

Computer Programs in Physics

GPR_calculator: An on-the-fly surrogate model to accelerate massive nudged elastic band calculations

Isaac Onyango^{a,b}, Byungkyun Kang^c, Qiang Zhu^{a,b,*}^a Department of Mechanical Engineering and Engineering Science, University of North Carolina at Charlotte, Charlotte, NC 28223, USA^b North Carolina Battery Complexity, Autonomous Vehicle and Electrification (BATT CAVE) Research Center, Charlotte, NC, 28223, USA^c College of Arts and Sciences, University of Delaware, Newark, DE 19716, USA

ARTICLE INFO

The review of this paper was arranged by Prof. Blum Volker

Dataset link: https://github.com/MaterSim/GPR_calculator

Keywords:

Machine learning descriptor
Transition state theory
Gaussian process regression

ABSTRACT

We present GPR_calculator, a package based on Python and C++ programming languages to build an on-the-fly surrogate model using Gaussian Process Regression (GPR) to approximate computationally expensive electronic structure calculations. The key idea is to dynamically train a GPR model during the simulation that can accurately predict energies and forces with uncertainty quantification. When the uncertainty is high, the costly electronic structure calculation is performed to obtain the ground truth data, which is then used to update the GPR model. To illustrate the effectiveness of GPR_calculator, we demonstrate its application in Nudged Elastic Band (NEB) simulations of surface diffusion and reactions, achieving 3-10 times acceleration compared to pure ab initio calculations. The source code is available at https://github.com/MaterSim/GPR_calculator.

Program summary

Program Title: GPR_calculator

CPC Library link to program files: <https://doi.org/10.17632/vyhpdf9fkh.1>

Licensing provisions: MIT [1]

Programming language: Python 3 & C++

Nature of problem: Many atomistic simulations—such as geometry optimization, barrier calculations, molecular dynamics, and equation-of-state simulations—require sampling a large number of atomic configurations in a compact phase space. While Density Functional Theory (DFT) provides good accuracy and relatively scalable performance for systems with fewer than hundreds of atoms, it can become prohibitively expensive for massive simulations. This is particularly evident in energy barrier calculations for surface diffusion or reaction studies, where hundreds or thousands of energy and force evaluations are needed.

Solution method: The GPR_calculator is an On-the-Fly Atomistic Calculator based on Gaussian Process Regression (GPR), designed as an add-on module that can be used with the popular Atomic Simulation Environment (ASE). It is essentially a hybrid approach that consists of: (i) a base calculator to provide ground truth reference energy and forces for the given input structure, and (ii) a surrogate model serving as the less expensive approximation trained on-the-fly. When the uncertainty of the GPR prediction exceeds a user-defined threshold, the base calculator is invoked to obtain accurate results and update the GPR model. This adaptive approach ensures accuracy while significantly reducing computational cost.

References

- [1] <https://opensource.org/licenses/MIT>

1. Introduction

The role of computational modeling approaches has become increasingly important in the design of catalysts. Developing reliable computa-

tional models requires a fundamental understanding of catalytic surface reactions, particularly the relevant elementary steps and their kinetics (including the activation barriers and reaction rates). A common approach to study surface reactions is to identify the minimum energy

* Corresponding author.

E-mail address: qzhu8@charlotte.edu (Q. Zhu).

<https://doi.org/10.1016/j.cpc.2025.109781>

Received 13 May 2025; Received in revised form 26 June 2025; Accepted 22 July 2025

pathway (MEP), where the energy maximum (saddle point) along this path gives the activation barrier. This barrier is then used to calculate kinetic rates within the harmonic transition state theory (HTST) framework [1].

In solid state chemistry, the nudged elastic band (NEB) method [2–5], with its extension of the solid state phase transition [6–8], is arguably the most popular approach used to search for MEPs. Generally speaking, the NEB method is a refinement of the chain of states method in which the path between two local minima is initialized as a discrete chain of configurations, commonly referred to as *images*, generated by interpolation between the two local minima. The path is then optimized by iteratively minimizing an objective function defined as the sum of the energies of the images plus a Hookean energy term as a penalty that keeps the images distributed along the path through. However, optimization of this objective function tends to lead to *corner cutting* where the spring force perpendicular to the path pulls images off the MEP leading to over estimation of the activation barrier if the spring constant is too large. If the spring constant is too small, the net spring force acts to balance the true force (negative gradient of the energies of the images) parallel to the path making the images slide toward lower-energy regions (*sliding down*) and avoid the activation barrier. NEB alleviates the *corner cutting* and *sliding down* issues by projecting out the spring forces that are perpendicular to the path and the true force components parallel to the path during optimization.

In a typical NEB simulation, 5–10 images are used, with their positions iteratively adjusted during the optimization. Converging to the desired MEP often requires hundreds of optimization steps, translating to hundreds of energy and force evaluations per image using accurate electronic structure methods, most commonly based on density functional theory (DFT). The majority of computational effort is dedicated to these DFT calculations. Each electronic structure evaluation typically takes tens or even hundreds of CPU minutes, making NEB calculations very demanding. Consequently, a single NEB simulation may take several hours to days to complete. Furthermore, surface chemical reactions often involve multiple elementary steps, each requiring its own NEB calculation, further increasing computational costs. However, the accuracy of the energy landscape, where most of this computational effort is spent, is not the primary concern; rather, the key objective is to identify the saddle point. Thus, leveraging an approximate energy landscape that still enables saddle point identification could significantly reduce computational costs.

Over the past several years, various efforts have been made to accelerate NEB simulations by using the surrogate machine learning force field (MLFF) models as substitutes for expensive DFT calculations. Peterson et al. [9] developed a neural network (NN) approach using the Behler-Parrinello descriptors [10] to represent the local atomic environment for the images. While this leads to significantly faster simulations, it requires an iterative data collection procedure to train the model. After the model is built, it lacks an in-time validation mechanism and one cannot infer the uncertainty of the simulation results.

Jonsson et al. [11,12] proposed a Gaussian Process Regression (GPR) model, where the covariance matrix was constructed using a radial basis function (RBF) kernel based on relative Cartesian distances and inverse interatomic distances. This approach leverages the uncertainty estimates from GPR, selectively adding only the most uncertain images to the training dataset, thereby improving the efficiency of data collection. However, the model refinement closely follows that of Peterson et al. [9], with the key distinction that only the image with the highest uncertainty estimate from the GPR is added to the training dataset after validating the final images in the GPR-optimized MEP against *ab initio* calculations.

Following Jonsson [11,12], several other GPR-based approaches have been introduced. In an effort to reduce the amount of training data, Torres et al. [13] initiated the GPR training with an image located one-third of the way between the initial and final states in the initial MEP guess. The GPR is then used to optimize the MEP, and an image

with either the highest uncertainty or the highest energy along the optimized path is selected to update the GPR model. Teng et al. [14,15] introduced prior mean functions — either force-field-based or classical mechanical descriptions — into the GPR, in contrast to the widely used zero-mean GPR models. These prior means provide an initial estimate of the energy landscape, improving the performance of the GPR model. Furthermore, the prior means are not fixed and can be adjusted dynamically throughout the calculation. More recently, Schaaf et al. [16] developed a protocol aimed at creating a general MLFF capable of accurately predicting a broad range of surface phenomena, including molecular adsorption, surface rearrangement, and both single-step and multi-step reactions. Before beginning NEB or geometry optimizations, the MLFF is trained using molecular dynamics.

Broadly speaking, all these approaches can be categorized as *one-shot* MLFF models, where the MLFF is trained using different strategies, employed to optimize the MEP and subsequently updated by validating the final images against their *ab initio* calculations. While such *one-shot* MLFFs (either based on NN or GPR models) are becoming increasingly popular, they may not be suitable for exploratory simulation tasks that demand very high accuracy.

To address this challenge, we propose an on-the-fly approach where MLFF models are dynamically updated during NEB calculation thus integrating both the GPR model and DFT (*ab initio*) calculations during MEP optimization. This approach is inspired by Jinnouchi et al. [17] who implemented the on-the-fly strategy to speed up *ab initio* molecular dynamics (MD) simulations for the study of phase transition, electrochemical reaction and other complex phenomena. Additionally, we integrate our approach with the Atomic Simulation Environment (ASE) [18] to ensure compatibility with a wide variety of electronic structure calculation methods besides DFT. In the following sections, we will provide a detailed description of the GPR model and its implementation in the `GPR_calculator` package. We will also demonstrate its application in NEB simulations of surface diffusion and reactions, showcasing its ability to achieve 3–10 times acceleration compared to pure *ab initio* calculations.

2. Computational methodology

2.1. Gaussian process regression

Suppose we have a set of sample data $\{\mathbf{x}, Y\}$, where \mathbf{x} represents the input vector and Y is the corresponding learning target. The GPR model assumes that the target values are drawn from a Gaussian process characterized by a kernel function $k(\mathbf{x}, \mathbf{x}')$ [19]. This kernel function defines the relationships between input vectors. Among various kernel functions, the RBF kernel is widely used and it is defined as:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \theta^2 \exp[-(\mathbf{x}_m - \mathbf{x}_n)^2 / 2l^2] \quad (1)$$

Further, the covariance matrix \mathbf{C} is defined as:

$$\mathbf{C}_{mn} = \mathbf{C}(\mathbf{x}_m, \mathbf{x}_n) = k(\mathbf{x}_m, \mathbf{x}_n) + \beta \delta_{mn}, \quad (2)$$

where β is the noise variance, and δ_{mn} is the Kronecker delta function. For N samples, \mathbf{C} is a square matrix of $N \times N$. Each sample value can be considered as the linear combination of these covariances.

$$\mathbf{Y}_m = \sum_{i=1}^N \mathbf{G}_{ai} \mathbf{C}(\mathbf{x}_m, \mathbf{x}_i) \quad (3)$$

Hence, one only needs to determine \mathbf{G}_a from the previous training data. However, a computational bottleneck in implementing GP regression is the $O(N^3)$ computational complexity associated with inverting or factorizing the \mathbf{C} . When \mathbf{C}_{mn} is smooth, the covariance matrix can be expected to be rank-deficient, i.e., its eigenvalues are likely to decay rather rapidly. A reasonable rank- M approximation to \mathbf{C}_N can be obtained by switching off its last $N - M$ eigenvalues to zero. Therefore

it is possible to reduce the computational time to in $O(NM^2)$ time, by Cholesky factorization of C to its lower triangular matrix L ,

$$C = LL^T. \quad (4)$$

And then solve the typical linear equation of $CG_\alpha = Y$ from L .

For a new input x_{N+1} , C can be expanded as follows,

$$C_{N+1} = \begin{pmatrix} C_N & \mathbf{k} \\ \mathbf{k}^T & c \end{pmatrix} \quad (5)$$

where $c = k(x_{N+1}, x_{N+1}) + \beta$. And the vector \mathbf{k} has elements $k(x_n, x_{N+1})$ for $n = 1, \dots, N$.

Thus the mean and variance of the new sample can be derived as:

$$Y_{N+1} = C_{N+1} G_\alpha \quad (6)$$

$$\sigma_{N+1}^2 = c - \mathbf{k}^T C_N^{-1} \mathbf{k} \quad (7)$$

We can further improve the performance by optimizing θ and l leading to the maximum log-likelihood (MLL) function. Assuming that the Gaussian noise (ϵ_n) is applied to the observed y_n ,

$$t_n = y_n + \epsilon_n. \quad (8)$$

The MLL can be evaluated from the following [19],

$$\ln p(t|\theta) = -\frac{1}{2} \left[\ln \det(C_N) + t^T C_N^{-1} t + N \ln(2\pi) \right] \quad (9)$$

To obtain the best MLL, one can generate multiple trial l values and then optimize each with the gradient information. In Python programming, this step can be conveniently done by calling the `scipy.optimize` library and selecting a fast optimization algorithm like L-BFGS-B [20].

$$\frac{\partial}{\partial \theta} \ln p(t|\theta) = -\frac{1}{2} \left[\text{Tr} \left(C_N^{-1} \frac{\partial C_N}{\partial \theta} \right) - t^T C_N^{-1} \frac{\partial C_N}{\partial \theta} C_N^{-1} t \right] \quad (10)$$

2.2. Structural descriptor

Recently, several structural descriptors [10,21,22] have been proposed to characterize the local atomic environment by observing the translation, rotation and permutation symmetry invariance. Following our previous works [23,24], we focus on the use of SO(3) descriptor derived from the power-spectrum of spherical harmonic expansion coefficients. The main idea here is to use the combined orthonormal radial and angular basis functions to reconstruct the smoothed neighbor density function

$$\rho(\mathbf{r}) = \sum_i^{r_i \leq r_c} f_{\text{cut}}(r_i) e^{(-\alpha|r-\mathbf{r}_i|^2)} \quad (11)$$

where α (typical chosen as 1-2 Å) is the parameter to control the smoothness of the neighbor density at a given location, and f_{cut} is a cutoff function to ensure the density smoothly decays to zero when r_i approaches r_c .

$$f_{\text{cut}}(r) = \begin{cases} \frac{1}{2} \left[\cos\left(\frac{\pi r}{r_{\text{cut}}}\right) + 1 \right], & r \leq r_{\text{cut}} \\ 0, & r > r_{\text{cut}} \end{cases} \quad (12)$$

According to Refs. [21,23,24], the weighted neighbor density function can be expanded as

$$\rho(\mathbf{r}) = \sum_{nlm} c_{nlm} g_n(r) Y_{lm}(\hat{\mathbf{r}}), \quad (13)$$

where $g_n(r)$ is a set of orthonormal radial basis function derived from the polynomials, Y_{lm} is the spherical harmonics, r is the radial distance, $\hat{\mathbf{r}}$ is the unit vector of \mathbf{r} . And the expansion coefficients c_{nlm} have three indices to denote the contributions from radial basis (n) and the spherical harmonics (l and m), respectively.

The analytical expression of c_{nlm} can be derived as follows,

$$c_{nlm} = 4\pi \sum_i^{r_i \leq r_c} f_{\text{cut}}(r_i) e^{-\alpha r_i^2} Y_{lm}^*(\hat{\mathbf{r}}_i) \int_0^{r_c} r^2 g_n(r) e^{-\alpha r^2} I_l(2\alpha r r_i) dr, \quad (14)$$

in which I_l is the modified spherical Bessel function of the first kind.

The c_{nlm} coefficients are complex-valued and rotation variant. To ensure the rotation-invariance, we take its power spectrum combining these expansion coefficients:

$$x_{n_1 n_2 l} = \sum_{m=-l}^{+l} c_{n_1 l m} c_{n_2 l m}^*. \quad (15)$$

Hence, each atom in the structure can be described as an $n_{\text{max}} \times n_{\text{max}} \times l_{\text{max}}$ array. After symmetry reduction, this can be further reduced to an 1D array of $n_{\text{max}} \times (n_{\text{max}} + 1) \times (l_{\text{max}} + 1)/2$. In addition, the derivative of \mathbf{x} with respect to all atomic coordinates ($\partial \mathbf{x} / \partial \mathbf{R}$) will be needed for the computation of forces. The full expression and derivations can be found in Ref. [23].

For practical applications, we choose $n_{\text{max}} = 3$, and $l_{\text{max}} = 4$, resulting in a 30-length array to represent a single atomic environment (\mathbf{x}_i). If a cutoff distance 5-6 Å is chosen, one expects to find 30-50 neighboring atoms, and thus each atom's $\partial \mathbf{x} / \partial \mathbf{R}$ array is expected to have a size of (30, 50, 3).

2.3. Gaussian Kernel choices

After the structure descriptor is known, we define the similarity between two atoms (encoded as \mathbf{x}_1 and \mathbf{x}_2) from the cosine distance:

$$d(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{|\mathbf{x}_1| |\mathbf{x}_2|}, \quad (16)$$

resulting a scalar between 0 and 1. To improve the distinction capability, one can further apply the power law to get the distance metric (D),

$$D(\mathbf{x}_1, \mathbf{x}_2) = d^\zeta \quad (17)$$

where ζ is a integer to control the distribution.

The corresponding derivatives are:

$$\begin{aligned} \frac{\partial d}{\partial \mathbf{x}_1} &= \frac{\mathbf{x}_2 |\mathbf{x}_1| - (\mathbf{x}_1 \cdot \mathbf{x}_2) \frac{\mathbf{x}_1}{|\mathbf{x}_1|}}{|\mathbf{x}_2| |\mathbf{x}_1|^2} \\ &= \frac{\mathbf{x}_2}{|\mathbf{x}_1| |\mathbf{x}_2|} - \frac{(\mathbf{x}_1 \cdot \mathbf{x}_2) \mathbf{x}_1}{|\mathbf{x}_1|^3 |\mathbf{x}_2|} \end{aligned} \quad (18)$$

$$\begin{aligned} \frac{\partial^2 d}{\partial \mathbf{x}_1 \partial \mathbf{x}_2} &= \frac{\partial}{\partial \mathbf{x}_2} \left(\frac{\mathbf{x}_2}{|\mathbf{x}_1| |\mathbf{x}_2|} - \frac{(\mathbf{x}_1 \cdot \mathbf{x}_2) \mathbf{x}_1}{|\mathbf{x}_1|^3 |\mathbf{x}_2|} \right) \\ &= \frac{\mathbf{I}}{|\mathbf{x}_1| |\mathbf{x}_2|} - \frac{\mathbf{x}_2 \otimes \mathbf{x}_2}{|\mathbf{x}_1| |\mathbf{x}_2|^3} + \frac{(\mathbf{x}_1 \cdot \mathbf{x}_2) \mathbf{x}_1 \otimes \mathbf{x}_2}{|\mathbf{x}_1|^3 |\mathbf{x}_2|^3} - \frac{\mathbf{x}_1 \otimes \mathbf{x}_1}{|\mathbf{x}_1|^3 |\mathbf{x}_2|} \end{aligned} \quad (19)$$

Following the definition of D , the derivatives are:

$$\begin{aligned} \frac{\partial D}{\partial \mathbf{x}_1} &= \zeta d^{\zeta-1} \frac{\partial d}{\partial \mathbf{x}_1} \quad \frac{\partial D}{\partial \mathbf{x}_2} = \zeta d^{\zeta-1} \frac{\partial d}{\partial \mathbf{x}_2} \\ \frac{\partial^2 D}{\partial \mathbf{x}_1 \partial \mathbf{x}_2} &= \frac{\partial}{\partial \mathbf{x}_2} \left(\frac{\partial D}{\partial \mathbf{x}_1} \right) = \frac{\partial}{\partial \mathbf{x}_2} \left(\zeta d^{\zeta-1} \frac{\partial d}{\partial \mathbf{x}_1} \right) \\ &= \zeta \left[d^{\zeta-1} \frac{\partial^2 d}{\partial \mathbf{x}_1 \partial \mathbf{x}_2} + (\zeta - 1) d^{\zeta-2} \frac{\partial d}{\partial \mathbf{x}_1} \frac{\partial d}{\partial \mathbf{x}_2} \right] \end{aligned} \quad (20)$$

Once D is determined, we can construct two types of kernel functions.

$$k(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} \theta^2 \exp[\frac{1}{2} D(\mathbf{x}_1, \mathbf{x}_2) / l^2] & \text{RBF} \\ \theta^2 [\sigma_0^2 + D(\mathbf{x}_1, \mathbf{x}_2)] & \text{Dot Product} \end{cases} \quad (21)$$

For both RBF and Dot Product kernels, we use the following relations to compute the derivatives:

$$\begin{aligned}\frac{\partial k(\mathbf{x}_1, \mathbf{x}_2)}{\partial \mathbf{x}_1} &= \frac{\partial k}{\partial D} \frac{\partial D}{\partial \mathbf{x}_1} \\ \frac{\partial k(\mathbf{x}_1, \mathbf{x}_2)}{\partial \mathbf{x}_2} &= \frac{\partial k}{\partial D} \frac{\partial D}{\partial \mathbf{x}_2}\end{aligned}\quad (22)$$

$$\frac{\partial k}{\partial D} = \begin{cases} \frac{0.5\theta^2}{l^2} \exp(-0.5(1-D)/l^2) & \text{RBF} \\ \theta^2 & \text{Dot Product} \end{cases}\quad (23)$$

$$\frac{\partial^2 k}{\partial \mathbf{x}_1 \partial \mathbf{x}_2} = \begin{cases} \frac{\partial k}{\partial D} \left[\frac{\partial^2 D}{\partial \mathbf{x}_1 \partial \mathbf{x}_2} + \frac{1}{2l^2} \frac{\partial D}{\partial \mathbf{x}_1} \frac{\partial D}{\partial \mathbf{x}_2} \right] & \text{RBF} \\ \frac{\partial k}{\partial D} \frac{\partial^2 D}{\partial \mathbf{x}_1 \partial \mathbf{x}_2} & \text{Dot Product} \end{cases}\quad (24)$$

2.4. Total energy and forces

In a structure s_1 with the associated descriptors $\{\mathbf{x}^{s_1}, \partial \mathbf{x}^{s_1} / \partial \mathbf{R}\}$, its total energy is $E = \sum_1^{N_1} E_i$, where E_i is the linear combination of some basis functions. We can define the summed energy kernel (k_{ee}) to express the covariance of energies between two structures (s_1 and s_2)

$$k_{ee}(\mathbf{x}^{s_1}, \mathbf{x}^{s_2}) = \frac{1}{N_{s_1} N_{s_2}} \sum_{i=1}^{N_{s_1}} \sum_{j=1}^{N_{s_2}} k(\mathbf{x}_i^{s_1}, \mathbf{x}_j^{s_2}), \quad (25)$$

in which N_{s_1} and N_{s_2} denote the number of atoms in each structure, respectively.

F is related to energy by:

$$\mathbf{F}_i = -\frac{\partial E_{\text{total}}}{\partial \mathbf{R}_i} = -\sum_j^{\text{all atoms}} \frac{\partial E_j}{\partial \mathbf{R}_i} \quad (26)$$

Therefore, we build the derivative kernel (k_{ff}) between \mathbf{F}_i and \mathbf{F}_j , as long as the descriptor information $\{\mathbf{x}_i, \partial \mathbf{x}_i / \partial \mathbf{R}\}$ and $\{\mathbf{x}_j, \partial \mathbf{x}_j / \partial \mathbf{R}\}$ are known.

$$\begin{aligned}k_{ff}(\mathbf{x}_i, \mathbf{x}_j) &= \sum_{i'}^{N_i} \sum_{j'}^{N_j} k \left(\frac{\partial E_i}{\partial \mathbf{x}_{i'}} \frac{\partial \mathbf{x}_{i'}}{\partial \mathbf{R}_i}, \frac{\partial E_j}{\partial \mathbf{x}_{j'}} \frac{\partial \mathbf{x}_{j'}}{\partial \mathbf{R}_j} \right) \\ &= \sum_{i'}^{N_i} \sum_{j'}^{N_j} \left(\frac{\partial \mathbf{x}_{i'}}{\partial \mathbf{R}_i} \right)^T \frac{\partial^2 k(\mathbf{x}_{i'}, \mathbf{x}_{j'})}{\partial \mathbf{x}_{i'} \partial \mathbf{x}_{j'}} \frac{\partial \mathbf{x}_{j'}}{\partial \mathbf{R}_j},\end{aligned}\quad (27)$$

where N_i and N_j denote the number of neighbors within the cutoff distances.

Next, we also need the kernel (k_{ef}) between energy E_{s_1} and \mathbf{F}_j

$$k_{ef}(\mathbf{x}^{s_1}, \mathbf{x}_j) = k \left(E_{s_1}, \frac{\partial E_j}{\partial \mathbf{R}_j} \right) = \frac{1}{N_{s_1}} \sum_{i=1}^{N_{s_1}} \sum_{j'=1}^{N_j} \frac{\partial k(\mathbf{x}_i, \mathbf{x}_{j'})}{\partial \mathbf{x}_{j'}} \frac{\partial \mathbf{x}_{j'}}{\partial \mathbf{R}_j} \quad (28)$$

Assuming there exists N_e energy points ($\mathbf{x}^{s_1}, \mathbf{x}^{s_2}, \dots$) and N_f force points ($\mathbf{x}_1^0, \mathbf{x}_2^0$), the total GPR covariance is a combination of energy/force kernels

$$\mathbf{C}_{\text{total}} = \begin{pmatrix} \mathbf{K}_{ee} & \mathbf{K}_{ef} \\ \mathbf{K}_{ef}^T & \mathbf{K}_{ff} \end{pmatrix} + \beta \delta_{mn}$$

which is a $N_c \times N_c$ square matrix ($N_c = N_e + 3 \times N_f$).

After determining the covariance matrix $\mathbf{C}_{\text{total}}$, the \mathbf{G}_α can be obtained.

To predict the energy and forces for a new structure $\{\mathbf{x}^t, \partial \mathbf{x}^t / \partial \mathbf{R}\}$, one can follow eq. (6) to get the energy and force predictions.

$$\mathbf{E}^t = \begin{bmatrix} k_{ee}(\mathbf{x}^t, \mathbf{x}^{s_1}) \\ \vdots \\ k_{ee}(\mathbf{x}^t, \mathbf{x}^{s_n}) \\ k_{ef}(\mathbf{x}^t, \mathbf{x}_1^0) \\ \vdots \\ k_{ef}(\mathbf{x}_t, \mathbf{x}_{3n}^0) \end{bmatrix} \cdot \mathbf{G}_\alpha \quad \mathbf{F}_i^t = \begin{bmatrix} k_{fe}(\mathbf{x}_i, \mathbf{x}_{s_1}) \\ \vdots \\ k_{fe}(\mathbf{x}_i, \mathbf{x}_{s_n}) \\ k_{ff}(\mathbf{x}_i, \mathbf{x}_1^0) \\ \vdots \\ k_{ff}(\mathbf{x}_i, \mathbf{x}_{3n}^0) \end{bmatrix} \cdot \mathbf{G}_\alpha \quad (29)$$

3. Algorithm implementation

To enable the GPR framework for practical NEB calculation, we have developed different modules to handle descriptor calculation, GPR regression, hybrid calculator implementation, and interfaces with existing NEB calculation tools. We also incorporate parallelization to enable fast computation.

3.1. Core modules

The package consists of five main modules that handle different aspects of the calculations, as summarized in Fig. 1:

1. `gpr_calc.descriptor`: Processes input structures and computes structural descriptors.
2. `gpr_calc.kernel`: Initializes RBF and Dot Product kernels with predefined parameters, compute different kernel functions (k_{ee} , k_{ef} , k_{ff}), and constructs the total covariance matrix $\mathbf{C}_{\text{total}}$.
3. `gpr_calc.GP`: Provides the framework for optimizing kernel parameters, predicts energies for new structures, and enables dynamic updates of the training dataset and covariance matrix.
4. `gpr_calc.calculator`: Implements the hybrid calculator that combines base electronic structure calculations with on-the-fly GPR predictions based on uncertainty thresholds.
5. `gpr_calc.NEB`: Contains utilities and interfaces with ASE's `ase.mep.NEB` module, including functions for initializing images, running NEB calculations, and plotting the results.

3.2. Acceleration and parallelization

Compared to other machine learning models such as neural networks, GPR is known to suffer from scalability issues, with computational costs scaling cubically with dataset size. In our application, additional expensive components include the computation of descriptors and kernels, particularly for k_{ff} , k_{ef} and k_{fe} computations. For large numbers of structures, descriptor computation can also become a bottleneck, especially in Python implementations.

To address these challenges while maintaining Python's flexibility and ease of use, we have implemented several optimization strategies:

First, we rewrote the kernel computation in C++ and used the `cfffi` library to interface with the Python code. This allows us to leverage C++'s computational speed while preserving Python's flexibility and rich ecosystem.

Second, we parallelized the GPR calculator using the `mpi4py` library [25]. In the GPR calculation, each processor computes the covariance matrix for a subset of the data points. This enables efficient scaling to large datasets by distributing computations across multiple processors. When a large training database is loaded, the computation of structural descriptors is also parallelized across multiple processors.

3.3. The overall NEB-GPR workflow

Fig. 2 shows the overall workflow for a practical NEB calculation using the GPR calculator. The NEB-GPR workflow consists of two main steps: (1) the NEB module handles the input images to generate initial trajectory and then optimize the trajectory based on the calculated energy and forces for each image; (2) the GPR calculator predicts the energy and forces for each image in the NEB calculation. If the GPR calculator is uncertain about its predictions, it calls the base calculator to compute the energy and forces for that image. The GPR model is then updated with the new image's energy and forces. This process continues until the NEB calculation converges.

To ensure the GPR calculator is not overly reliant on the base calculator, we set a threshold for the maximum uncertainty in energy and forces (e.g. $\sigma_E = 0.05$ eV/structure for energy and $\sigma_F = 0.075$ eV/Å for

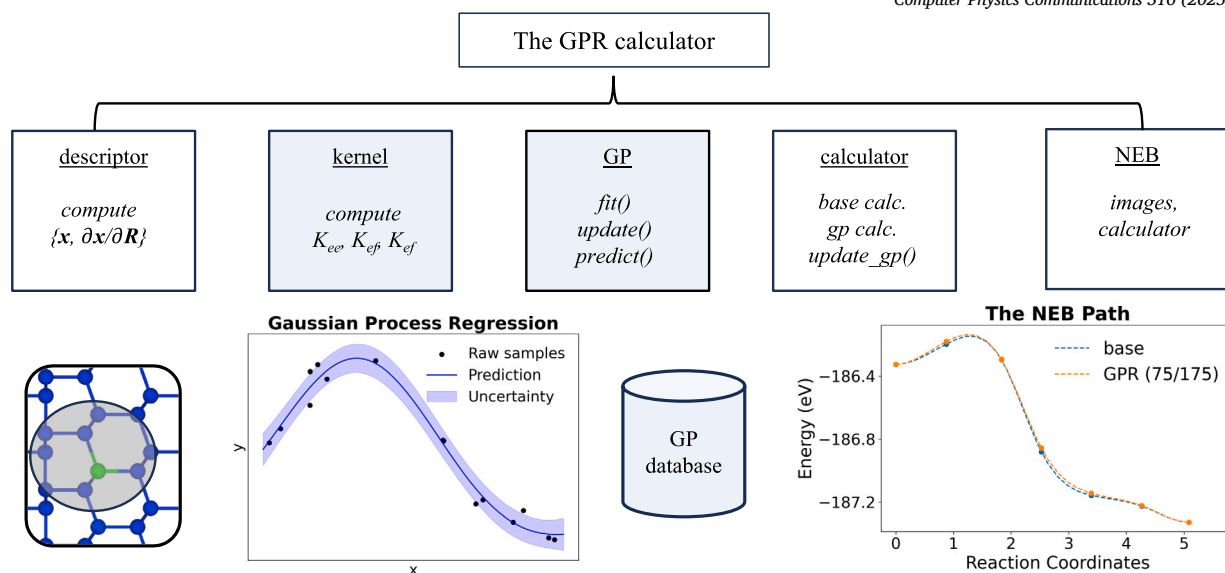


Fig. 1. The list of available modules in the `GPR_calculator` package and their corresponding outputs. The modules with acceleration are also highlighted in light blue boxes. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

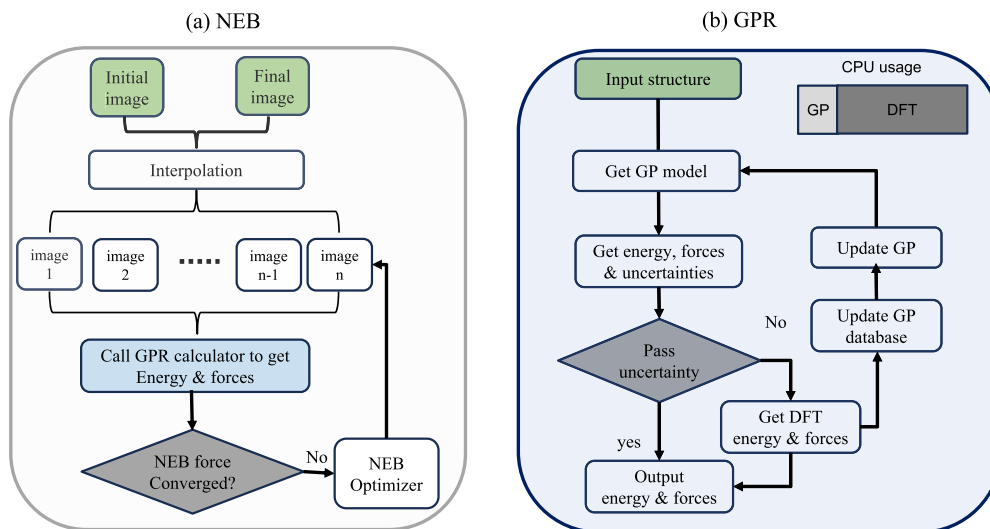


Fig. 2. The workflow of NEB-GPR calculation. (a) In a NEB run, each image's energy and forces are predicted by the GPR calculator until the NEB calculation converges. (b) The GPR calculator use either GP model or the base calculator to yield the energy and forces. If the base calculator is called, the GP model is updated with the new image's energy and forces.

forces). If the predicted uncertainty exceeds this threshold, the GPR calculator will call the base calculator to compute the energy and forces for that image. This allows us to balance the computational efficiency of the GPR calculator with the accuracy of the base calculator. Since a practical NEB calculation mainly seeks to minimize the spring forces, we choose to use only σ_F . For each single point energy calculation, the GPR calculator will check the predicted forces and compare them with the threshold (σ_F). If the predicted forces are larger than the threshold and one third of the predicted forces, the GPR calculator will call the base calculator to compute the energy and forces for that image. The DFT energy, as well as the atomic forces with large uncertainty, will be then added to the GP database and then used to update the GP model. This allows us to ensure that the NEB calculation converges to a physically meaningful result.

For NEB calculations on supercomputer nodes, we partition the available physical cores into two groups: one for the GPR calculator and the other for the base calculator. The GPR calculator is responsible for computing the covariance matrix and predicting energies, while the base

calculator handles the actual electronic structure calculations. This division of labor allows us to fully utilize the available computational resources and achieve significant speedup in NEB calculations. To avoid frequent GP model update, we usually check if there is a need to update the model at the end of each NEB iteration.

4. Dependencies

The majority of codes are written in Python 3.9 or higher. Like many other Python packages, it relies on several external Python libraries. This library is intended to interface with ASE for practical calculations and hence it inherits all package dependencies (e.g., Numpy [26], Scipy [27]) required by ASE.

In addition, the `cfi` [28] and `mpi4py` [25] libraries are required for the C++ kernel implementation and parallelization, respectively. As such, the package requires a C++ compiler and an OpenMPI implementation to be installed on the system. The package is compatible with both Linux and macOS operating systems.

5. Installation and example usages

The `GPR_calculator` package can be installed using `pip`. The package is compatible with Python 3.9 and later versions. To install, simply download the source code from https://github.com/MaterSim/GPR_calculator, change directory to the root folder and then run `pip install .` from the terminal.

The following simple example demonstrates how to set up a GPR calculator, perform a NEB calculation, and visualize the results.

```
from ase.calculators.emt import EMT
from gpr_calc.gaussianprocess import GP
from gpr_calc.calculator import GPR
from gpr_calc.NEB import neb_calc, get_images, plot_path

# Set parameters
init = 'database/initial.traj'
final = 'database/final.traj'
fmax = 0.05

# Run NEB with EMT calculator
images = get_images(init, final, 5)
neb = neb_calc(images, EMT(), fmax=fmax)
label = f'EMT ({steps*(len(imgs)-2)+2})'
dat = [(neb.images, neb.energies, label)]

# Run NEB with gpr calculators
for tol in [0.05, 0.1]:
    imgs = get_images(init, final, 5)

    # Initialize GPR model & calculator
    noise_e = tol / len(images[0])
    gp = GP.set_GPR(images, EMT(),
                    noise_e=noise_e,
                    noise_f=tol)
    calc = GPR(base=EMT(), ff=gp)

    # Run NEB calculation
    neb = neb_calc(images, calc, fmax=fmax)
    N1 = gp.use_base
    N2 = gp.use_surrogate
    label = f'GPR-{tol:.2f} ({N1}/{N2})'
    dat.append((neb.images, neb.energies, label))

# Plot the results
neb_plot_path(dat, filename='path.png',
              title='Au diffusion on Al(100)')
```

Listing 1: A Python script to perform a NEB calculation using `GPR_calculator`

In this example, we first import the necessary modules and set the initial and final structures, as well as the maximum force tolerance. We then run the NEB calculation using the effective medium theory (EMT) calculator [29] and store the results in the `dat` list. Next, we run the NEB calculation using the GPR calculator for different energy/force tolerances and store the results in the `data` list. Finally, we plot the results using the `plot_path` function.

The resulting plot in Fig. 3 illustrates the energy profile of Au diffusion on the Al(100) surface, comparing the EMT and GPR calculators. Notably, the GPR calculator with a smaller tolerance setting (0.05 eV for energy and 0.05 eV/Å for forces) produces results close to the EMT-driven NEB calculation, but requires only 8 EMT single-point energy evaluations, compared to 56 for the pure EMT calculation. Conversely, the GPR calculator with a larger tolerance setting (0.10 eV for energy and 0.10 eV/Å for forces) exhibits a more noticeable deviation.

Due to the existence of uncertainty, it is common that surrogate-driven NEB simulation requiring more iterations to converge to the desired minimum. Despite this, the GPR calculator still substantially reduces the computational effort needed for convergence. Given that DFT calculations, which are more accurate and expensive than EMT, are typically used as the base calculator, a significant speedup (3-10 times) is anticipated. For optimal performance, users can adjust the tolerance values to balance accuracy and computational efficiency.

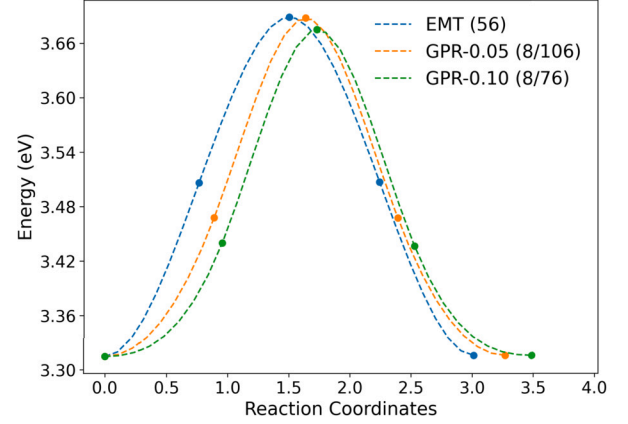


Fig. 3. The simulated Au diffusion on Al(100) surface from the EMT and GPR calculators with different energy tolerances. The number of EMT/GPR calculations required for convergence is shown in parentheses.

6. Practical applications

To demonstrate the capabilities of the `GPR_calculator` package, we focus on the use of GPR models to accelerate NEB calculations for surface diffusion and dissociation reactions, relative to traditional DFT-driven approaches. The examples presented here are intended to showcase the versatility and efficiency of the package in a variety of scenarios.

In both examples, our DFT calculations are performed using the Vienna *ab initio* simulation package (VASP) [30–32]. The core electrons and the electron-electron exchange correlation effects are treated using the projector augmented wave (PAW) [33,34] method and the Perdew-Burke-Ernzerhof (PBE) functional [35], respectively. The valence electrons are modeled using a plane-wave basis set expanded to a cutoff energy of 400 eV. The Methfessel-Paxton smearing method [36] with a smearing width of 0.1 eV, and the calculations are deemed to have converged with energy and force tolerances of 10^{-4} eV and 0.03 eV/Å, respectively. For NEB optimization, the calculations were considered to have converged when the force on the images is less than 0.075 eV/Å using the FIRE algorithm [37]. The image dependent pair potential (IDPP) method [38] is preferred over linear interpolation to generate the initial guess of the MEP because it produces a more physically meaningful pathway. IDPP prevents atoms from being placed too close to each other, which would otherwise lead to large energy and force spikes. This helps avoid potential divergence in electronic structure calculations.

In the following examples, we run each system with a single compute node of AMD EPYC 9654P 96-Core Processor with 2.40 GHz. For GPR calculations, we set up the GPR calculator with a noise of 0.05 eV/structure for energy and 0.075 eV/Å for forces, a cutoff distance of 5.0 Å for the descriptor calculation, and the energy and forces are predicted using the RBF kernel. Out of 96 cores, 24 cores are used for the GPR calculator, while the remaining 72 cores are used for the base calculator. For the pure VASP calculations, all 96 cores were used for each single point energy calculation of the NEB image.

6.1. Pd_4 cluster diffusion on $\text{MgO}(100)$ surface

We first consider the diffusion of a Pd_4 cluster on the $\text{MgO}(100)$ surface, which is a classical NEB example as discussed in the previous literature [39]. The initial state consists of a Pd_4 cluster adsorbed on the $\text{MgO}(100)$ surface, while the final state features the same cluster in a different position on the surface. $\text{MgO}(100)$ is modeled as a three-layer surface (72 atoms per layer) with atoms in the top two layers relaxed and those in the bottom two layers fixed in their bulk positions, consistent with Henkelman et al. [39]. A 15 Å vacuum is included and the Brillouin zone (BZ) is sampled using a $(2 \times 2 \times 1)$ Monkhorst-Pack k-point mesh.

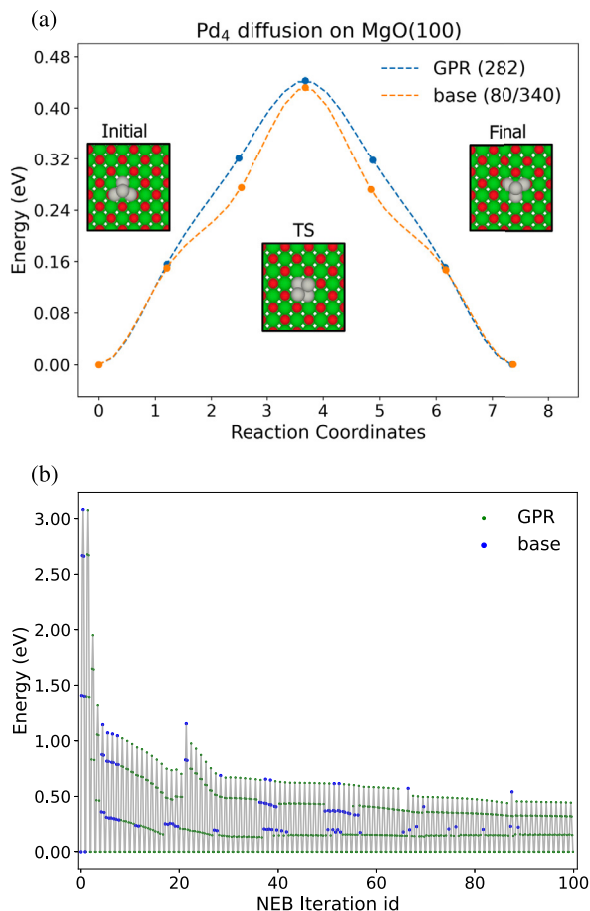


Fig. 4. (a) The simulated MEP of Pd_4 's dissociation on the $\text{MgO}(100)$ surface from both the GPR and pure VASP calculators. The representative structures along the transition path are also shown in the inset. (b) The usage of GPR and base calls during the MEP optimization. Each NEB iteration includes 7 consecutive images.

Fig. 4a shows the MEP for the diffusion of the Pd_4 cluster on the $\text{MgO}(100)$ surface, as calculated using both the GPR and pure VASP calculators. Clearly, both calculators yield nearly identical MEPs. The difference in the energy barrier is negligible: 0.44 eV from the GPR calculator compared to 0.43 eV from the pure VASP calculator. Additionally, the transition states (TS), featured as a rolling of Pd_4 to an adjacent site, are nearly identical in both cases.

Remarkably, the GPR calculator demonstrates substantial acceleration, completing the MEP optimization within 6 hours, consisting of 80 VASP calls and 340 GPR calls. As compared to the VASP results (36.5 hours consisting of 337 DFT calls), the GPR achieves resulting a 5 times speed-up. The GPR calculator's efficiency is attributed to its ability to predict energies and forces on-the-fly. In fact, the majority of NEB simulation can be achieved with less than 50 VASP calls and 110 GPR calls when the maximum NEB force is 0.09 eV/Å. The rest of simulation were mostly used to refine the final results.

For a typical diffusion energy barrier less than 0.50 eV, the GPR can achieve satisfactory accuracy with a small number of VASP calls. This example highlights the potential of the GPR calculator to accelerate NEB calculations while maintaining accuracy.

In addition to the use of self-adaptive surrogate model, one may intuitively speculate that periodically updating the model (e.g., retraining the model every 10 NEB steps) can equally guide the MEP optimization process well. However, this proposal is unlikely to work according to our analysis as shown in Fig. 4b, which plots the detailed VASP/GPR usage in each NEB iteration in our calculation. In the first 20 NEB iterations, the NEB update is mainly powered by VASP calculations, allowing

construction of a relatively accurate GPR surrogate model. Between iterations 20-40, the majority of force calculations are done by the GPR model, suggesting that the NEB calculation is exploring a suboptimal pathway with only minor structural updates. Next, some intermediate images undergo notable changes during the NEB path exploration between iterations 40-60, eventually leading to the converged path with a low energy barrier. In all cases, one needs to detect the magnitude of structural changes in time and invoke DFT calculations when necessary. Clearly, the occurrence of major structural updates is neither periodic nor uniformly distributed, making periodic model updates ineffective. On the other hand, the GPR calculator's on-the-fly model update strategy allows it to adaptively adjust to the changing energy landscape, ensuring that the NEB calculation remains efficient and accurate.

6.2. H_2S dissociation on $\text{Pd}(100)$ surface

To test the feasibility of the GPR calculator for chemical reactions, we consider a more challenging system: the dissociation of H_2S on a $\text{Pd}(100)$ surface. This system is not only of great interest in catalysis and surface chemistry, but also computationally more challenging as it involves the breaking of chemical bonds and the formation of new ones [40,41]. The initial state consists of a single H_2S molecule adsorbed on the $\text{Pd}(100)$ surface, while the final state features two H atoms and an S atom adsorbed on the surface. The $\text{Pd}(100)$ surface is modeled as a four-layer slab (9 atoms per layer) with the top two layers relaxed and the bottom two layers fixed in their bulk positions. A 15 Å vacuum is included and the Brillouin zone (BZ) is sampled using a $(3 \times 3 \times 1)$ Monkhorst-Pack k-point mesh. The NEB calculations for this system are considered to have converged when the maximum force on each image is less than 0.075 eV/Å. To ensure the saddle point is correctly identified along the NEB path, we used the climbing image method [3] instead of the regular NEB method.

Fig. 5 shows the MEP of H_2S dissociation on the $\text{Pd}(100)$ surface, calculated using the GPR calculator. For comparison, the final NEB trajectories are also recomputed with the pure VASP calculator. Clearly, the energy profiles from GPR and VASP are nearly identical (with no more than 0.015 eV for each image), confirming the accuracy of GPR model. Due to the inherent uncertainty in GPR models, we found that convergence becomes challenging when the maximum NEB forces approach 0.075 eV/Å. To validate the reliability of this GPR-optimized NEB trajectory, we performed a NEB calculation using a pure VASP calculator with the GPR-generated path as the initial MEP guess. After 25 NEB iterations, the trajectory successfully converged to a maximum NEB force below 0.05 eV/Å, exhibiting only minimal structural changes.

As shown in Fig. 5, the dissociation proceeds in two steps, with activation barriers of 0.24 eV and 0.31 eV, respectively. The mechanism involves sequential H-S bond breaking: first $\text{H}_2\text{S} \rightarrow \text{HS} + \text{H}$, followed by $\text{HS} + \text{H} \rightarrow 2\text{H} + \text{S}$. In the initial state, the H-S bond length is 1.34 Å. During dissociation, both H-S bonds gradually elongate as the hydrogen atoms separate from the sulfur. In the first step, the dissociated H-S and H fragment from H_2S initially on the top site migrates to adjacent bridge sites, with the H-S distance extending to 1.88 Å at the transition state (TS_1). In the second step, the H-S and H fragments begin at hollow sites, their most favorable adsorption positions. The H-S fragment first rotates, after which the S atom remains at the hollow site while the H migrates to second nearest neighbor bridge site, causing the H-S length to increase from 1.33 Å to 1.42 Å at TS_2 . Beyond TS_2 , the H atom follows a curved path to reach its final bridge site position.

In terms of computational efficiency, the GPR calculator completes the whole MEP optimization in 4.5 hours with 395 VASP calls and 1350 GPR calls. Given that the GPR cost is much lower than that of VASP computations, this approximately represents a 3.5 times speedup. This acceleration is slightly less than in the previous example due to the more complex and asymmetric energy surface requiring more VASP calls.

This example highlights the GPR calculator's capability to handle complex chemical reactions, while maintaining accuracy and effi-

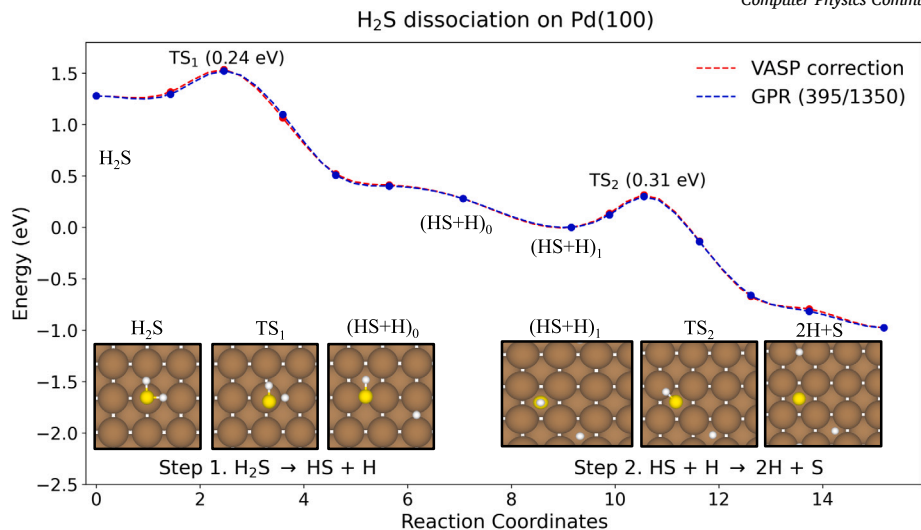


Fig. 5. The simulated MEP of H_2S 's dissociation on the Pd(100) surface from both the GPR and pure VASP calculators. The representative structures along the transition path are also shown in the inset.

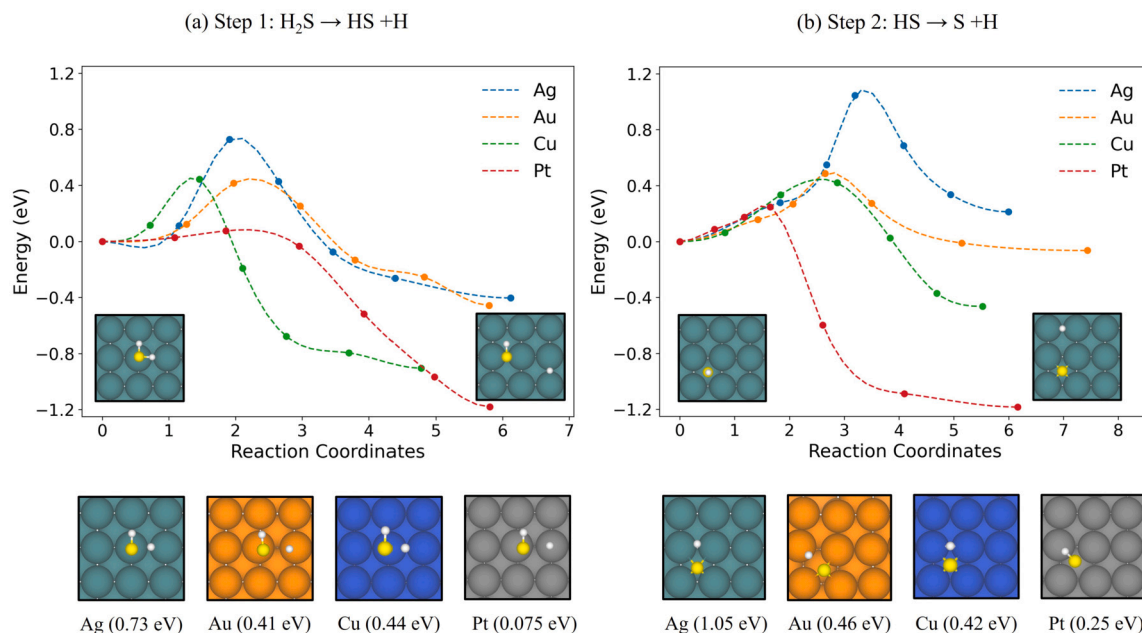


Fig. 6. The simulated MEP of H_2S dissociation on the (100) surfaces using the GPR calculator for (a) $\text{H}_2\text{S} \rightarrow \text{HS} + \text{H}$; and (b) $\text{HS} \rightarrow \text{S} + \text{H}$. In the lower panel, the transition states for each step are also shown. Green, orange, blue, gray, yellow, and white spheres represent Ag, Au, Cu, Pt, S and H atoms, respectively.

ciency. When the energy profile is more complex and it requires more NEB iterations to reach the converged pathway, the surrogate model is expected to result in a better speed up. In computational catalysis design, the ability to accurately predict transition states and reaction pathways is crucial for understanding catalytic mechanisms. The GPR_calculator's performance in this example demonstrates its potential as a valuable tool for studying complex chemical processes.

6.3. H_2S dissociation on (100) transition metal surfaces

Having demonstrated the capabilities of the GPR calculator for modeling H_2S dissociation on Pd, we extend its application to other (100) transition metal surfaces: Au, Ag, Cu, and Pt. The computational setup is similar to that of the Pd example above, with stricter criterion that NEB calculations are considered converged when the maximum force on each image falls below $0.05 \text{ eV}/\text{\AA}$ and a $(2 \times 2 \times 1)$ k-point mesh. The

whole results, including both energy profile and transition state configurations, are summarized in Fig. 6.

The first step of dissociation, as shown in Fig. 6a, is analogous to that on Pd, where H-S and H dissociated fragments from the H_2S adsorbed on a top site, migrate to adjacent bridge sites. At the transition state, the H fragment is closer to the top site for Pt, whereas it on the bridge site for Au, Ag and Cu. Additionally, the distances between S atom and the migrating H atom at the transition state increase from 1.36 \AA to 2.37 \AA , 2.12 \AA , 2.17 \AA , and 1.75 \AA for Au, Ag, Pt, and Cu, respectively. Among these, Pt exhibits the most favorable energetics for hydrogen abstraction, with a low reaction barrier of 0.075 eV and a highly exothermic reaction energy of -1.18 eV , indicating facile and strongly favorable dissociation. Cu and Au also facilitate this step, with reaction energies of -0.91 eV and -0.46 eV , and moderate barriers of 0.44 eV and 0.41 eV , respectively. Ag is the least reactive surface, with reaction energy of -0.40 eV and the highest barrier at 0.73 eV .

Table 1

The detailed GPR/VASP usage and timing for H₂S dissociation on different (100) metal surfaces on a single AMD EPYC 9654P 96-Core code.

Surface	VASP calls	GPR calls	Total time (hours)
Ag(100)	513	1912	4.60
Au(100)	691	1304	9.55
Cu(100)	317	1133	2.75
Pt(100)	683	2207	14.43

In the second dissociation step (Fig. 6b), the H-S fragment begins at a hollow site, with the sulfur atom remaining at that position while the hydrogen migrates to a bridge site. The reaction path is approximately linear on Ag and Cu but appears curved on Au and Pt, as observed on Pd. The S-H distances at the transition state increase from 1.37 Å to 1.83 Å, 2.13 Å, 1.56 Å, and 1.40 Å for Au, Ag, Cu, and Pt, respectively. This step is energetically unfavorable on Ag and Au, with Ag being endothermic (0.21 eV) and Au nearly thermoneutral (-0.07 eV), accompanied by barriers of 1.05 eV and 0.46 eV, respectively. Conversely, the reaction is exothermic on Cu(-0.46 eV) and Pt(-1.18 eV), with low barriers of 0.42 eV and 0.25 eV, respectively. As in the first step, Pt(100) exhibits the most favorable energetics, combining the lowest activation energy with the most exothermic reaction profile.

Overall, our simulation results suggest that the H₂S dissociation is less favorable on noble metals (Ag, Au) than on transition metals (Cu, Pd, Pt). Both Pd and Pt surfaces readily facilitate dissociation, characterized by low energy barriers and strongly exothermic reaction profiles. These trends align with previous DFT studies by Alfonso et al. [41], confirming the accuracy of the GPR calculator.

Table 1 summarizes the breakdowns for all calculations. Generally, the number of GPR evaluations exceeded that of VASP by at least a factor of three, with the exception of the Au(100) surface, which required slightly less than 2 times more GPR calls. This deviation likely stems from a more intricate potential energy surface associated with Au(100), characterized by more substantial atomic displacements of the surface atoms during relaxation, especially pronounced in the second dissociation step. These larger structural rearrangements make it more challenging for the GPR model to accurately interpolate and predict forces across configurations. Overall, we anticipate that the GPR calculator achieves a speedup of 3-5 times compared to pure VASP calculations.

7. Remarks on the limitation and potential serendipity

As demonstrated in the previous sections, surrogate models often require more iterations to converge. When the system approaches the minimum energy pathway, high force accuracy becomes crucial, necessitating frequent calls to the base calculator and updates to the GP model. This can lead to unnecessary optimization steps in NEB iteration. To prevent such inefficient exploration, it is recommended to set a slightly larger force tolerance (e.g., 0.075 - 0.100 eV/Å). For barrier calculations, we found that this approach typically yields activation energies within 0.05 eV error of the true value. When DFT-level accuracy is required, one can perform a quick pure DFT-NEB calculation using the GP-optimized trajectory as the initial guess, which typically converges rapidly.

On the other hand, the inherent noise in GPR models can sometimes be beneficial, particularly in preventing convergence to shallow local minima during NEB pathway optimization. In practical NEB calculations, the optimized trajectory heavily depends on the initial guess. For example, in the second step of H₂S dissociation (HS + H → 2H + S) on Pd(100), a linear dissociation path is often assumed as the initial guess. Fig. 7 shows two distinct pathways found by DFT and GPR calculations. Starting from a linear initial guess, the DFT-driven NEB converges to a linear pathway with a 0.46 eV activation barrier after 53 iterations. In

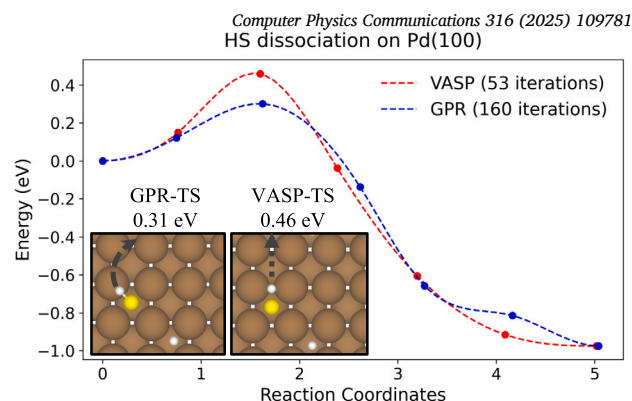


Fig. 7. The simulated MEP of HS dissociation on the Pd(100) surface from both the GPR and pure VASP calculators. The transition states are shown in the inset, with the marks of linear and curved arrows to indicate the migration pathway for the H atom. The force convergence criteria of 0.075 eV/Å was used for the NEB calculations.

contrast, our GPR model's uncertainty-driven exploration, also explored the linear pathway in the beginning, but then revealed a curved pathway with a lower activation barrier of 0.31 eV after 160 iterations. Moreover, when the curved path revealed by the GPR model was used as an initial guess for pure DFT-NEB calculation, the path retained its curvature with no structural changes to the transition state configuration, even under a tightened force convergence criterion of 0.05 eV/Å. This serendipitous finding suggests that the stochastic nature of GPR predictions can occasionally help discover unexpected yet physically meaningful reaction pathways.

8. Conclusion

In this work, we introduce `GPR_calculator`, a Python and C++ package designed to construct on-the-fly surrogate models using Gaussian Process Regression to expedite computationally intensive electronic structure calculations. A key feature of `GPR_calculator` is its ability to dynamically train a GPR model during simulations, accurately predicting energies and forces while quantifying uncertainty. The package is designed for ease of use and seamless integration with the ASE. We have demonstrated the capabilities of `GPR_calculator` through benchmark examples, showcasing its acceleration of NEB calculations for surface diffusion and reactions. Our results indicate that `GPR_calculator` can substantially decrease the computational demands of these simulations without sacrificing accuracy.

The framework is designed to be extensible, allowing users to easily add new calculators other than VASP (e.g., Quantum Espresso [42], DFTB+ [43]), new NEB optimization algorithms (e.g., L-BFGS [20]), as well as new GPR functionalities (e.g., new descriptors [21], new kernel choices [19]). In addition, the training data collected from the on-the-fly simulation can also be used for developing other flavors of machine learning force fields such as Neural Networks [9]. In the future, we will focus on improving the efficiency of the GPR model, particularly in terms of scaling to larger datasets, parallelizing the calculations over NEB images, and enhancing the accuracy of predictions. We anticipate that `GPR_calculator` will serve as a valuable asset for researchers in computational materials science and catalysis, enabling more efficient exploration of complex chemical processes.

CRedit authorship contribution statement

Isaac Onyango: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Data curation. **Byungkyun Kang:** Writing – review & editing, Writing – original draft, Software, Investigation. **Qiang Zhu:** Writing – review & editing, Writing – original draft, Supervision, Software, Project admin-

istration, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare no competing financial interest.

Acknowledgements

This research was sponsored by the U.S. Department of Energy Office of Science, Office of Basic Energy Sciences and the Established Program to Stimulate Competitive Research (EPSCoR) under the DOE Early Career Award No. DE-SC0024866. The computing resources are provided by ACCESS (TG-DMR180040). The authors thank Dr. Artur Wolek (UNC Charlotte) and Dr. Yu Lin (SLAC) for insightful discussions. During the preparation of this work the authors used GitHub Copilot in order to improve the code readability and documentation. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Data availability

The source code is available at https://github.com/MaterSim/GPR_calculator.

References

- [1] G. Gershinsky, E. Pollak, Quantum harmonic transition state theory—application to isomerization of stilbene in liquid ethane, *J. Chem. Phys.* 108 (7) (1998) 2756–2764, <https://doi.org/10.1063/1.475667>.
- [2] G. Henkelman, H. Jónsson, Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points, *J. Chem. Phys.* 113 (22) (2000) 9978–9985, <https://doi.org/10.1063/1.1323224>.
- [3] G. Henkelman, B.P. Uberuaga, H. Jónsson, A climbing image nudged elastic band method for finding saddle points and minimum energy paths, *J. Chem. Phys.* 113 (22) (2000) 9901–9904, <https://doi.org/10.1063/1.1329672>.
- [4] H.C. Herbol, J. Stevenson, P. Clancy, Computational implementation of nudged elastic band, rigid rotation, and corresponding force optimization, *J. Chem. Theory Comput.* 13 (7) (2017) 3250–3259, <https://doi.org/10.1021/acs.jctc.7b00360>.
- [5] H. Jónsson, G. Mills, K.W. Jacobsen, Nudged Elastic Band Method for Finding Minimum Energy Paths of Transitions, *World Scientific*, 1998, pp. 385–404, Ch. 16.
- [6] K.J. Caspersen, E.A. Carter, Finding transition states for crystalline solid–solid phase transformations, *Proc. Natl. Acad. Sci. USA* 102 (19) (2005) 6738–6743, <https://doi.org/10.1073/pnas.0408127102>.
- [7] D. Sheppard, P. Xiao, W. Chemelewski, D.D. Johnson, G. Henkelman, A generalized solid-state nudged elastic band method, *J. Chem. Phys.* 136 (7) (2012), <https://doi.org/10.1063/1.3684549>.
- [8] G.-R. Qian, X. Dong, X.-F. Zhou, Y. Tian, A.R. Oganov, H.-T. Wang, Variable cell nudged elastic band method for studying solid–solid structural phase transitions, *Comput. Phys. Commun.* 184 (9) (2013) 2111–2118, <https://doi.org/10.1016/j.cpc.2013.04.004>.
- [9] A.A. Peterson, Acceleration of saddle-point searches with machine learning, *J. Chem. Phys.* 145 (7) (2016) 074106, <https://doi.org/10.1063/1.4960708>.
- [10] J. Behler, M. Parrinello, Generalized neural-network representation of high-dimensional potential-energy surfaces, *Phys. Rev. Lett.* 98 (2007) 146401, <https://doi.org/10.1103/PhysRevLett.98.146401>.
- [11] O.-P. Koistinen, F.B. Dagbjartsdottir, V. Asgeirsson, A. Vehtari, H. Jónsson, Nudged elastic band calculations accelerated with Gaussian process regression, *J. Chem. Phys.* 147 (2017) 152720, <https://doi.org/10.1021/acs.jctc.9b00692>.
- [12] O.P. Koistinen, V. Asgeirsson, A. Vehtari, H. Jónsson, Nudged elastic band calculations accelerated with Gaussian process regression based on inverse interatomic distances, *J. Chem. Theory Comput.* 15 (12) (2019) 6738–6751, <https://doi.org/10.1021/acs.jctc.9b00692>.
- [13] J.A. Garrido Torres, P.C. Jennings, M.H. Hansen, J.R. Boes, T. Bligaard, Low-scaling algorithm for nudged elastic band calculations using a surrogate machine learning model, *Phys. Rev. Lett.* 122 (15) (2019) 156001, <https://doi.org/10.1103/PhysRevLett.122.156001>.
- [14] C. Teng, Y. Wang, D. Huang, K. Martin, J.B. Tristan, J.L. Bao, Dual-level training of Gaussian processes with physically inspired priors for geometry optimizations, *J. Chem. Theory Comput.* 18 (9) (2022) 5739–5754, <https://doi.org/10.1021/acs.jctc.2c00546>.
- [15] C. Teng, Y. Wang, J.L. Bao, Physical prior mean function-driven Gaussian processes search for minimum-energy reaction paths with a climbing-image nudged elastic band: a general method for gas-phase, interfacial, and bulk-phase reactions, *J. Chem. Theory Comput.* 20 (10) (2024) 4308–4324, <https://doi.org/10.1021/acs.jctc.4c00291>.
- [16] L.L. Schaaf, E. Fako, S. De, A. Schäfer, G. Csányi, Accurate energy barriers for catalytic reaction pathways: an automatic training protocol for machine learning force fields, *npj Comput. Mater.* 9 (1) (2023), <https://doi.org/10.1038/s41524-023-01124-2>.
- [17] R. Jinnouchi, F. Karsai, G. Kresse, On-the-fly machine learning force field generation: application to melting points, *Phys. Rev. B* 100 (2019) 014105, <https://doi.org/10.1103/PhysRevB.100.014105>.
- [18] A.H. Larsen, J.J. Mortensen, J. Blomqvist, I.E. Castelli, R. Christensen, M. Dułak, J. Friis, M.N. Groves, B. Hammer, C. Hargus, E.D. Hermes, P.C. Jennings, P.B. Jensen, J. Kermode, J.R. Kitchin, E.L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J.B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K.S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, K.W. Jacobsen, The atomic simulation environment—a python library for working with atoms, *J. Phys. Condens. Matter* 29 (27) (2017) 273002, <https://doi.org/10.1088/1361-648x/aa680e>.
- [19] C.M. Bishop, N.M. Nasrabadi, *Pattern Recognition and Machine Learning*, vol. 4, Springer, 2006.
- [20] C. Zhu, R.H. Byrd, P. Lu, J. Nocedal, Algorithm 778: L-bfgs-b: fortran subroutines for large-scale bound-constrained optimization, *ACM Trans. Math. Softw.* 23 (4) (1997) 550–560, <https://doi.org/10.1145/279232.279236>.
- [21] A.P. Bartók, R. Kondor, G. Csányi, On representing chemical environments, *Phys. Rev. B* 87 (2013) 184115, <https://doi.org/10.1103/PhysRevB.87.184115>.
- [22] R. Drautz, Atomic cluster expansion for accurate and transferable interatomic potentials, *Phys. Rev. B* 99 (2019) 014104, <https://doi.org/10.1103/PhysRevB.99.014104>.
- [23] D. Zagaceta, H. Yanxon, Q. Zhu, Spectral neural network potentials for binary alloys, *J. Appl. Phys.* 128 (4) (2020) 045113, <https://doi.org/10.1063/5.0013208>.
- [24] H. Yanxon, D. Zagaceta, B. Tang, D.S. Matteson, Q. Zhu, Pyxtal ff: a python library for automated force field generation, *Mach. Learn.: Sci. Technol.* 2 (2) (2020) 027001, <https://doi.org/10.1088/2632-2153/abc940>.
- [25] M. Rogowski, S. Aseeri, D. Keyes, L. Dalcin, mpi4py. Futures: Mpi-based asynchronous task execution for python, *IEEE Trans. Parallel Distrib. Syst.* 34 (2) (2022) 611–622, <https://doi.org/10.1109/TPDS.2022.3225481>.
- [26] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T.E. Oliphant, Array programming with NumPy, *Nature* 585 (7825) (2020) 357–362, <https://doi.org/10.1038/s41586-020-2649-2>.
- [27] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, Í. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0—fundamental algorithms for scientific computing in python, *arXiv e-prints*, arXiv:1907.10121, 2019.
- [28] J. Bielman, L. Oliveira, cffi-v1.16.0, <https://cffi.readthedocs.io>.
- [29] K.W. Jacobsen, P. Stoltze, J. Nørskov, A semi-empirical effective medium theory for metals and alloys, *Surf. Sci.* 366 (2) (1996) 394–402, [https://doi.org/10.1016/0039-6028\(96\)00816-3](https://doi.org/10.1016/0039-6028(96)00816-3).
- [30] G. Kresse, J. Furthmüller, Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set, *Comput. Mater. Sci.* 6 (1) (1996) 15–50, [https://doi.org/10.1016/0927-0256\(96\)00008-0](https://doi.org/10.1016/0927-0256(96)00008-0).
- [31] G. Kresse, J. Furthmüller, Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set, *Phys. Rev. B* 54 (16) (1996) 11169, <https://doi.org/10.1103/PhysRevB.54.11169>.
- [32] G. Kresse, J. Hafner, Ab initio molecular dynamics for liquid metals, *Phys. Rev. B* 47 (1993) 558–561, <https://doi.org/10.1103/PhysRevB.47.558>.
- [33] P.E. Blöchl, Projector augmented-wave method, *Phys. Rev. B* 50 (1994) 17953–17979.
- [34] G. Kresse, D. Joubert, From ultrasoft pseudopotentials to the projector augmented-wave method, *Phys. Rev. B* 59 (3) (1999) 1758, <https://doi.org/10.1103/PhysRevB.59.1758>.
- [35] J.P. Perdew, K. Burke, M. Ernzerhof, Generalized gradient approximation made simple, *Phys. Rev. Lett.* 77 (1996) 3865–3868, <https://doi.org/10.1103/PhysRevLett.77.3865>.
- [36] M. Methfessel, A.T. Paxton, High-precision sampling for Brillouin-zone integration in metals, *Phys. Rev. B* 40 (6) (1989) 3616, <https://doi.org/10.1103/PhysRevB.40.3616>.
- [37] E. Bitzek, P. Koskinen, F. Gähler, M. Moseler, P. Gumbsch, Structural relaxation made simple, *Phys. Rev. Lett.* 97 (2006) 170201, <https://doi.org/10.1103/PhysRevLett.97.170201>.
- [38] S. Smidstrup, A. Pedersen, K. Stokbro, H. Jónsson, Improved initial guess for minimum energy path calculations, *J. Chem. Phys.* 140 (2014) 214106, <https://doi.org/10.1063/1.4878664>.
- [39] D. Sheppard, R. Terrell, G. Henkelman, Optimization methods for finding minimum energy paths, *J. Chem. Phys.* 128 (2008) 134106, <https://doi.org/10.1063/1.2841941>.
- [40] D.R. Alfonso, A.V. Cugini, D.C. Soreescu, Adsorption and decomposition of h₂s on pd(111) surface: a first-principles study, *Catal. Today* 99 (3–4) (2005) 315–322, <https://doi.org/10.1016/j.cattod.2004.10.006>.

- [41] D.R. Alfonso, First-principles studies of h₂s adsorption and dissociation on metal surfaces, *Surf. Sci.* 602 (16) (2008) 2758–2768, <https://doi.org/10.1016/j.susc.2008.07.001>.
- [42] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G.L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A.P. Seitsonen, A. Smogunov, P. Umari, R.M. Wentzcovitch, Quantum espresso: a modular and open-source software project for quantum simulations of materials, *J. Phys. Condens. Matter* 21 (39) (2009) 395502, <https://doi.org/10.1088/0953-8984/21/39/395502>.
- [43] B. Hourahine, B. Aradi, V. Blum, F. Bonafé, A. Buccheri, C. Camacho, C. Cevallos, M. Deshayé, T. Dumitrică, A. Dominguez, et al., Dftb+, a software package for efficient approximate density functional theory based atomistic simulations, *J. Chem. Phys.* 152 (12) (2020) 124101, <https://doi.org/10.1063/1.5143190>.