Error-informed Gaussian process regression for predicting DFT quantities

Anna Paulish[1]

[1]École Polytechnique Fédérale de Lausanne.

June 27, 2025

Nowadays, data-driven methods have gained popularity in computational materials science. However, most atomistic machine learning approaches assume that errors in the training data follow small Gaussian noise distributions, which requires all simulations to be of uniformly high quality. This means that numerical parameters – such as discretisation basis, k-point sampling, and tolerance settings – have to be accurate throughout the entire data generation process, leading to high computational costs. This work aims to develop a heteroscedastic Gaussian process regression (GPR) framework for predicting Density Functional Theory (DFT) quantities by incorporating discretization error estimates, moving from the standard Gaussian noise assumption to a more quantitative description of the error. The proposed error-informed GPR approach will be able to better model datasets of different quality, giving a way to reduce data generation costs while maintaining high predictive accuracy. Furthermore, we will identify the key aspects necessary to implement this GPR model as a practical tool enabling accelerated and cost-efficient computational materials discovery.

1 Introduction

Materials discovery and design are crucial for advancements in various domains, including electronics, energy storage, medicine, and environmental sustainability [1, 2]. Finding materials with target properties through experimental trial-and-error methods is both time-consuming and resource-intensive. Therefore, computations based on *ab initio* material modeling, where the atomic structure is the only input and no emperical parameters are used, are widely used to study material structures and predict a range of atomic-scale properties. One of the most widely used computational methods for discovering new materials with specific properties is a DFT-based high-throughput materials screening [3, 4]. Despite its popularity, DFT-based simulations often remain challenging due to their high computational cost. In its general formulation, the mathematical problem underlying DFT can be seen as the following minimization problem:

$$\min_{P \in \mathcal{P}} E(P), \tag{1}$$

where $E(P)$ is a non-linear DFT energy functional, and $\mathcal{P}$ is a manifold of density matrices. The ground-state energy of an atomic system corresponds to $E(P_*)$, where P_* is the resulting minimizer,

$$P_* = \arg\min_{P \in \mathcal{P}} E(P), \tag{2}$$

from which other ground-state quantities of interest $Q(P_*)$ can be derived. While a single DFT calculation can take several CPU hours, incorporating it into the material search workflow can immediately become prohibitively expensive.

In recent years, data-driven approaches have become a popular alternative accelerating computational materials discovery [5], particularly in the development of machine-learned interatomic potentials (MLIPs) [6, 7, 8, 9]. MLIPs play a crucial role in accurately modeling atomic-scale interactions within materials while significantly reducing the computational cost compared to direct quantum mechanical methods.

The main goal of MLIPs models is to approximate a mapping $\mathbf{x} \mapsto Q(\mathbf{x})$ using the dataset of n observations $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$. Each input vector $\mathbf{x}_i$ encodes an atomic structure in a representation that is invariant to translations, rotations, and permutations of identical atoms [10]. Respectively, each observation y_i provides a numerical estimate of a target property $Q(\mathbf{x}_i)$, usually the atomic energies and forces, coming from a DFT simulation of an atomic system.

Most common data-driven approaches assume that errors in the training data are small, independent, and identically distributed (i.i.d.) Gaussian noise:

$$y_i = Q(\mathbf{x}_i) + \varepsilon_i, \text{ where } \varepsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2). \tag{3}$$

This assumption implies that for a large training set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$, containing thousands or even millions of samples, all simulations must be of **homogeneously** high quality [7]. Achieving this level of precision requires accurate numerical parameters, such as discretisation basis, k-point sampling, and tolerance settings, throughout the entire data generation process, making it a computationally challenging task.

In our case, the errors in DFT simulations arise from several sources [11], including:

- *Model error* inherent in the DFT approximations itself, such as the exchange-correlation functional, and pseudopotentials.
- *Discretization error* coming from finite basis set.
- *Algorithmic error* arising from the iterative algorithm used to solve the underlying equations for the energy minimization.
- *Arithmetic error* due to finite-precision arithmetic.

Recent progress in understanding and estimating the numerical error in plane-wave based DFT simulations has led to the development of a method for estimating the discretization error ε_i associated with a DFT quantity $Q(\mathbf{x}_i)$ on gapped systems (insulators or semiconductors) at zero temperature [12]. This provides a more quantitative description of the error and allows us to incorporate a heteroscedastic noise model:

$$y_i = Q(\mathbf{x}_i) + \varepsilon_i, \text{ where } \varepsilon_i \sim \mathcal{N}(0, \sigma(\mathbf{x}_i)^2). \tag{4}$$

To summarize, a sequence of random variables ε_i is called **homoscedastic** if all ε_i have the same finite variance σ; otherwise, they are **heteroscedastic**.

Gaussian process regression (GPR) is a powerful non-parametric regression technique [13] that has been successfully applied to atomistic modeling tasks, including in the construction of MLIPs [7, 14, 15]. However, most of existing GPR-based models rely on a homoscedastic noise model for the DFT error.

On the other hand, by incorporating quantitative estimates of the discretization error, we are able to develop heteroscedastic GPR [16], which can better handle datasets of varying quality, provide reliable uncertainty estimates together with predictions, and reduce the computational cost of data generation while maintaining high predictive accuracy.

2 Research objective

The primary goal of this work is to develop robust and practical frameworks for integrating error control techniques into surrogate models, specifically focusing on GPR, to enable accelerated and cost-efficient data-driven materials modeling. By incorporating advanced error estimation methods, this work aims to reduce the cost of data generation without compromising predictive accuracy. The specific objectives are:

1. Development of an error-informed GPR model.
   - 1.1 Implement a heteroscedastic GPR model and test its performance compared to homoscedastic GPR on a simple numerical integration problem.
   - 1.2 Test heteroscedastic GPR in atomistic modeling context on the example of energy-volume curve prediction for symmetric silicon structure.
   - 1.3 Develop heteroscedastic GPR for a widely used GP-based MLIP framework such as Gaussian Approximated Potential (GAP) for predicting ground-state properties of the materials. Test its performance on silicon dataset and evaluate the potential data generation cost savings while keeping on a prediction accuracy.
2. Include discretization error estimates as part of heteroscedastic GAP model.
   - 2.1 Investigate several noise models with set of hyperparameters, that need to be tuned by comparing with the actual distribution of the errors in ground-state properties.
   - 2.2 Explore strategies for including discretization error as an additional regression target, to improve prediction accuracy for both quantities of interest and their associated errors.
   - 2.3 Incorporate uncertainty quantification from the error-informed GPR model into a Bayesian active learning method to selectively sample new data points to efficiently guide the training of GAP models.
3. Development of a multitask GPR framework.
   - 3.1 Investigate multitask GPR model trained with DFT data obtained using multiple E_{cut} values for simultaneously predicting ground-state properties and their associated uncertainties, improving both efficiency and predictive power.
4. Application to broader datasets.
   - 4.1 Validate the proposed error-informed GPR using well-established datasets in materials modeling with varying levels of numerical accuracy, such as general purpose silicon dataset [7], and verification dataset containing a wider range of material systems [17].
   - 4.2 Compare the performance of the proposed models against traditional GPR approaches to quantify improvements in accuracy, robustness, and computational efficiency.

3 Kohn-Sham density functional theory

DFT is a fundamental quantum mechanical framework widely used to study the electronic structure of materials. The key idea of DFT is that the ground-state properties of a many-electron system can be uniquely determined from its electron density $\rho(\mathbf{r})$, rather than solving the many-body Schrödinger equation explicitly.

In the Kohn-Sham (KS) formulation of DFT, the complex many-body problem is mapped onto a system of non-interacting particles moving in an effective potential [18]. The total energy functional of the system is expressed as:

$$E[\rho] = T_s[\rho] + \int v_{\text{ext}}(\mathbf{r})\rho(\mathbf{r})\, d\mathbf{r} + E_{\text{H}}[\rho] + E_{\text{xc}}[\rho], \tag{5}$$

where $T_s[\rho]$ is the kinetic energy of non-interacting electrons, $v_{\text{ext}}(\mathbf{r})$ is the external potential, $E_{\text{H}}[\rho]$ is the Coulomb interaction energy, and $E_{\text{xc}}[\rho]$ is the exchange-correlation energy, which accounts for quantum many-body effects.

The effective potential $v_{\text{KS}}(\mathbf{r})$ is given by:

$$v_{\text{KS}}(\mathbf{r}) = v_{\text{ext}}(\mathbf{r}) + v_{\text{H}}(\mathbf{r}) + v_{\text{xc}}(\mathbf{r}), \tag{6}$$

and the Kohn-Sham equations, which describe the behavior of single-particle orbitals, are solved self-consistently:

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + v_{\text{KS}}(\mathbf{r})\right]\psi_i(\mathbf{r}) = \epsilon_i\psi_i(\mathbf{r}), \tag{7}$$

where $\psi_i(\mathbf{r})$ are the Kohn-Sham orbitals, and ϵ_i are their corresponding eigenvalues. To obtain the ground-state electron density, the self-consistent field (SCF) loop is employed. This iterative procedure starts with an initial guess for the electron density, solves the Kohn-Sham equations to obtain new orbitals, updates the electron density, and repeats the process until convergence is reached.

3.1 Plane-wave discretization

There exist different methods to numerically solve the Kohn-Sham equations for materials. The choice of the basis depends on the system on studies. For periodic crystals, the most popular method is the plane-wave expansion, in which the wavefunctions $\psi_i(\mathbf{r})$ are expanded in a plane-wave basis:

$$\psi_i(\mathbf{r}) = \sum_{\mathbf{G}} c_{i,\mathbf{G}} e^{i\mathbf{G}\cdot\mathbf{r}}, \tag{8}$$

where $\mathbf{G}$ are reciprocal lattice vectors, and $c_{i,\mathbf{G}}$ are the expansion coefficients.
The plane-wave basis offers several advantages:

- **Systematic Convergence:** The accuracy can be controlled by adjusting the cutoff energy E_{cut}, which determines the maximum $|\mathbf{G}|^2$ included in the expansion.
- **Periodic Systems:** Plane waves naturally incorporate periodic boundary conditions, making them particularly suitable for modeling crystalline solids and other periodic systems.

Despite these advantages, plane-wave methods can be computationally expensive, as the number of basis functions grows with the system size and the chosen energy cutoff E_{cut}. To address this, pseudopotentials are used, replacing the explicit treatment of core electrons with an effective potential while describing only the valence electrons [19, 20].

4 Probabilistic regression

In this section, we abstract from the specific models used to describe material properties and formulate the general problem of probabilistic regression. The goal is to approximate an unknown function:

$$f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R} \tag{9}$$

given a set of data points $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$, consisting of n observed samples of the function at specific input locations. Probabilistic regression extends standard regression by not only estimating the mean value of $f(\mathbf{x}_*)$ at a new test input $\mathbf{x}_*$ but also quantifying the uncertainty in the prediction.

In this work, we employ Gaussian process regression, a non-parametric Bayesian framework, to infer the relationship between inputs – representations of atomic configurations – and outputs, which correspond to ground-state properties such as energy, atomic forces, and stresses, obtained from DFT simulations.

A Gaussian process is a stochastic process $f(\mathbf{x})$ such that any finite collection of its random variables follows a multivariate Gaussian distribution. It is characterized by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$, defined as:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \end{aligned} \tag{10}$$

Then, the GP prior on the function f, can be written as:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{11}$$

The covariance function $k(\mathbf{x}, \mathbf{x}')$ encodes the correlation between the function values at two input locations $\mathbf{x}$ and $\mathbf{x}'$.

4.1 Gaussian process regression

In GPR, the observed data $\mathcal{D}$ is used to update the Gaussian process prior into a posterior distribution over functions. Observations y_i are assumed to be corrupted by Gaussian noise ε_i with variance $\sigma^2(\mathbf{x}_i)$, i.e.,

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2(\mathbf{x}_i)). \tag{12}$$

Further,X denotes a $d \times n$ design matrix, composed of all n input column vectors, while $\mathbf{y}$ represents the corresponding target vector.

The joint distribution of the observed target values $\mathbf{y}$ and the function values $\mathbf{f}_* = f(X_*)$ at test inputs X_* under the Gaussian process prior is:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) + \Sigma & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right), \tag{13}$$

where $K(X, X)$ and $K(X_*, X_*)$ are covariance matrices, and $K(X, X_*)$ represents the cross-covariance between training and test points. Here, Σ models the noise covariance matrix:

$$\Sigma = \begin{cases} \sigma^2 I & \text{(homoscedastic noise)}, \\ \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2) & \text{(heteroscedastic noise)}. \end{cases}$$

The posterior predictive distribution for the function values at test inputs X_* is derived by conditioning on the observed data $\mathcal{D}$. The key predictive equations are:

$$\mathbf{f}_* \mid X, \mathbf{y}, X_* \sim \mathcal{N}(\overline{\mathbf{f}_*}, \text{cov}(\mathbf{f}_*)), \tag{14}$$

$$\overline{\mathbf{f}_*} = \mathbf{k}_*^\top (K + \Sigma)^{-1} \mathbf{y}, \tag{15}$$

$$\text{cov}(\mathbf{f}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (K + \Sigma)^{-1} \mathbf{k}_*, \tag{16}$$

where $\mathbf{k}_* = K(X, X_*)$ represents the covariance between training and test points, and $k(\mathbf{x}_*, \mathbf{x}_*)$ is the prior variance of the function at the test point.

4.2 Subset of regressors

The Subset of Regressors (SR) is an approximation to the standard GPR model, designed to reduce the computational complexity of GPR for large datasets. It defines the kernel function $k(\mathbf{x}, \mathbf{x}')$ using a subset of m representative points, $\{\mathbf{z}_j\}_{j=1}^m$, where $m \ll n$. The approximated covariance function is given by:

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = \mathbf{k}_m(\mathbf{x})^\top K_{mm}^{-1} \mathbf{k}_m(\mathbf{x}'), \tag{17}$$

where:

- $K_{mm} \in \mathbb{R}^{m \times m}$ is the kernel matrix of the representative points, where $[K_{mm}]_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$,
- $\mathbf{k}_m(\mathbf{x}) \in \mathbb{R}^m$ is the kernel vector between $\mathbf{x}$ and the representative points, defined as $[\mathbf{k}_m(\mathbf{x})]_i = k(\mathbf{x}, \mathbf{z}_i)$.

The kernel matrix $K \in \mathbb{R}^{n \times n}$ for all training points can be written as:

$$K \approx K_{mn}^\top K_{mm}^{-1} K_{mn},$$

where $K_{mn} \in \mathbb{R}^{n \times m}$ is the kernel matrix between the n training points and the m representative points, and $[K_{mn}]_{ij} = k(\mathbf{x}_i, \mathbf{z}_j)$. This approximation corresponds to the Nyström approximation of the kernel matrix K [21].

The SR model for homoscedastic noise is described in [13]. In this work, we derived the SR model formulation under the assumption of heteroscedastic noise that makes heteroscedastic sparse GPR implementation possible. For detailed derivations, see Section A of the Appendix. The resulting predictive mean and variance of the SR model at a test point $\mathbf{x}_*$ are given by:

$$\mathbb{E}[f_{\text{SR}}(\mathbf{x}_*)] = \mathbf{k}_m(\mathbf{x}_*)^\top \left(K_{mn}\Sigma^{-1}K_{nm} + K_{mm}\right)^{-1} K_{mn}\Sigma^{-1}\mathbf{y}, \tag{18}$$

$$\text{Var}[f_{\text{SR}}(\mathbf{x}_*)] = \mathbf{k}_m(\mathbf{x}_*)^\top (K_{mn}\Sigma^{-1}K_{nm} + K_{mm})^{-1}\mathbf{k}_m(\mathbf{x}_*) \tag{19}$$

where:

- $\mathbf{k}_m(\mathbf{x}_*) \in \mathbb{R}^m$ is the kernel vector between the test point $\mathbf{x}_*$ and the representative points,
- $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$ is the diagonal noise covariance matrix.

Gaussian process regression using this reduced-rank representation of the kernel matrix makes it possible to have a computationally efficient approximation for Gaussian process regression [13] and reduces the computational cost from $O(n^3)$ (for a full GP) to $O(nm^2)$, where $m \ll n$.

5 Proof of principle and application to realistic system

5.1 Numerical integration

To test the concept on a simple example, we consider the numerical integration of a one-dimensional function:

$$f(x) = \int_{-3}^{\sin(2\pi x)} \exp\left(-\left(\sin\left(xz\right)\right)^2 - z^2\right) dz = \int_{-3}^{\sin(2\pi x)} g(x, z)\, dz. \tag{20}$$

The goal is to approximate $f(x)$ with GP regression using the knowledge about the actual heteroscedastic error inherent to numerical quadrature. Here, we use the AbstractGPs.jl package [22] in Julia, which provides a framework for working with Gaussian processes.

In this case we have a control over the integration error, computed as a difference between the accurate and crude numerical quadrature, that will be incorporated in the GP model. The function $g(x, z)$ to be integrated is illustrated in Figure 1.
The plot on the left of Figure 2 shows the exact underlying function (black dashed line), the input data for fitting the model (blue dots) sampled from a numerical approximation (blue line). On the right plot you can see the posterior distribution for the homoscedastic GPR model (yellow line), and the heteroscedastic model, which was trained on the same input data but with incorporated information about the actual numerical errors (green line). Analyzing these plots we can conclude that heteroscedastic GPR model indeed provides better predictive accuracy compared to the homoscedastic model.

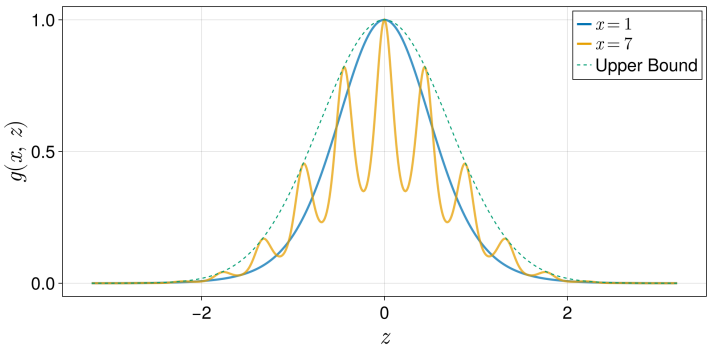

Figure 1: The function $g(x, z)$ used for numerical integration.

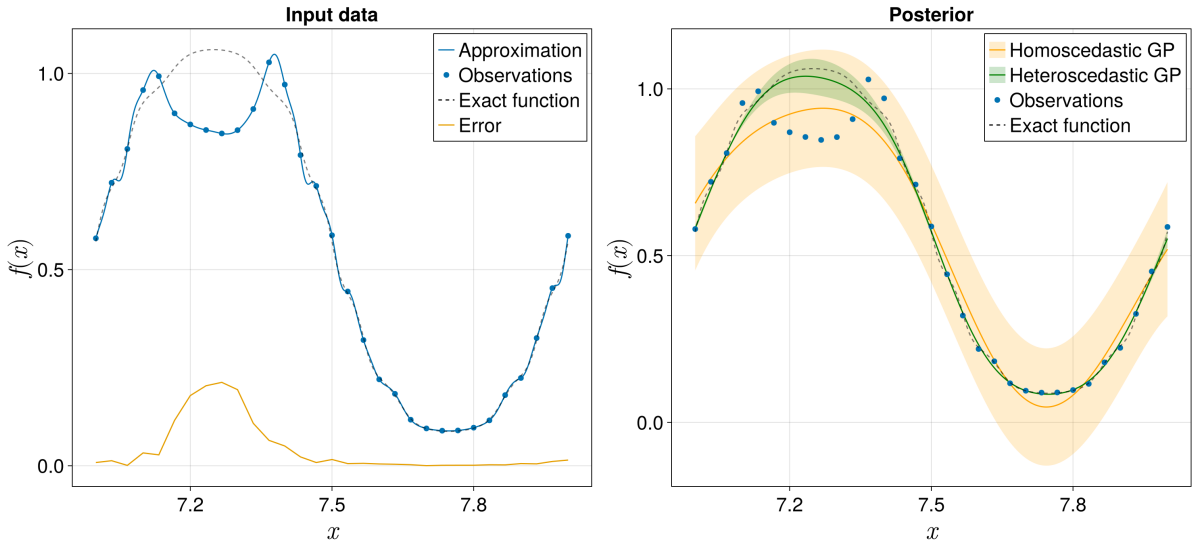

Figure 2: Left: The dashed black line represents the result of a more accurate numerical integration, while the blue line depicts a crude approximation from which the input data points are sampled. The orange line indicates the numerical error, calculated as the difference between the accurate and crude numerical quadrature. Right: Posterior distributions of the fitted GP regression model with both heteroscedastic and homoscedastic assumptions. The lines represent predicted mean functions, while the shaded regions correspond to the predicted variance of the corresponding GPR models.

5.2 Prediction of material property: equation of state

As a simple example in the atomistic modeling context, we consider a bulk crystalline silicon structure and calculate an energy-volume curve (or equation of state), which is a quantity that describes the behavior of a solid under compression or expansion.
The goal is to approximate this quantity using a GP regression model on a mixed-fidelity dataset. To construct the dataset, we perform DFT calculations for a set of lattice constants a_i with varying plane-wave cutoff (E_{cut}) values using the Density-Functional ToolKit (DFTK), an open-source Julia-based DFT package [23]. Lower E_{cut} values result in computationally cheaper but less accurate calculations. Here, the dataset has been constructed to model the case where it is insufficient for some structures (here compressed lattices). By comparing these calculations with reference values computed at a sufficiently high E_{cut}, we can estimate the actual errors in the low-fidelity data and incorporate this information into a heteroscedastic noise model:

$$y_i = E(a_i) + \varepsilon_i, \text{ where } \varepsilon_i \sim \mathcal{N}(0, \sigma^2(\mathbf{x}_i)), \tag{21}$$

where $E(a_i)$ represents the actual energy of the system for the lattice constant a_i, while y_i is the the result of DFT calculations. Errors ε_i can also be estimated using the method derived in [12].

We begin by training a homoscedastic GPR model using only high-fidelity data points and obtain the posterior distribution, shown as the red line in Figure 3. Next, we train a heteroscedastic GPR model using the same high-fidelity data points, augmented with low-fidelity data and information about the underlying errors. The posterior distribution of the heteroscedastic GPR model is shown as the green line, while the reference energy-volume curve is depicted as a dashed black line in Figure 3.

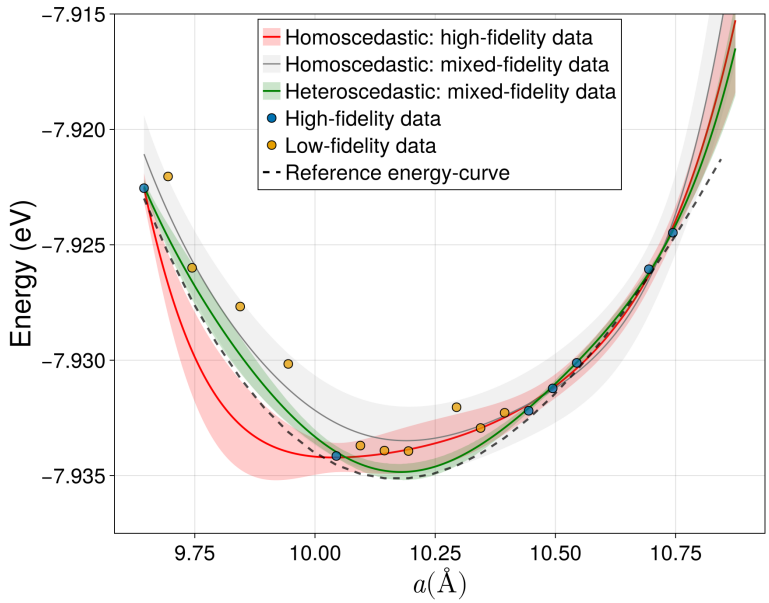

Figure 3: Posterior distributions of the GPR models approximating an energy volume curve for silicon structure fitted on high-fidelity data (red) and on mixed-fidelity data with incorporated information about the errors (green). The gray line represents the posterior distribution of the homoscedastic model trained on mixed-fidelity dataset. The shaded regions correspond to the predicted variance of the corresponding models.

From the results in Figure 3, it is evident that incorporating low-fidelity data points along with error estimates improves the predictive accuracy of the model. This demonstrates the value of leveraging mixed-fidelity datasets and heteroscedastic GPR modeling for computational efficiency without compromising accuracy.

In this section, we validated our approach on an example of numerical integration and an energy-volume relationship for symmetric silicon structure. In the next section, we extend this methodology to a more general case, demonstrating its applicability to predicting the ground-state properties of a wider range of silicon structures using the dataset from [7].

6 Gaussian Approximation Potentials framework

As previously discussed, while DFT provides accurate predictions of material properties, it comes at the cost of significant computational resources, making large-scale simulations infeasible. MLIPs offer an efficient alternative, allowing accurate approximations of DFT-level accuracy. One of the most established and widely used GP-based MLIPs is the GAP framework. Our goal was to develop and validate an error-informed GPR framework within the GAP methodology, incorporating knowledge about the errors in the training data to overcome the challenge of requiring homogeneously high-quality datasets.

The idea behind the GAP framework is to approximate the total potential energy of a

system as a sum of contributions from local atomic environments:

$$E_{\text{total}} = \sum_i E_i, \tag{22}$$

where E_i is the local atomic energy, which depends on the configuration of atoms in the local environment of the atom i. Each local energy E_i is modeled using GP regression, which predicts the energy based on a set of training data through a weighted sum of kernel evaluations that measure the similarity between atomic environments in the training set. The predicted energy is given by:

$$E_i = \sum_j \alpha_j k(\mathbf{d}_i, \mathbf{d}_j), \tag{23}$$

where α_j are weights determined during training, $\mathbf{d}_i$ represents the descriptor for the local environment of atom i, $k(\mathbf{d}_i, \mathbf{d}_j)$ is the kernel function that quantifies the similarity between atomic environments $\mathbf{d}_i$ and $\mathbf{d}_j$.

In this work, we employ the Smooth Overlap of Atomic Positions (SOAP) descriptor [24, 25] which encodes the spatial distribution of neighboring atoms in a rotationally, translationally, and permutationally invariant way. The SOAP kernel can be written as a scalar product:

$$k(\mathbf{d}_i, \mathbf{d}_j) = |\mathbf{d}_i \cdot \mathbf{d}_j|^{\zeta}, \tag{24}$$

where ζ is a tunable parameter controlling the sensitivity of the kernel. Once trained, the GAP model can efficiently predict energies and forces for new configurations, achieving near-DFT accuracy while being significantly faster.

6.1 Dataset

In this study, we use DFT calculations to generate a mixed-fidelity dataset for training surrogate models. Simulations are carried out using the DFTK package. The dataset consists of a subset of the GAP silicon dataset from [7], with varying discretization errors controlled by the plane-wave energy cutoff (E_{cut}) parameter. The configuration types of silicon structures used in this study can be found in Figure 4.

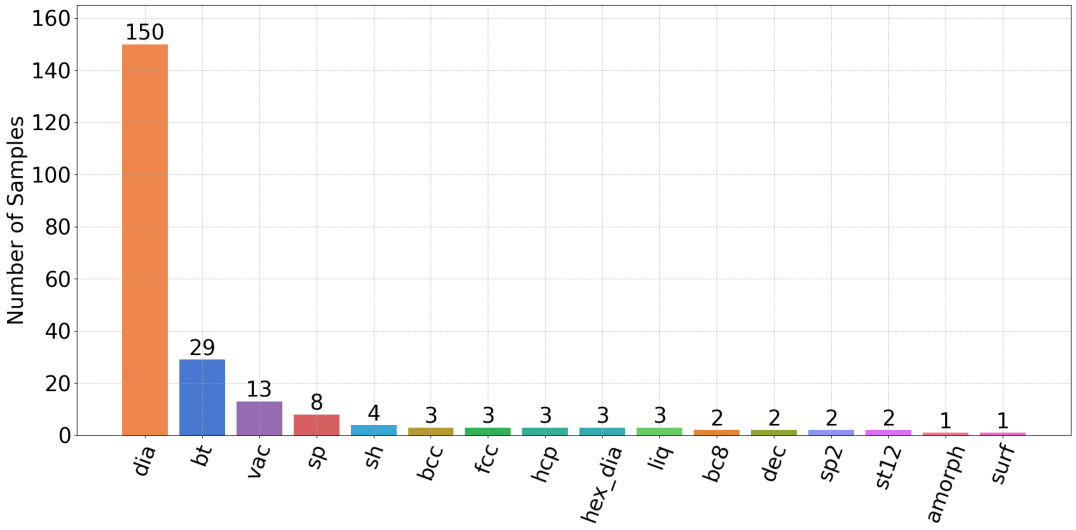

Figure 4: Configuration types in subset of GAP silicon dataset from [7].

DFT calculations are performed with the following settings. The plane-wave cutoff energies (E_{cut}) are set to 10 Ha, 18 Ha, and 26 Ha. The maximum k-point spacing is 0.1

inverse Bohrs (approximately $2\pi \cdot 0.03$ Å^{-1}). The SCF convergence criterion is defined as $\Delta\rho < 10^{-6}$. The temperature is set to 10^{-3} atomic units. The exchange-correlation functional used is the Perdew–Burke–Ernzerhof (PBE) model [26]. The pseudopotential is taken from PseudoDojo for silicon [27].

6.2 Results

To explore the applicability of our approach to machine-learned interatomic potentials, we implemented heteroscedastic GPR within the GAP framework using Metatrain, which is a Python package to train and evaluate atomistic models of various architectures [28]. We consider two training datasets:

- **High-fidelity dataset**, consisting of 25 diamond structures with an energy cutoff of $E_{\text{cut}} = 26$ Ha, corresponding to 50 local atomic environments.
- **Mixed-fidelity dataset**, consisting of the same 25 structures, 25 additional β-Sn ($E_{\text{cut}} = 18$ Ha), and 10 more structures (5 diamond, 5 vacancy) with a lower energy cutoff of $E_{\text{cut}} = 10$ Ha – resulting in a total 60 structures and 495 local atomic environments.

The test dataset consists of 123 structures (115 diamond and 8 vacancy), corresponding to 2162 local atomic environments.

We compare the standard **homoscedastic GAP** model trained on both training datasets (see Figure 5, panel (a) and (b)) with the developed **heteroscedastic GAP** model trained on mixed-fidelity dataset, incorporating the actual underlying energy error, which was calculated as the deviation from the reference data with $E_{\text{cut}} = 26$ Ha (see Figure 5, panel (c)). For the homoscedastic model, the noise variance for energies was set to $\sigma = 0.001$ eV/atom, a typical value for crystal configuration [7], and we applied the same noise variance for forces.

Analyzing Figure 5 we observe that the developed error-informed GAP model trained even on only low-fidelity data has improved accuracy compared to the standard GAP model trained on a smaller amount but high-fidelity data, confirming the benefits of integrating error-aware training in machine-learned interatomic potentials.

7 Implementation details

In this work, we implemented a custom GPR framework for learning energy and force targets from DFT calculations, using SOAP descriptors. Our implementation, developed in Python using PyTorch and Metatensor, extends previous baseline models in two key ways:

- **Uncertainty quantification:** The model computes predictive variances for both energies and (optionally) forces, enabling robust error estimation.
- **Heteroscedastic noise modeling:** Structure-dependent and atom-wise noise levels can be provided as inputs to the model, which scales its kernel matrix and targets accordingly, allowing it to account for varying fidelity in training data.

These capabilities were *not present* in earlier implementations, which assumed homoscedastic noise and lacked any form of confidence estimation.

The full model consists of several modular components:

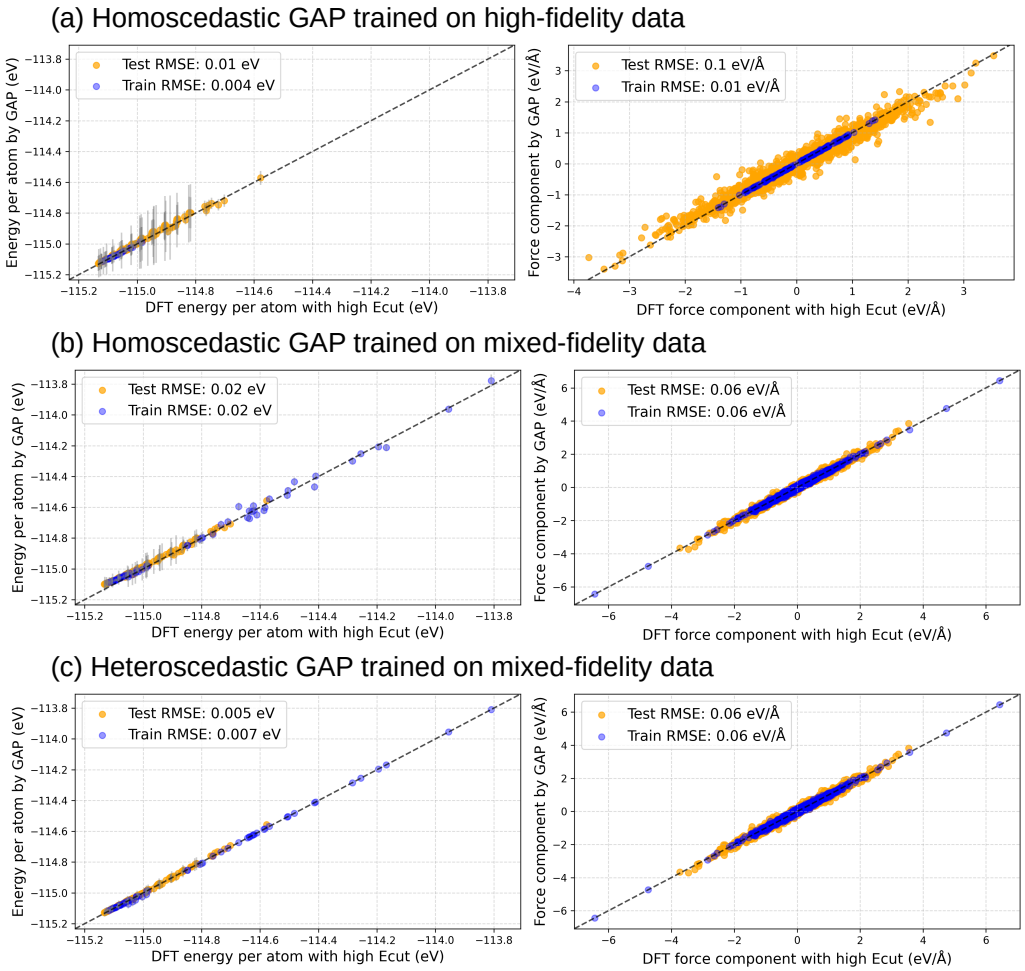

Figure 5: (a): Energy per atom and force component predictions using the homoscedastic GAP model trained on a limited high-fidelity dataset, compared against reference DFT calculations at $E_{\text{cut}} = 26$ Ha. (b): Energy per atom and force component predictions using the homoscedastic GAP model trained on a mixed-fidelity dataset, compared against reference DFT calculations at $E_{\text{cut}} = 26$ Ha. (c): Energy per atom and force components predictions using the heteroscedastic GAP model against reference DFT calculations at $E_{\text{cut}} = 26$ Ha. By incorporating mixed-fidelity data and explicit error estimates, the heteroscedastic model demonstrates improved predictive accuracy, reducing deviations from the test reference values observed in the homoscedastic model.

7.1 Descriptor calculation

SOAP descriptors are computed using the `SoapPowerSpectrum` class in the `featomic` package. These descriptors encode local atomic environments and are differentiable with respect to atomic coordinates. All hyperparameters (e.g., cutoff radius, basis function types) are specified via a YAML configuration.

Listing 1: `SoapCalculator`: Computes SOAP descriptors from atomic structures using specified hypers (lines 19–24 in `gpr.py`).

```
class SoapCalculator(torch.nn.Module):
    def __init__(self, soap_hypers: dict) -> None:
        super().__init__()
        self.soap_calculator = SoapPowerSpectrum(**soap_hypers)

    def forward(self, systems, gradients: List[str] = None) -> TensorMap:
        soap_vector = self.soap_calculator.compute(systems, gradients)
        soap_vector = soap_vector.keys_to_samples("center_type")
        soap_vector = soap_vector.keys_to_properties(["neighbor_1_type",
            "neighbor_2_type"])
        return soap_vector
```

7.2 Kernel construction

We use a polynomial kernel of degree $d = 2$ to compute pairwise similarities between SOAP descriptors, defined as:

$$K(\mathbf{x}, \mathbf{y}) = \left(\mathbf{x}^\top \mathbf{y}\right)^d \tag{25}$$

To reduce computational cost, we select a representative subset of representative atomic environments using Farthest Point Sampling (FPS):

Listing 2: Selects sparse SOAP features using Farthest Point Sampling (cell 6 in `gpr_example.ipynb`).

```
fps = _FPS(n_to_select=num_sparse_points)
soap_vector_train_fps = mts.remove_gradients(fps.fit_transform(soap_vector_train))
```

The `KernelCalculator` class computes the K_{nm} and K_{mm} matrices. The kernel matrix K_{nm} is constructed between all training points and the selected sparse points, while K_{mm} is precomputed from the sparse set. These matrices are used for the Subset of Regressors approximation to reduce the computational complexity of GPR from cubic to linear in the number of training points.

Listing 3: `PolynomialKernel`: Instantiates a degree-2 kernel and computes pairwise similarities between training and sparse SOAP vectors.

```
kernel = PolynomialKernel(degree=2)
kernel_matrix = kernel(soap_vector_train, soap_vector_train_fps)
```

7.3 Noise modeling

Unlike standard *homoscedasticity* GPR models that assume constant observation noise, our implementation supports *heteroscedastic noise*, where the uncertainty levels can vary across training samples. This is essential when working with mixed-fidelity or varying reliability data, as it allows the model to prioritize more reliable inputs. Our implementation applies left-side normalization of both the kernel rows and the target values:

$$K_{nm} \leftarrow \Sigma^{-1/2} K_{nm}, \quad y \leftarrow \Sigma^{-1} y$$

This corresponds to per-sample weighting during the regression step and is equivalent to minimizing a noise-weighted squared error loss. Here, $\Sigma = \mathrm{diag}(\alpha_1, \ldots, \alpha_n)$ encodes structure-dependent noise levels.

Listing 4: `GPR.fit`: Applies left-side noise scaling to kernel rows and target values (lines 192–193 in `gpr.py`).

```
k_nm_vals[:] /= alpha_energy[:, None]
energy_vals[:] /= alpha_energy[:, None]
```

Noise values can be specified either globally or per sample. If a scalar is provided, it is broadcast across all training structures; otherwise, a vector or tensor of values allows for fine-grained, sample-specific regularization:

Listing 5: `GPR.fit`: Scales kernel and targets using heteroscedastic noise estimates (lines 138–148 in `gpr.py`).

```
if isinstance(alpha_energy, float):
    alpha_energy = torch.ones(len(systems)) * alpha_energy

if isinstance(alpha_energy_grad, float):
    alpha_energy_grad = torch.ones(sum([len(s) for s in systems]), 3) * alpha_energy_grad
```

By incorporating noise information directly into the regression procedure, the model can dynamically adjust its sensitivity to each data point, improving robustness, especially for multi-fidelity or imperfect datasets.

7.4 Solving the regression system

We solve the GPR training problem using a numerically stable RKHS-QR method [29], incorporating a low-rank approximation via the SoR formulation. A small jitter is added to improve matrix conditioning:

Listing 6: `_SorKernelSolver.fit`: Solves the SoR problem using eigenvalue decomposition and QR factorization (lines 414–447 in `gpr.py`).

```
self._vk, self._Uk = scipy.linalg.eigh(KMM)
self._PKPhi = self._Uk[:, :self._nM] * 1 / np.sqrt(self._vk[:self._nM])
A = np.vstack([KNM @ self._PKPhi, np.eye(self._nM)])
Q, R = np.linalg.qr(A)
weights = self._PKPhi @ scipy.linalg.solve_triangular(R, Q.T @ np.vstack([Y, np.zeros((
    self._nM, Y.shape[1]))]))
```

This approach ensures stable computation of both weights and posterior covariances.

7.5 Configuration and hyperparameters

Model setup is controlled via a structured YAML input file that specifies all major components of the pipeline. This includes SOAP descriptor parameters, kernel definition, sparsification method, and noise regularization mode. An example configuration file is shown below:

Listing 7: YAML configuration defining kernel hyperparameters, noise settings, and training behavior.

```
krr:
  degree: 2
  num_sparse_points: 1000  # Set to null to use all training points

training:
  use_gradients: true

  alpha_energy:
    value: 0.001
    read_from: null  # Optional: path to file with per-structure energy uncertainties

  alpha_energy_grad:
    value: 0.001
    read_from: null  # Optional: path to file with per-atom force uncertainties

  predict_std_energy: true
  predict_std_energy_grad: false
```

The configuration fields are structured as follows:

- `krr`: Specifies kernel-related hyperparameters. In this case, a degree-2 polynomial kernel is used. The `num_sparse_points` field controls the number of representative environments used in the SoR approximation.
- `training.use_gradients`: If set to `true`, the model includes force data (energy gradients) during training.
- `alpha_energy` and `alpha_energy_grad`: Define the regularization strengths for energy and force targets, respectively. If the `read_from` path is set to a file, the model uses per-structure or per-atom noise values for heteroscedastic modeling; otherwise, the scalar `value` is used for homoscedastic noise.
- `predict_std_energy` and `predict_std_energy_grad`: Control whether the model outputs predictive uncertainty estimates for energy and forces, respectively.

This configuration format allows users to switch easily between homoscedastic and heteroscedastic training, and to activate uncertainty quantification features as needed for further tasks such as active learning or model validation.

7.6 Training procedure

The following steps summarize the full training pipeline described in the sections above:

1. Generate SOAP vectors and (optionally) force gradients.
2. Normalize targets by atom count and inverse noise levels.
3. Compute and scale kernel matrices using SoR.
4. Solve for GPR weights using the QR-based solver.

Both energy and force noise levels (α_{energy}, α_{grad}) can be customized for each data point.

7.7 Prediction and variance estimation

Given a test set, the model:

- Computes the test-train kernel matrix K_{tm} and (optionally) test-test kernel matrix K_{tt} for SOAP vectors.
- Applies the trained weights to predict mean energies and optionally gradients (forces).
- Adds back the mean composition-based baseline model to reverse the normalization applied during training.
- Computes predictive standard deviations for uncertainty-aware analysis.

Force predictions are computed via PyTorch `autograd`.

7.8 Uncertainty quantification

At prediction time, the model returns both a mean prediction $\hat{f}$ and a variance estimate $\mathrm{Var}[f(\mathbf{x})]$:

$$\mathrm{Var}[f(\mathbf{x})] = K_{tt} - K_{tm}^{\top} A^{-1} K_{tm} \tag{26}$$

The standard deviation $\sigma = \sqrt{\mathrm{Var}[f]}$ is stored alongside each prediction.

Listing 8: `predict_variance`: Computes predictive variance using the trained kernel inverse (line 311 in `gpr.py`).

```
var_energy = self.kernel_calculator.solver.predict_variance(KTM, KTT)
```

This uncertainty can be used to:

- Identify outliers or unreliable predictions.
- Drive active learning, e.g., query points with largest σ.
- Quantify the confidence of the model in later stages of computational pipelines, including molecular dynamics, geometry optimization, or screening procedures.

7.9 Evaluation pipeline

We evaluate model performance in terms of both predictive accuracy and uncertainty calibration. Accuracy metrics measure how close the predictions are to the true values, while calibration metrics assess whether the predicted uncertainties correctly reflect actual errors. Accuracy metrics:

- Root Mean Square Error (RMSE) for both energies and forces, computed across the test set.

- Parity plots with uncertainty intervals, visualizing the agreement between predicted and true values, with predicted standard deviations ($\pm\sigma$) as error bars.

Uncertainty calibration metrics can be computed as follows:

- Pearson correlation coefficient between predicted standard deviations and absolute prediction errors $|\hat{y} - y|$. A high correlation indicates that the model can correctly identify uncertain regions.
- Fraction of underconfident predictions, defined as the proportion of samples where the absolute error exceeds a multiple of the predicted uncertainty:

$$|\hat{y} - y| > k \cdot \sigma$$

  We use $k = 2.58$, corresponding to a 99% confidence interval under the assumption of normally distributed residuals.
- Calibration plots, where predictions are grouped into bins by predicted σ, and the empirical RMSE in each bin is compared to the mean predicted uncertainty. Well-calibrated models lie close to the identity line.

These metrics provide insights into the reliability and calibration of the predicted uncertainties. Beyond these standard methods, other metrics that can be applied:

- Negative Log-Likelihood (NLL): Assumes predictions follow a normal distribution and computes the average log-likelihood under that assumption:

$$\text{NLL} = \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2}\left(\frac{y - \hat{y}}{\sigma}\right)^2$$

- Expected Normalized Calibration Error (ENCE) [29]: Measures average deviation between predicted standard deviations and empirical errors after standardization:

$$\text{ENCE} = \frac{1}{B}\sum_{b=1}^{B}\left|\frac{\text{RMSE}_b - \overline{\sigma}_b}{\overline{\sigma}_b}\right|$$

  where each bin b groups data by predicted σ.
- Standardized residual analysis: A well-calibrated model will produce standardized residuals $\frac{y-\hat{y}}{\sigma}$ that follow a standard normal distribution $\mathcal{N}(0, 1)$. Deviations from this can be quantified using histogram comparisons, Kolmogorov–Smirnov tests, or visualized via QQ plots.

7.10 Key contributions

- We integrated **predictive uncertainty estimation** into the SOAP-GAP framework using the Metatensor infrastructure, enabling rigorous error quantification for both energy and force predictions.
- Our model supports **heteroscedastic noise modeling**, allowing per-structure and per-atom regularization. This improves robustness and flexibility when learning from multi-fidelity datasets.
- The implementation is modular, efficient, and is well-suited for **active learning** workflows where uncertainty-driven sampling is critical.

8 Conclusion

This work introduces a heteroscedastic GPR-based GAP model for predicting ground-state properties, such as energies and forces. By integrating quantitative error information directly into the training process, we achieve improvements in both predictive accuracy and computational efficiency.

In this work, we present the following contributions:

- Development of a heteroscedastic GPR model capable of handling mixed-fidelity datasets, incorporating error directly into the learning process.
- Demonstration of its effectiveness on simple examples: numerical integration and equation of state prediction for symmetric silicon structure.
- Extension and validation of the methodology across a broader range of silicon structures, confirming that heteroscedastic GPR-based GAP improves the predictive power of MLIPs while reducing data generation costs.

9 Ongoing work and future steps

Building on the findings of this study, our ongoing research aims to further improve the efficiency of the heteroscedastic GPR framework for error-informed materials modeling. Continuing the work with the GAP dataset from [7], we are pursuing several key directions, each expected to result in a publication.

9.1 Error-informed GPR model

- Incorporate discretisation error estimates as part of $\sigma(\mathbf{x_i})$ and test different noise models, for example:
$$\sigma(\mathbf{x_i}) = \alpha \cdot \delta Q(\mathbf{x_i}) + \beta, \tag{27}$$
where α is a bias hyperparameter, β is a scaling coefficient, and $\delta Q(\mathbf{x_i})$ represents an estimated discretization error from [12].
- Evaluate vacancy formation energy and vacancy migration as target quantities to assess the accuracy and robustness of the heteroscedastic GPR-based GAP model.
- Implement Bayesian active learning techniques to adaptively select the most informative training points, minimizing the need for high-fidelity DFT calculations.
- Explore sparsification techniques to select representative points for sparse GPR models dealing with large-scale mixed-fidelity datasets.
- Provide long-term access to the silicon dataset computed with different cutoff energies as a heterogeneous dataset, making it available in an online repository.

9.2 Beyond independent noise assumption

- Develop GPR model to simultaneously predict a DFT quantity and its discretization error. In this case, consider one GP for the ground-state properties:
$$Q(\mathbf{x}) \sim \mathcal{GP}(\mu_q(\mathbf{x}), k_q(\mathbf{x}, \mathbf{x}')), \tag{28}$$
and another GP as discretization error:
$$\delta Q(\mathbf{x}) \sim \mathcal{GP}(\mu_\delta(\mathbf{x}), k_\delta(\mathbf{x}, \mathbf{x}')), \tag{29}$$
- Consider latent variable Gaussian process regression, which extends GPR by introducing latent variables that modulate the covariance function, making it suitable for modeling non-stationary and multi-modal processes [30].

9.3 Uncertainty-aware multitask GPR model

- Develop a multitask GPR framework that simultaneously models multiple fidelity levels by leveraging correlations between data points obtained at different plane-wave cutoffs.
- Explore error-informed multitask GPR, where low-fidelity predictions are refined using a hierarchical model:

$$y^{(h)}(x) = \rho y^{(l)}(x) + \delta(x), \tag{30}$$

  where $y^{(h)}$ is the high-fidelity output, $y^{(l)}$ is the low-fidelity prediction, ρ is a scaling hyperparameter, and δ represents the residual correction.

9.4 Datasets

- Use large-scale dataset of 6 virtual oxide structures and 4 unary systems covering each element of the periodic table [17]. This dataset is now integrated into the AiiDA workflow manager [31, 32] and is accessible from DFTK.

By addressing these challenges, our goal is to create an error-aware modeling approach that improves computational efficiency and prediction reliability across diverse materials systems.

A Appendix

A.1 Equivalence of SR and GPR

First of all, we recall the *matrix inversion lemma*, also known as the Woodbury matrix identity [33], which provides an efficient way to compute the inverse of a matrix perturbed by a low-rank update. The lemma is stated as follows:

Lemma 1 (Matrix Inversion Lemma) *Let $Z \in \mathbb{R}^{n\times n}$, $W \in \mathbb{R}^{m\times m}$, and $U, V \in \mathbb{R}^{n\times m}$. Then, assuming the relevant inverses exist, we have:*

$$(Z + UWV^\top)^{-1} = Z^{-1} - Z^{-1}U\left(W^{-1} + V^\top Z^{-1}U\right)^{-1}V^\top Z^{-1}. \tag{L.1}$$

With this tool in mind, we now proceed to establish the equivalence between the SR and the standard GPR predictive equations.

A.1.1 Predictive mean

The predictive mean of the GPR is given by $\mathbb{E}[f(\mathbf{x}_*)] = \mathbf{k}_*^\top(K+\Sigma)^{-1}\mathbf{y}$. Replacing all the occurrences of $k(\mathbf{x},\mathbf{x}')$ with $\tilde{k}(\mathbf{x},\mathbf{x}')$, we obtain the following:

$$\begin{aligned}\mathbb{E}[\tilde{f}(\mathbf{x}_*)] &= \tilde{\mathbf{k}}(\mathbf{x}_*)^\top(\tilde{K}+\Sigma)^{-1}\mathbf{y}, &(31)\\ &= \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1}K_{mn}(K_{nm}K_{mm}^{-1}K_{mn}+\Sigma)^{-1}\mathbf{y}. &(32)\end{aligned}$$

Using the matrix inversion lemma (L.1), we can rewrite:

$$(K_{nm}K_{mm}^{-1}K_{mn}+\Sigma)^{-1} = \Sigma^{-1} - \Sigma^{-1}K_{nm}Q^{-1}K_{mn}\Sigma^{-1}, \tag{33}$$

where $Q = K_{mm} + K_{mn}\Sigma^{-1}K_{nm}$. Therefore,

$$\begin{aligned}\mathbb{E}[\tilde{f}(\mathbf{x}_*)] &= \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1}K_{mn}\left[\Sigma^{-1} - \Sigma^{-1}K_{nm}Q^{-1}K_{mn}\Sigma^{-1}\right]\mathbf{y}, &(34)\\ &= \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1}K_{mn}\left[I_n - \Sigma^{-1}K_{nm}Q^{-1}K_{mn}\right]\Sigma^{-1}\mathbf{y}, &(35)\\ &= \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1}\left[K_{mn}\cdot I_n - K_{mn}\Sigma^{-1}K_{nm}Q^{-1}K_{mn}\right]\Sigma^{-1}\mathbf{y}. &(36)\end{aligned}$$

Since $K_{mn}\cdot I_n = I_m \cdot K_{mn}$, we have:

$$\begin{aligned}\mathbb{E}[\tilde{f}(\mathbf{x}_*)] &= \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1}\left[I_m - K_{mn}\Sigma^{-1}K_{nm}Q^{-1}\right]K_{mn}\Sigma^{-1}\mathbf{y}, &(37)\\ &= \mathbf{k}_m(\mathbf{x}_*)^\top K_{mm}^{-1}\left[K_{mm}Q^{-1}\right]K_{mn}\Sigma^{-1}\mathbf{y}, &(38)\end{aligned}$$

using $I_m = QQ^{-1} = (K_{mm} + K_{mn}\Sigma^{-1}K_{nm})Q^{-1}$. Finally, we get:

$$\begin{aligned}\mathbb{E}[\tilde{f}(\mathbf{x}_*)] &= \mathbf{k}_m(\mathbf{x}_*)^\top Q^{-1}K_{mn}\Sigma^{-1}\mathbf{y}, &(39)\\ &= \mathbf{k}_m(\mathbf{x}_*)^\top(K_{mm}+K_{mn}\Sigma^{-1}K_{nm})^{-1}K_{mn}\Sigma^{-1}\mathbf{y}, &(40)\end{aligned}$$

which corresponds to the SR predictive mean, as given in equation (18).

A.1.2 Predictive variance

The predictive variance of the standard GPR with noise covariance Σ is:

$$\text{Var}[f(\mathbf{x}_*)] = k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top(K+\Sigma)^{-1}\mathbf{k}_*.$$

Replacing $k(\mathbf{x},\mathbf{x}')$ with the approximated kernel $\tilde{k}(\mathbf{x},\mathbf{x}')$, we get:

$$\text{Var}[\tilde{f}(\mathbf{x}_*)] = \tilde{k}(\mathbf{x}_*,\mathbf{x}_*) - \tilde{\mathbf{k}}(\mathbf{x}_*)^\top(\tilde{K}+\Sigma)^{-1}\tilde{\mathbf{k}}(\mathbf{x}_*). \tag{41}$$

Substituting $\tilde{k}(\mathbf{x},\mathbf{x}') = \mathbf{k}_m(\mathbf{x})^\top K_{mm}^{-1}\mathbf{k}_m(\mathbf{x}')$ and following similar steps as above, the predictive variance becomes:

$$\text{Var}[\tilde{f}(\mathbf{x}_*)] = \mathbf{k}_m(\mathbf{x}_*)^\top(K_{mm}+K_{mn}\Sigma^{-1}K_{nm})^{-1}\mathbf{k}_m(\mathbf{x}_*). \tag{42}$$

References

[1] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin A. Persson. The materials project: A materials genome approach to accelerating materials innovation. *Nature Reviews Materials*, 1:15004, 2016.

[2] Shunxi Luo, Ting Li, Xue Wang, et al. Machine learning for quantum mechanical properties of atoms and molecules. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 11(6):e1489, 2021.

[3] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin A. Persson. The materials project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013.

[4] Stefano Curtarolo, Wahyu Setyawan, Jianpeng Wang, Jihui Xue, Kesong Yang, Richard H. Taylor, Lance J. Nelson, Gus L.W. Hart, Stefano Sanvito, Marco Buongiorno Nardelli, Natalio Mingo, and Ohad Levy. Aflow: An automatic framework for high-throughput materials discovery. *Computational Materials Science*, 58:218–226, 2012.

[5] Keith T. Butler, Daniel W. Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Data-driven materials science: Status, challenges, and perspectives. *Nature*, 559:547–555, 2018.

[6] Keith T. Butler, Daniel W. Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559:547–555, 2018.

[7] Albert P. Bartók, James Kermode, Noam Bernstein, and Gábor Csányi. Machine learning a general-purpose interatomic potential for silicon. *Phys. Rev. X*, 8:041048, Dec 2018.

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

[9] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and Weinan E. Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics. *Physical Review Letters*, 120(14):143001, 2018.

[10] Felix Musil, Andrea Grisafi, Albert P. Bartók, Christoph Ortner, Gábor Csányi, and Michele Ceriotti. Physics-inspired structural representations for molecules and materials. *Chemical Reviews*, 121(16):9759–9815, Aug 2021.

[11] Richard M. Martin. *Electronic Structure: Basic Theory and Practical Methods*. Cambridge University Press, 2 edition, 2020.

[12] Eric Cancès, Geneviève Dusson, Gaspard Kemlin, and Antoine Levitt. Practical error bounds for properties in plane-wave electronic structure calculations. *SIAM Journal on Scientific Computing*, 44(5):B1312–B1340, 2022.

[13] C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[14] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi. Machine-learning of atomic-scale properties based on physical principles. *Physical Review Letters*, 104(13):136403, 2010.

[15] Ryosuke Jinnouchi, Ferenc Karsai, and Georg Kresse. On-the-fly active learning of interpretable bayesian force fields for atomistic rare events. *Physical Review Letters*, 122(22):225701, 2019.

[16] Kristian Kersting, Clemens Plagemann, Patrick Pfaff, and Wolfram Burgard. Most likely heteroscedastic gaussian process regression. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 393–400, 2007.

[17] E. Bosoni, L. Beal, M. Bercx, et al. How to verify the precision of density-functional-theory implementations via reproducible and universal workflows. *Nature Reviews Physics*, 6:45, 2023.

[18] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Physical Review*, 140(4A):A1133–A1138, 1965.

[19] N. Troullier and J. L. Martins. Efficient pseudopotentials for plane-wave calculations. *Physical Review B*, 43(3):1993–2006, 1991.

[20] D. Vanderbilt. Soft self-consistent pseudopotentials in a generalized eigenvalue formalism. *Physical Review B*, 41(11):7892–7895, 1990.

[21] Petros Drineas and Michael W. Mahoney. On the nystrom method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

[22] William Tebbutt, Marc Finzi, David R. Burt, et al. Abstractgps.jl: A lightweight, intuitive gaussian process library for julia, 2021. Version 0.X, accessed on [DATE].

[23] Dftk features documentation. https://docs.dftk.org/stable/features/, 2024. Accessed: 2024-11-12.

[24] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical Review Letters*, 104:136403, 2010.

[25] A. P. Bartók, R. Kondor, and G. Csányi. On representing chemical environments. *Physical Review B*, 87(18):184115, 2013.

[26] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Physical Review Letters*, 77(18):3865–3868, 1996.

[27] M. J. van Setten, M. Giantomassi, E. Bousquet, M. J. Verstraete, D. R. Hamann, X. Gonze, and G.-M. Rignanese. The pseudodojo: Training and grading a 85 element optimized norm-conserving pseudopotential table. *Computer Physics Communications*, 226:39–54, 2018.

[28] Metatensor Contributors. Metatrain: Training and evaluating machine learning models for atomistic systems. https://github.com/metatensor/metatrain, 2025. Accessed: 2025-01-31.

[29] Leonard Foster, Alex Waagen, Nisar Aijaz, Mark Hurley, Alex Luis, Jason Rinsky, Chandrika Satyavolu, Ananth Srivastava, Minerva Tu, and Charles Varner. Stable and efficient gaussian process calculations. *Journal of Machine Learning Research*, 10:857–882, 2009.

[30] Erik Bodin, Markus Kaiser, Ieva Kazlauskaite, Zhenwen Dai, Neill Campbell, and Carl Henrik Ek. Modulating surrogates for bayesian optimization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 970–979. PMLR, 13–18 Jul 2020.

[31] S. P. Huber, S. Zoupanos, M. Uhrin, et al. Aiida 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance. *Scientific Data*, 7:300, 2020.

[32] M. Uhrin, S. P. Huber, J. Yu, et al. Workflows in aiida: Engineering a high-throughput, automation-ready environment for computational science. *Computational Materials Science*, 187:110086, 2021.

[33] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.