no. 14 143001.

[9] E. Mansimov, O. Mahmood, S. Kang, and K. Cho, *Molecular geometry prediction using a deep generative graph neural network*, *Scientific reports* **9** (2019), no. 1 20381.

[10] N. Gebauer, M. Gastegger, and K. Schütt, *Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules*, *Advances in neural information processing systems* **32** (2019).

[11] 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** (2021), no. 16 10073–10141.

[12] A. P. Bartók, J. Kermode, N. Bernstein, and G. Csányi, *Machine learning a general-purpose interatomic potential for silicon*, *Physical Review X* **8** (2018), no. 4 041048.

[13] J. Cui and R. V. Krems, *Gaussian process model for collision dynamics of complex molecules*, *Physical review letters* **115** (2015), no. 7 073202.

[14] S. Conti, J. P. Gosling, J. E. Oakley, and A. O'Hagan, *Gaussian process emulation of dynamic computer codes*, *Biometrika* **96** (2009), no. 3 663–676.

[15] R. M. Hathout and A. A. Metwally, *Towards better modeling of drug-loading in solid lipid nanoparticles: Molecular dynamics, docking experiments and Gaussian processes machine learning*, *European Journal of Pharmaceutics and Biopharmaceutics* **108** (2016) 262–268.

[16] H. Sugisawa, T. Ida, and R. V. Krems, *Gaussian process model of 51-dimensional potential energy surface for protonated imidazole dimer*, *The Journal of Chemical Physics* **153** (2020), no. 11.

[17] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, *Neural message passing for quantum chemistry*, in *International conference on machine learning*, pp. 1263–1272, PMLR, 2017.

[18] L. Zepeda-Núñez, Y. Chen, J. Zhang, W. Jia, L. Zhang, and L. Lin, *Deep density: circumventing the kohn-sham equations via symmetry preserving neural networks*, *Journal of Computational Physics* **443** (2021) 110523.

[19] L. Zhang, J. Han, H. Wang, W. Saidi, R. Car, *et. al.*, *End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems*, *Advances in neural information processing systems* **31** (2018).

[20] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky, *E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials*, *Nature communications* **13** (2022), no. 1 2453.

[21] J. Sacks, S. B. Schiller, and W. J. Welch, *Designs for computer experiments*, *Technometrics* **31** (1989), no. 1 41–47.

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

[23] 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** (2012), no. 5 058301.

[24] J. Sacks, W. J. Welch, T. J. Mitchell, H. P. Wynn, *et. al.*, *Design and analysis of computer experiments*, *Statistical science* **4** (1989), no. 4 409–423.

[25] M. J. Bayarri, J. O. Berger, R. Paulo, J. Sacks, J. A. Cafeo, J. Cavendish, C.-H. Lin, and J. Tu, *A framework for validation of computer models*, *Technometrics* **49** (2007), no. 2 138–154.

[26] D. Higdon, J. Gattiker, B. Williams, and M. Rightley, *Computer model calibration using high-dimensional output*, *Journal of the American Statistical Association* **103** (2008), no. 482 570–583.

[27] J. Muré, *Propriety of the reference posterior distribution in Gaussian process modeling*, *The Annals of Statistics* **49** (2021), no. 4 2356–2377.

[28] E. T. Spiller, M. Bayarri, J. O. Berger, E. S. Calder, A. K. Patra, E. B. Pitman, and R. L. Wolpert, *Automating emulator construction for geophysical hazard maps*, *SIAM/ASA Journal on Uncertainty Quantification* **2** (2014), no. 1 126–152.

[29] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.

[30] J. Nocedal, *Updating quasi-newton matrices with limited storage*, *Mathematics of computation* **35** (1980), no. 151 773–782.

[31] D. C. Liu and J. Nocedal, *On the limited memory bfgs method for large scale optimization*, *Mathematical programming* **45** (1989), no. 1-3 503–528.

[32] R. Li and A. Sudjianto, *Analysis of computer experiments using penalized likelihood in gaussian kriging models*, *Technometrics* **47** (2005), no. 2.

[33] P. Ranjan, R. Haynes, and R. Karsten, *A computationally stable approach to gaussian process interpolation of deterministic computer simulation data*, *Technometrics* **53** (2011), no. 4 366–378.

[34] M. Gu, X. Wang, and J. O. Berger, *Robust Gaussian stochastic process emulation*, *The Annals of Statistics* **46** (2018), no. 6A 3038–3066.

[35] J. O. Berger, V. De Oliveira, and B. Sansó, *Objective Bayesian analysis of spatially correlated data*, *Journal of the American Statistical Association* **96** (2001), no. 456 1361–1374.

[36] M. J. Bayarri, J. O. Berger, E. S. Calder, K. Dalbey, S. Lunagomez, A. K. Patra, E. B. Pitman, E. T. Spiller, and R. L. Wolpert, *Using statistical and computer models to quantify volcanic hazards*, *Technometrics* **51** (2009), no. 4 402–413.

[37] C. Ren, D. Sun, and C. He, *Objective bayesian analysis for a spatial model with nugget effects*, *Journal of Statistical Planning and Inference* **142** (2012), no. 7 1933–1946.

[38] H. Kazianka and J. Pilz, *Objective Bayesian analysis of spatial data with uncertain nugget and range parameters*, *Canadian Journal of Statistics* **40** (2012), no. 2 304–327.

[39] R. Paulo, *Default priors for Gaussian processes*, *Annals of statistics* **33** (2005), no. 2 556–582.

[40] M. Gu and J. O. Berger, *Parallel partial Gaussian process emulation for computer models with massive output*, *Annals of Applied Statistics* **10** (2016), no. 3 1317–1347.

[41] M. Gu, *Jointly robust prior for Gaussian stochastic process in emulation, calibration and variable selection*, *Bayesian Analysis* **14** (2018), no. 1.

[42] C. E. Rasmussen, *Gaussian processes for machine learning*. MIT Press, 2006.

[43] C. Linkletter, D. Bingham, N. Hengartner, D. Higdon, and Q. Y. Kenny, *Variable selection for gaussian process models in computer experiments*, *Technometrics* **48** (2006), no. 4 478–490.

[44] M. Gu, J. Palomo, and J. O. Berger, *RobustGaSP: Robust Gaussian Stochastic Process Emulation in R*, *The R Journal* **11** (2019), no. 1 112–136.

[45] M. Gu, *RobustGaSP: Robust Gaussian Process Emulation In MATLAB*, August, 2019. Zenodo, https://doi.org/10.5281/zenodo.3370575.

[46] "Python package: PRobustGP." `https://github.com/HaoLiHL/PyRobustGaSP`. Accessed: 2023-05-30.

[47] G. Wahba, *Spline models for observational data*. SIAM, 1990.

[48] M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur, *Gaussian processes and kernel methods: A review on connections and equivalences*, *arXiv preprint arXiv:1807.02582* (2018).

[49] H. Wendland, *Scattered data approximation*, vol. 17. Cambridge university press, 2004.

[50] M. Gu, F. Xie, and L. Wang, *A theoretical framework of the scaled Gaussian stochastic process in prediction and calibration*, *SIAM/ASA Journal on Uncertainty Quantification* **10** (2022), no. 4 1435–1460.

[51] M. Gu and L. Wang, *Scaled Gaussian stochastic process for computer model calibration and prediction*, *SIAM/ASA Journal on Uncertainty Quantification* **6** (2018), no. 4 1555–1583.

[52] A. Patra, A. Bauer, C. Nichita, E. B. Pitman, M. Sheridan, M. Bursik, B. Rupp, A. Webber, A. Stinton, L. Namikawa, and C. Renschler, *Parallel adaptive numerical simulation of dry avalanches over natural terrain*, *Journal of Volcanology and Geothermal Research* **139** (2005), no. 1 1–21.

[53] S. Conti and A. O'Hagan, *Bayesian emulation of complex multi-output and dynamic computer models*, *Journal of statistical planning and inference* **140** (2010), no. 3 640–651.

[54] R. Paulo, G. García-Donato, and J. Palomo, *Calibration of computer models with multivariate output*, *Computational Statistics and Data Analysis* **56** (2012), no. 12 3959–3974.

[55] M. Gu and W. Shen, *Generalized probabilistic principal component analysis of correlated data*, *Journal of Machine Learning Research* **21** (2020), no. 13.

[56] M. Gu, Y. Lin, V. C. Lee, and D. Qiu, *Probabilistic forecast of nonlinear dynamical systems with uncertainty quantification*, *Accepted in Physica D: nonlinear phenomena, arXiv preprint arXiv:2305.08942* (2023).

[57] S. A. Hollingsworth and R. O. Dror, *Molecular dynamics simulation for all*, *Neuron* **99** (2018), no. 6 1129–1143.

[58] D. Lu, H. Wang, M. Chen, L. Lin, R. Car, E. Weinan, W. Jia, and L. Zhang, *86 pflops deep potential molecular dynamics simulation of 100 million atoms with ab initio accuracy*, *Computer Physics Communications* **259** (2021) 107624.

[59] E. Snelson and Z. Ghahramani, *Sparse Gaussian processes using pseudo-inputs*, *Advances in neural information processing systems* **18** (2006) 1257.

[60] H. Wang, L. Zhang, J. Han, and E. Weinan, *Deepmd-kit: A deep learning package for many-body potential energy representation and molecular dynamics*, *Computer Physics Communications* **228** (2018) 178–184.

[61] S. Chmiela, H. E. Sauceda, I. Poltavsky, K.-R. Müller, and A. Tkatchenko, *sGDML: Constructing accurate and data efficient molecular force fields using machine learning*, *Computer Physics Communications* **240** (2019) 38–45.

[62] N. Cressie and G. Johannesson, *Fixed rank kriging for very large spatial data sets*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **70** (2008), no. 1 209–226.

[63] C. G. Kaufman, M. J. Schervish, and D. W. Nychka, *Covariance tapering for likelihood-based estimation in large spatial data sets*, *Journal of the American Statistical Association* **103** (2008), no. 484 1545–1555.

[64] A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand, *Hierarchical nearest-neighbor gaussian process models for large geostatistical datasets*, *Journal of the American Statistical Association* **111** (2016), no. 514 800–812.

[65] F. Lindgren, H. Rue, and J. Lindström, *An explicit link between gaussian fields and gaussian markov random fields: the stochastic partial differential equation approach*, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **73** (2011), no. 4 423–498.

[66] R. B. Gramacy and D. W. Apley, *Local Gaussian process approximation for large computer experiments*, *Journal of Computational and Graphical Statistics* **24** (2015), no. 2 561–578.

[67] F. de Roos, A. Gessner, and P. Hennig, *High-dimensional Gaussian process inference with derivatives*, *arXiv preprint arXiv:2102.07542* (2021).

[68] D. Eriksson, K. Dong, E. H. Lee, D. Bindel, and A. G. Wilson, *Scaling Gaussian process regression with derivatives*, *arXiv preprint arXiv:1810.12283* (2018).

[69] Z. Xie and J. M. Bowman, *Permutationally invariant polynomial basis for molecular energy surface fitting via monomial symmetrization*, *Journal of Chemical Theory and Computation* **6** (2010), no. 1 26–34.

[70] B. Jiang and H. Guo, *Permutation invariant polynomial neural network approach to fitting potential energy surfaces. iii. molecule-surface interactions*, *The Journal of Chemical Physics* **141** (2014), no. 3 034109.

[71] D. Koner and M. Meuwly, *Permutationally invariant, reproducing kernel-based potential energy surfaces for polyatomic molecules: From formaldehyde to acetone*, *Journal of Chemical Theory and Computation* **16** (2020), no. 9 5474–5484.

[72] S. Chmiela, *Towards exact molecular dynamics simulations with invariant machine-learned models*. Technische Universitaet Berlin (Germany), 2019.

[73] S. Umeyama, *An eigendecomposition approach to weighted graph matching problems*, *IEEE transactions on pattern analysis and machine intelligence* **10** (1988), no. 5 695–703.

[74] O. Roustant, D. Ginsbourger, and Y. Deville, *Dicekriging, diceoptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization*, .

[75] E. Snelson and Z. Ghahramani, *Sparse Gaussian processes using pseudo-inputs*, *Advances in neural information processing systems* **18** (2005).

[76] J. Quinonero-Candela and C. E. Rasmussen, *A unifying view of sparse approximate Gaussian process regression*, *The Journal of Machine Learning Research* **6** (2005) 1939–1959.

[77] M. W. Mahoney and P. Drineas, *Cur matrix decompositions for improved data analysis*, *Proceedings of the National Academy of Sciences* **106** (2009), no. 3 697–702.

[78] Y. Zhang, C. Hu, and B. Jiang, *Embedded atom neural network potentials: Efficient and accurate machine learning with a physically inspired representation*, *The journal of physical chemistry letters* **10** (2019), no. 17 4962–4967.

[79] O. T. Unke and M. Meuwly, *Physnet: a neural network for predicting energies, forces, dipole moments, and partial charges*, *Journal of chemical theory and computation* **15** (2019), no. 6 3678–3693.

[80] O. T. Unke, S. Chmiela, M. Gastegger, K. T. Schütt, H. E. Sauceda, and K.-R. Müller, *Spookynet: Learning force fields with electronic degrees of freedom and nonlocal effects*, *Nature communications* **12** (2021), no. 1 1–14.

[81] J.-f. Xia, Y.-l. Zhang, and B. Jiang, *Efficient selection of linearly independent atomic features for accurate machine learning potentials*, *Chinese Journal of Chemical Physics* **34** (2022), no. 6 695.

[82] B. A. P. Kermode, James R, M. C. Payne, R. Kondor, and G. Csányi, "QUIP." `http://www.libatoms.org`.

[83] J. Snoek, H. Larochelle, and R. P. Adams, *Practical bayesian optimization of machine learning algorithms*, *Advances in neural information processing systems* **25** (2012).

[84] 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** (2015), no. 1 148–175.

[85] R. Cerbino and V. Trappe, *Differential dynamic microscopy: probing wave vector dependent dynamics with a microscope*, *Physical review letters* **100** (2008), no. 18 188102.

[86] M. Gu, Y. He, X. Liu, and Y. Luo, *Ab initio uncertainty quantification in scattering analysis of microscopy*, *arXiv preprint arXiv:2309.02468* (2023).

[87] M. Gu, X. Fang, and Y. Luo, *Data-driven model construction for anisotropic dynamics of active matter*, *PRX Life* **1** (2023), no. 1 013009.

[88] S. R. Marder, D. N. Beratan, and L.-T. Cheng, *Approaches for optimizing the first electronic hyperpolarizability of conjugated organic molecules*, *Science* **252** (1991), no. 5002 103–106.

[89] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical recipes: The art of scientific computing, cambridge university press, cambridge, new york, new rochelle, melbourne, sydney, 1986, 20+ 818 pp.*, 1990.

[90] P. I. Frazier, *A tutorial on Bayesian optimization*, *arXiv preprint arXiv:1807.02811* (2018).

[91] T. J. Ypma, *Historical development of the newton–raphson method*, *SIAM review* **37** (1995), no. 4 531–551.

[92] I. Štich, R. Car, M. Parrinello, and S. Baroni, *Conjugate gradient minimization of the energy functional: A new method for electronic structure calculation*, *Physical Review B* **39** (1989), no. 8 4997.

[93] M. Hoffmann and F. Noé, *Generating valid euclidean distance matrices*, *arXiv preprint arXiv:1910.03131* (2019).

[94] K. Korovina, S. Xu, K. Kandasamy, W. Neiswanger, B. Poczos, J. Schneider, and E. Xing, *Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations*, in *International Conference on Artificial Intelligence and Statistics*, pp. 3393–3403, PMLR, 2020.

[95] R.-R. Griffiths and J. M. Hernández-Lobato, *Constrained Bayesian optimization for automatic chemical design using variational autoencoders*, *Chemical science* **11** (2020), no. 2 577–586.

[96] A. Stuke, P. Rinke, and M. Todorović, *Efficient hyperparameter tuning for kernel ridge regression with Bayesian optimization*, *Machine Learning: Science and Technology* **2** (2021), no. 3 035022.

[97] H. J. Kushner, *A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise*, .

[98] J. Močkus, *On Bayesian methods for seeking the extremum*, in *Optimization Techniques IFIP Technical Conference: Novosibirsk, July 1–7, 1974*, pp. 400–404, Springer, 1975.

[99] D. R. Jones, M. Schonlau, and W. J. Welch, *Efficient global optimization of expensive black-box functions*, *Journal of Global optimization* **13** (1998) 455–492.

[100] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, *Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization*, *ACM Transactions on mathematical software (TOMS)* **23** (1997), no. 4 550–560.

[101] T. L. Lai, H. Robbins, *et. al.*, *Asymptotically efficient adaptive allocation rules*, *Advances in applied mathematics* **6** (1985), no. 1 4–22.

[102] T. G. authors, "GPyOpt: A Bayesian optimization framework in python." `http://github.com/SheffieldML/GPyOpt`, 2016.

[103] F. Nogueira, *Bayesian Optimization: Open source constrained global optimization tool for Python*, 2014–.

[104] D. Ackley, *A connectionist machine for genetic hillclimbing kluwer acad*, 1987.

[105] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et. al.*, *Pytorch: An imperative style, high-performance deep learning library*, *Advances in neural information processing systems* **32** (2019).

[106] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *arXiv preprint arXiv:1412.6980* (2014).

[107] X. Fang, M. Gu, and J. Wu, *Reliable emulation of complex functionals by active learning with error control*, *The Journal of Chemical Physics* **157** (2022), no. 21.

[108] K. R. Anderson, I. A. Johanson, M. R. Patrick, M. Gu, P. Segall, M. P. Poland, E. K. Montgomery-Brown, and A. Miklius, *Magma reservoir failure and the onset of caldera collapse at kīlauea volcano in 2018*, *Science* **366** (2019), no. 6470.

[109] H. Zhao, F. Amann, and J. Kowalski, *Emulator-based global sensitivity analysis for flow-like landslide run-out models*, *Landslides* **18** (2021), no. 10 3299–3314.

[110] P. Ma, G. Karagiannis, B. A. Konomi, T. G. Asher, G. R. Toro, and A. T. Cox, *Multifidelity computer model emulation with high-dimensional output: An application to storm surge*, *Journal of the Royal Statistical Society Series C: Applied Statistics* **71** (2022), no. 4 861–883.

[111] M. Gu, Y. Lin, V. C. Lee, and D. Y. Qiu, *Probabilistic forecast of nonlinear dynamical systems with uncertainty quantification*, *Physica D: Nonlinear Phenomena* **457** (2024) 133938.

[112] J. Oakley, *Bayesian uncertainty analysis for complex computer codes*. PhD thesis, University of Sheffield, 1999.

[113] "Python package: Atomized Force Fields."
`https://github.com/UncertaintyQuantification/AFF`. Accessed: 2022-04-28.

[114] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[115] 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** (2014), no. 4 A1500–A1524.

[116] A. V. Vecchia, *Estimation and model identification for continuous spatial processes*, *Journal of the Royal Statistical Society: Series B (Methodological)* **50** (1988), no. 2 297–312.

[117] M. Katzfuss and J. Guinness, *A general framework for Vecchia approximations of Gaussian processes*, .

[118] M. Katzfuss, J. Guinness, and E. Lawrence, *Scaled Vecchia approximation for fast computer-model emulation*, *SIAM/ASA Journal on Uncertainty Quantification* **10** (2022), no. 2 537–554.