Creating Gaussian Process Regression Models for Molecular Simulations Using Adaptive Sampling

Matthew J. Burn[1,2] and Paul L. A. Popelier[1,2] *

[1] Manchester Institute of Biotechnology, The University of Manchester, Manchester, M1 7DN, UK

[2] Department of Chemistry, The University of Manchester, Manchester, M13 9PL, UK

ABSTRACT

FFLUX is a new force field that combines the accuracy of quantum mechanics with the speed of force fields, without any link to the architecture of classical force fields. This force field is atom-focused and adopts the parameter-free topological atom from Quantum Chemical Topology (QCT). FFLUX uses Gaussian Process Regression (GPR) (aka kriging) models to make predictions of atomic properties, which in this work are atomic energies according to QCT's Interacting Quantum Atom (IQA) approach. Here we report the adaptive sampling technique Maximum Expected Prediction Error (MEPE) to create data-compact, efficient and accurate kriging models (sub kJ mol^{-1} for water, ammonia, methane and methanol, and sub kcal mol^{-1} for N-methylacetamide (NMA)). The models cope with large molecular distortions and are ready for use in molecular simulation. A brand new press-one-button Python pipeline, called ICHOR, carries out the training.

1 Introduction

The need for fast but accurate force fields continues. Numerous projects in biochemistry and drug design involve simulations of many thousands of atom systems, which is only made possible when employing force fields. Unfortunately, traditional force fields in use today are actually not[1, 2] as reliable as they should be in order to tackle real world systems such as disordered proteins[3], for example. Large discrepancies in resulting structure and dynamics, both between force field versions and when comparing to experiment, already emerge for much simpler systems such as trialanine[4]. This inadequacy[5] is inherently due to the design of the force field itself, the obstinate use of point charges being one issue. Indeed, there is considerable evidence[6] that multipolar electrostatics are the future-proof way forward. Fortunately, incorporating multipole moments is a decision shared by a good number of next-generation[7] force fields such as AMOEBA[8], NEMO[9], SIBFA[10], XED[11], EFP[12], DMACRYS[13], Gaussian Multipole Model (GMM)[14], the Exact Potential Multipole Method (EPMM)[15], and the water potential[16] family ASP-Wn[17] and DPP2[18], which is a non-exhaustive list. In summary, force fields continue to be needed in the foreseeable future but they are only as trustworthy as the assumptions behind them, which includes their mathematical shape and the values of the various parameters present.

FFLUX[19] is a new type of force field that bridges the gap between traditional force fields and ab initio methods. FFLUX uses Gaussian Process Regression[20] (GPR), also known as kriging, to predict quantum mechanical properties, which are used for real-time simulations. At no point are classical expressions of bond (stretch), valence angle (bend) or torsion energy contribution invoked. Instead, FFLUX “sees the electrons” and stays close to the reduced density matrices underpinning the behavior of the molecules. Predictions are made at atomic level where the atoms are defined via Quantum Chemical Topology (QCT)[21, 22]. The energies of these topological atoms (both intra- and inter-atomic) are calculated according to an approach called Interacting Quantum Atom[23] (IQA), which is part of QCT. Multipole moments (include charge transfer) are also calculated according to the topological partitioning method, which is parameter-free and also reference-state free. However, they do not feature in this work, which focuses on intramolecular energy only. The atomic energies are then used to train GPR models, which can subsequently predict atomic energies in previously unseen molecular

geometries. This prediction is essentially an interpolation in the high-dimensional space of the internal coordinates (i.e. machine learning "features"). FFLUX can then provide this quantum mechanical information during simulations but faster than ab initio methods, by orders of magnitude.

Previous work has proven the usefulness of using GPR models to predict the potential energy surfaces of molecules[24]. We have constructed successful models for many small molecule systems such as water[25], NMA[26], capped glycine[27] and alanine[28], as well the multipolar electrostatic energy for all 20 natural amino acids[29], to name a few. The GPR models produced have been used in a number of different ways such as geometry optimisations[30], molecular dynamics simulations of water clusters[25], and proving that GPR models are transferable and can be used in larger systems than what they were trained for[31]. All of the models produced in the past were built with an old pipeline called GAIA[26, 32], which was written in Perl. GAIA allowed for models to be built for FFLUX by providing a standard interface between each program. Unfortunately, the models that GAIA produced were static, the number of training points in the models were fixed and the model would need to be remade if it did not meet the required accuracy for the application. These limitations are all overcome in the current work.

GPR models not only require data for the training of a model but the data are part of the model itself. This characteristic makes the need for acquiring reliable data a crucial step in accurate model building. The traditional solution to this problem is to build large datasets to train the model with, and assume that the data that are required for accurate predictions are within the dataset. If the model fails to produce accurate predictions, then more points will be added to the dataset.

In this paper we present a deterministic approach to building training sets for GPR models in a dynamic way, based on an adaptive sampling method called the maximum expected prediction error[33] (MEPE). This approach leads to the production of models that are reproducible with minimal function evaluations, allowing for fast model training and fast model prediction. We present a new pipeline called ICHOR, which not only provides an interface between each program in the pipeline but also automates the process so that no manual intervention is required.

2 Methods

2.1 Generation of GPR Models for FFLUX using ICHOR

Generating models for use in FFLUX with adaptive sampling involves a complex pipeline of multiple programs including quantum mechanical (QM) calculations, QCT calculations, and GPR machine learning. If done manually, this process can become tedious and time consuming, which is why the in-house Python program ICHOR was developed in order to automate this pipeline.

ICHOR is a one-button-press automation pipeline for the generation of GPR models using adaptive sampling, which takes advantage of HPC clusters to speed up the process by running multiple calculations simultaneously. Figure 1 outlines the flow of ICHOR's data stream and computer programs.

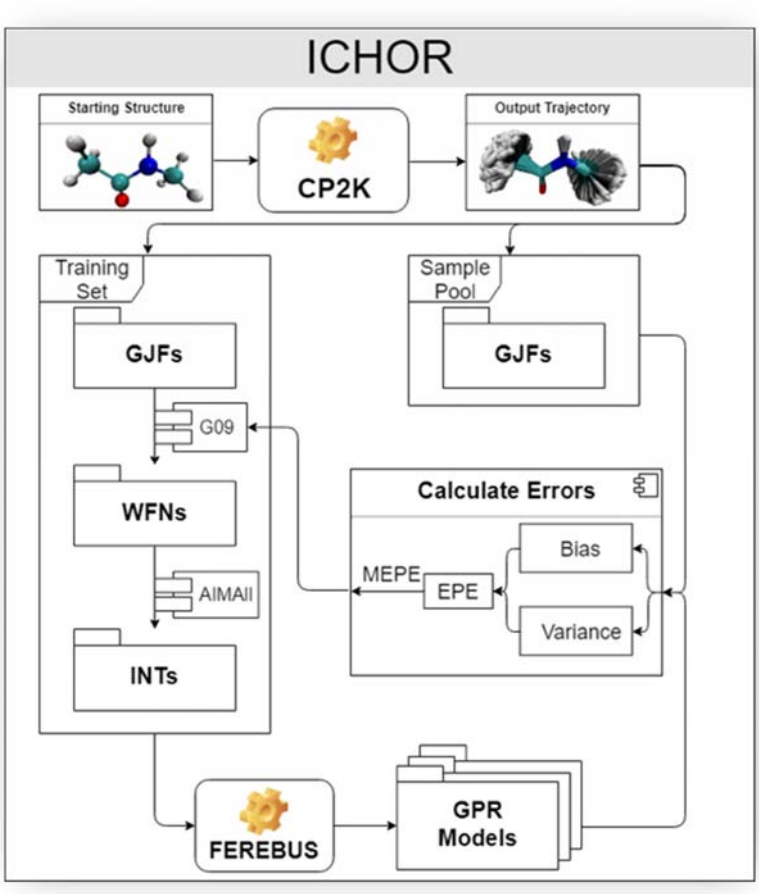

Figure 1. Schematic of ICHOR pipeline for generation of FFLUX models using adaptive sampling.

The process begins with the generation of molecular configurations from an Ab Initio Molecular Dynamics (AIMD) simulation carried out by the program CP2K[34-48]. These molecular configurations are then split into an initial training set consisting of points that will be in the first GPR model and a *sample pool*, which is a random subsample of the remaining points in the AIMD trajectory. The molecular configurations must be converted to GAUSSIAN[49] input files (GJFs) in order to generate wavefunctions (WFNs). These wavefunctions are then given to the program AIMAll[50], which calculates the atomic properties such as IQA energies in the form of "int" files (INTs). The geometries (i.e. input or features) and atomic properties (i.e. output) are then used to generate machine learning GPR models using the in-house software package FEREBUS[51], which outputs a GPR model for each atom. The expected error in prediction of the sample pool can then be calculated from these models and, using the maximum expected prediction error (MEPE) method, a new point can be selected to be added to the training set. This whole cycle is then repeated until the error in the predicted minimum is beneath a threshold (e.g. 1 kJ mol^{-1}) or the average prediction error of a test set is beneath a threshold (e.g. 1 kcal mol^{-1}).

ICHOR also provides an interface to the program DL_FFLUX to test the performance of the GPR models by performing geometry optimizations on the model system. DL_FFLUX is a locally developed derivative of the simulation package DL_POLY[52]. The interface allows for generating input files for the geometry optimization of a configuration that is not already in the training set using the GPR model generated by FEREBUS. The output from this geometry optimization is then tested against the energy minimum found by geometry optimization using GAUSSIAN09 at the same level of theory. The adaptive sampling cycle terminates once a suitable energy accuracy is reached, that is, the energy difference between the minimum energy geometry of FFLUX and that of GAUSSIAN09.

Previous work used a different pipeline implemented in an in-house program called GAIA[26]. The major difference between ICHOR and GAIA is that GAIA does not implement adaptive sampling. A minor current difference is that ICHOR does not implement a process known as *scrubbing*[30]. Scrubbing removes points with large AIMAll integration errors (i.e. large $L(\Omega)$ values). By removing points with large integration errors, the noise in the GPR model is reduced and a better overall fit to the true function is typically achieved. Scrubbing has intentionally been left out in ICHOR because scrubbing

would prevent some points from being added resulting in some iterations of the adaptive sampling having no effect. However, the absence of scrubbing may result in noisier GPR models.

2.2 Computational Details

2.2.1 Structure Generation

AIMD was used for the generation of molecular configurations to put into the training set and sample pool. For the AIMD simulation, the BLYP functional was combined with the 6-31G* basis set and the D3 correction was applied. For the MD simulation, a Nosé-Hoover Chain thermostat was set to a chain length of 3 and a time constant of 50 fs. The simulation was carried out using an NVE ensemble and periodic boundary conditions, with a single molecule in a vacuum. The distortion of the molecule was controlled by a single temperature and the simulation was carried out at 0.5 fs timesteps for 20,000 steps (10 ps). CP2K 6.1[34-48] was used for all AIMD simulations.

2.2.2 DFT Calculations

DFT calculations were used to generate molecular wavefunctions for the geometries used in the training set for the GPR model. These DFT calculations were carried out at B3LYP/6-31+G(d,p) level. All DFT calculations were performed using GAUSSIAN09[49].

2.2.3 Atomic Property Calculations

The atomic energies were obtained from the Interacting Quantum Atoms[23] (IQA) scheme, which is a parameter-free partitioning method that is part of Quantum Chemical Topology (QCT). IQA takes the gradient of the electron density to partition the molecule into atomic volumes to which atomic energies (E_{IQA}^{A}) are associated.

IQA energies can be further partitioned into intra- and interatomic energies, denoted E_{intra}^{A} and $E_{inter}^{AA'}$ where A' refers to all other atoms in the system. The intra- and interatomic energies can be broken down further into kinetic, electrostatic and exchange-correlation energies. In the past these

energies were predicted separately, but here we directly predict the IQA energy to save computing time and reduce cumulative prediction errors.

All IQA calculations were performed using AIMAll 17.11.14[50] with BOAQ=gs10 and IASMESH=fine. All other parameters were set to the default. How the IQA method was made compatible with B3LYP for the first time has been explained before[53].

2.3 Atomic Feature Calculation

Atomic positions are initially represented by atomic coordinates with respect to a global Cartesian axis system. In order to produce atomic models, coordinates need to be transformed from a global frame to a local frame, one for each atom, Figure 2 shows an example of how such an Atomic Local Frame (ALF)[54] is constructed. The atom for which a GPR model is constructed constitutes the origin of the ALF (called A in Figure 2). Two more atoms are then needed to fix the ALF: one to fix the x-axis, denoted atom Ax, and a second one to fix the xy-plane, denoted atom Axy. In order to decide which these two atoms are we follow the Cahn-Ingold-Prelog rules: the x-axis is defined by the atom with highest priority and the xy-plane is defined using the atom with second highest priority. Finally, the z axis is erected orthogonally to the xy-plane to complete a right-handed axes system. The first three features (i.e. machine learning input) are: (i) the distance (typically a bond length) between the ALF's origin A and the first atom Ax, (ii) the distance between the ALF origin and the second atom Axy, and (iii) the angle Ax-A-Axy. Every other atom, not involved in the ALF, is then defined by spherical coordinates relative to the ALF, which yields 3N-9 more features. Thus, in total there are 3+(3N-9)=3N-6 features per atom, a number that corresponds to the well-known number of internal coordinates. As a result of the rotational and translational independence of the coordinates, any atomic GPR model is also rotationally and translationally invariant. Figure 2 shows an example of the construction of an ALF for the amidic carbon in N-methylacetamide (NMA).

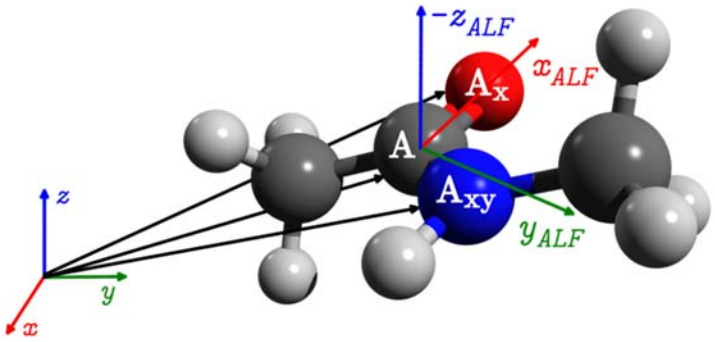

Figure 2. Example of how the atomic local frame (ALF) is defined for the amidic carbon in N-methylacetamide (NMA). Note that the positive z-axis points down in the right-handed axis system that is the ALF.

2.4 Gaussian Process Regression

GPR is a supervised learning technique, which means there is an input vector $\boldsymbol{x}$ and an output $\boldsymbol{y}$. The input $\boldsymbol{x}$ is a vector consisting of n training points, $\boldsymbol{x}_i$,

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} \tag{1}$$

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{2}$$

where each training point, $\boldsymbol{x}_i$, is a D-dimensional vector with D features, corresponding to a scalar output y_i. GPR is a machine learning technique that involves measuring the similarity between two points, $\boldsymbol{x}$ and $\boldsymbol{x}'$. This is done using a covariance or kernel function, k. The kernel function used in this study is the Radial Basis Function (RBF) kernel, which models the similarity between two points using a Gaussian distribution scaled by a set of hyperparameters denoted as a vector $\boldsymbol{\theta}$, one component (i.e. hyperparameter) for each feature. The kernel is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\sum_{d=1}^{D} \theta_d (x_d - x'_d)^2 \right) \tag{3}$$

The hyperparameters $\boldsymbol{\theta}$ scale each dimension so that the relevance of each dimension is automatically determined: the more relevant a dimension, the larger θ_d and *vice versa*. This is commonly referred to as automatic relevance determination. The argument of the exponential must of course be dimensionless, which is why the unit of each θ_d value must be the reciprocal of the unit of the corresponding feature. Thus, the angular features will lead to theta values in radian^{-2} while the distance features lead to theta values in Bohr^{-2}.

A brief comment is in order here on the treatment of cyclic features. As mentioned in Section 2.3, every third feature is a cyclic feature that can vary from $-\pi$ to π. The original RBF kernel would calculate the wrong distance between two angles, that is, not necessarily the shortest distance. For example, 2π/6 (60^o) and 9π/6 (270^o) are really 5π/6 (150^o) away and not 7π/6 (210^o). This is which we apply a cyclic feature correction to every third feature. The cyclic feature correction calculates the shortest distance between two angles, which is explained in detail in the Supplementary Material.

A covariance matrix ($\boldsymbol{R}$) is defined by calculating the covariance of the training set with itself,

$$\boldsymbol{R} = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix} \tag{4}$$

Central to GPR is the natural logarithm of the likelihood function ("log-likelihood"), which is defined as follows, ignoring the constant term –(n/2) ln(2π),

$$\ln L = -\frac{n}{2}\ln(\sigma^2) - \frac{1}{2}\ln(|\boldsymbol{R}|) - \frac{(\boldsymbol{y} - \mathbf{1}\mu)^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - \mathbf{1}\mu)}{2\sigma^2} \tag{5}$$

where *T* marks the transpose and **1** is a column vector of ones. By analytically optimizing the log-likelihood with respect to μ and σ^2 one obtains, the concentrated mean ($\hat{\mu}$) and concentrated variance ($\hat{\sigma}^2$), which are calculated as follows,

$$\hat{\mu} = \frac{\mathbf{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{y}}{\mathbf{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\mathbf{1}} \tag{6}$$

$$\hat{\sigma}^2 = \frac{(\boldsymbol{y} - \mathbf{1}\hat{\mu})^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - \mathbf{1}\hat{\mu})}{n} \tag{7}$$

Inserting these two equations into eq.(5) leads to the concentrated log-likelihood,

$$ln\mathcal{L} = -\frac{n}{2} ln\, \hat{\sigma}^2 - \frac{1}{2} ln|\boldsymbol{R}| \quad (8)$$

Each hyperparameter needs to be optimized by maximizing the concentrated log-likelihood. As $\boldsymbol{R}$ is a function of $\boldsymbol{\theta}$, $ln\mathcal{L}$ is also a function of $\boldsymbol{\theta}$, and in principle one can proceed with analytical optimization[55]. However, in order to explore all corners of this complicated theta landscape one better makes use of global optimization technique. Optimal θ values are found by the in-house program FEREBUS with the aid of a machine leaning method called particle swarm optimization (PSO).

PSO is an optimization algorithm that uses a number of particles to iteratively swarm towards the global maximum by moving towards the personal best found value and the globally best found value.

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \quad (9)$$

$$\boldsymbol{v}_i(t+1) = \omega \boldsymbol{v}_i(t) + c_1 r_{1_i} \left(\boldsymbol{x}_{i_{pb}}(t) - \boldsymbol{x}_i(t) \right) + c_2 r_{2_i} \left(\boldsymbol{x}_{gb}(t) - \boldsymbol{x}_i(t) \right) \quad (10)$$

where $\boldsymbol{x}_i(t)$ is the position of particle i at time t, $\boldsymbol{v}_i(t)$ is the velocity of particle i at time t, ω is the inertia weight, c_1 is the cognitive learning rate, c_2 is the social learning rate, while r_{1_i} and r_{2_i} are random numbers where $r_{n_i} \in [0,1]$. $\boldsymbol{x}_{i_{pb}}$ is the personal best position for particle i and $\boldsymbol{x}_{gb}$ is the global best position found by the swarm. By using swarm intelligence, all particles should move towards a maximum until the swarm obeys a stopping criterion. In our work, the stopping criterion is that the Euclidean distance for all particles between t and $t+1$ falls below the threshold of 1×10^{-4}. A detailed description of the implementation of the PSO algorithm in FEREBUS can be found in the Supplementary Material. The optimized $\boldsymbol{\theta}$ values are used to recalculate the covariance matrix allowing for predictions to be made with the following,

$$\hat{f}(\boldsymbol{x}) = \hat{\mu} + \boldsymbol{r}^{\mathrm{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{1}\hat{\mu}) \quad (11)$$

Where $\boldsymbol{r}$ is the vector defined by calculating the covariance between arbitrary point $\boldsymbol{x}^*$ and every point in the training set,

$$\boldsymbol{r} = \begin{bmatrix} k(\boldsymbol{x}^*, \boldsymbol{x}_1) \\ \vdots \\ k(\boldsymbol{x}^*, \boldsymbol{x}_n) \end{bmatrix} \quad (12)$$

Because $\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu})$ can be precomputed, the prediction is a dot product of two vectors and therefore prediction scales linearly with respect to the number of training points.

2.5 Adaptive Sampling

As the performance of a GPR model relies completely on the data, putting the right points in the training set is of great importance. The majority of machine learning pipelines involves generating a large amount of data from a random sample of the *domain space*. The domain space is the entire space that one strives to model. For our purpose, the domain space is defined by the AIMD simulation. The problem with this approach is that there is no guarantee that the random points selected are the best points to describe the domain space. Secondly, GPRs "take their training set with them" when making predictions, and thus, for computational economy, one wishes the training set to be as compact as possible.

Keeping the training set to a minimum while maintaining a high accuracy is therefore highly favorable and a method of doing so is adaptive sampling. As opposed to random sampling, adaptive sampling takes the model and predicts the point to add to the training set that will improve the model the most. Adaptive sampling aims at minimizing the prediction error of all the points in the domain space. The best way to do this would be at each iteration, by adding the point with the largest prediction error. Unfortunately, this requires the knowledge of the true function value as the prediction error (PE) is defined as follows,

$$PE^2(\boldsymbol{x}) = \left(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})\right)^2 \tag{13}$$

where $f(\boldsymbol{x})$ is the true value and $\hat{f}(\boldsymbol{x})$ is the predicted value for a given function. As the value of the true function is not known, an approximation of the true value must be made. In this study we use the Maximum Expected Prediction Error (MEPE) method. MEPE aims to estimate the prediction error using Leave-One-Out Cross-Validation (LOOCV) and the variance of prediction, by using the following equation,

$${EPE_{CV}}^2(\boldsymbol{x}) = \alpha {PE_{CV}}^2(\boldsymbol{x}) + (1-\alpha)s^2(\boldsymbol{x}) \tag{14}$$

where ${PE_{CV}}^2$ is the cross-validation prediction error for sample point $\boldsymbol{x}$, s^2 is the variance for sample point $\boldsymbol{x}$ and α is a balance factor where $\alpha \in [0, 1)$. The point with the maximum EPE is then selected and added to the training set,

$${MEPE_{CV}}^2(\boldsymbol{x}) = \max\left(\alpha {PE_{CV}}^2(\boldsymbol{x}) + (1-\alpha)s^2(\boldsymbol{x})\right) \tag{15}$$

$PE_{CV}{}^2$ is defined as the error in prediction for a training point $\boldsymbol{x}_i$ using a model with that same training point $\boldsymbol{x}_i$ removed, or

$$PE_{CV}{}^2(\boldsymbol{x}_i) = \left(f(\boldsymbol{x}_i) - \hat{f}^{-i}(\boldsymbol{x}_i)\right)^2 \tag{16}$$

Because point $\boldsymbol{x}_i$ is in the training set, the true value $f(\boldsymbol{x}_i)$ is known. The quantity $\hat{f}^{-i}(\boldsymbol{x}_i)$ is the prediction of $\boldsymbol{x}_i$ using a model that has point i removed. As this value must be computed for every point in the training set, a new model would need to be created for each point. This is computationally expensive, which is why we approximate this value using the following equation,

$$PE_{CV}{}^2(\boldsymbol{x}_i) \approx \left(\frac{(\boldsymbol{R}^{-1})_{i,:}\left(\boldsymbol{d} + \boldsymbol{H}_{:,i}\frac{\boldsymbol{d}_i}{1-\boldsymbol{H}_{ii}}\right)}{(\boldsymbol{R}^{-1})_{ii}}\right)^2 \tag{17}$$

where $\boldsymbol{H}_{:,i}$ is column i of matrix $\boldsymbol{H}$, $(\boldsymbol{R}^{-1})_{i,:}$ is row i of matrix $\boldsymbol{R}^{-1}$ while $\boldsymbol{d}$ and $\boldsymbol{H}$ are calculated using the following set of equations

$$\boldsymbol{d} = \boldsymbol{y} - \boldsymbol{F}\widehat{\boldsymbol{\beta}} \tag{18}$$

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{F})^{-1}\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{y} \tag{19}$$

$$\boldsymbol{H} = \boldsymbol{F}(\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{F})^{-\mathbf{1}}\boldsymbol{F}^{\boldsymbol{T}} \tag{20}$$

$$\boldsymbol{F} = [\boldsymbol{p}(\boldsymbol{x}_1),\ \ldots, \boldsymbol{p}(\boldsymbol{x}_n)]^{\boldsymbol{T}} \tag{21}$$

$$\boldsymbol{p}(\boldsymbol{x}) = [\mathbf{1},\ \ldots,\ \mathbf{1}]^{\boldsymbol{T}} \tag{22}$$

Equations 17-20 show the calculation of $\boldsymbol{d}$ and $\boldsymbol{H}$ for universal GPR, which is a GPR model with mean defined by a set of functions $\boldsymbol{p}(\boldsymbol{x})$. As we are using ordinary GPR (a GPR with constant mean), these equations can be simplified by setting $\boldsymbol{p}(\boldsymbol{x})$ to a vector of 1's shown in equation 22.

Because we are computing the $PE_{CV}{}^2$ value for training point $\boldsymbol{x}_i$ to use this value in the adaptive sampling method, a method of approximating the $PE_{CV}{}^2$of sample point $\boldsymbol{x}$ would be if it were in the training set is required. This is done by producing a Voronoi partition of the domain space using the points in the sample pool and approximating the $PE_{CV}{}^2(\boldsymbol{x})$ by $PE_{CV}{}^2(\boldsymbol{x}_i)$ if point $\boldsymbol{x}$ lies in the Voronoi cell (V_i) created by point $\boldsymbol{x}_i$ as illustrated in Figure 3,

$$\boldsymbol{x} \in V_i \rightarrow PE_{CV}{}^2(\boldsymbol{x}) = PE_{CV}{}^2(\boldsymbol{x}_i) \tag{23}$$

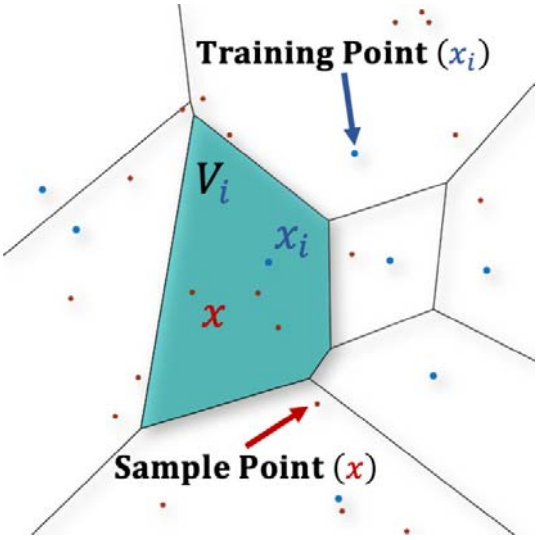

Figure 3. Illustration of the ${PE_{CV}}^2$ approximation for sample point $\boldsymbol{x}$. The green polygon V_i is the Voronoi cell corresponding to the training point x_i, training points are in blue and sample points in red.

${PE_{CV}}^2$ is an estimator of how well a certain region of space is known, this can be thought of as local exploitation. In order for the adaptive sampling to not oversample a specific region, a global exploration term is added, which is the variance s2. The variance of the prediction is easily calculated alongside the predicted mean and indicates how far a given point is away from other points in the training set. A variance of 0 indicates the point is in the training set and therefore the output is known perfectly, whereas a larger variance indicates that the point is in a region of space further away from the training set. The variance is calculated as follows

$$s^2(\boldsymbol{x}) = \sigma^2 \left[1 - \boldsymbol{r}^T \boldsymbol{R}^{-1} \boldsymbol{r} + \frac{(1 - \mathbf{1}^T \boldsymbol{R}^{-1} \boldsymbol{r})^2}{\mathbf{1}^T \boldsymbol{R}^{-1} \mathbf{1}} \right] \tag{24}$$

Once the PE_{CV} and s^2 values are calculated for point $\boldsymbol{x}$, a balance factor, α, is introduced to trade-off between local exploitation and global exploration. The balance factor takes into account how well

the ${PE_{CV}}^2$ estimated the true prediction error (${PE_{true}}^2$) for the point added in the previous iteration ($\boldsymbol{x}_{i+q-1}$). The balance factor α is calculated as follows

$$\alpha = \begin{cases} 0.5, & q = 1 \\ 0.99 \times min\left[0.5 \times \frac{{PE_{true}}^2(\boldsymbol{x}_{i+q-1})}{{PE_{CV}}^2(\boldsymbol{x}_{i+q-1})},\ 1\right], & q > 1 \end{cases} \quad (25)$$

Adaptive sampling using the MEPE method allows for models to be produced in a reproducible and efficient manner. By iteration, models can continually be improved through the addition of more points until the domain space is modelled sufficiently. As the prediction error is approximated using the combination of ${PE_{CV}}^2$ and the Voronoi partition, QCT calculations only need to be carried out on the points that are included in the training set. The sample pool can therefore be as big as required to describe the input space and the adaptive sampling method will only choose the points that are necessary.

2.6 Point Generation

The EPE method requires an initial training set and a sample pool to select points from. To generate the points to put into these sets we use the molecular simulation package CP2K. The points are produced by a single temperature AIMD simulation allowing us to control the distortion of the molecule by raising or lowering the temperature. All points to initialize the training set and sample pool are then taken from the trajectory of this AIMD run.

The initial training set ideally should be space-filling but we are dealing with high-dimensional chemical structures. This means that filling space in high dimensions would require many points. For example, using just 2 points in each dimension, for a 30D system (such as N-methylacetamide (NMA)), we would require 2^{30} starting points, which is totally infeasible. Fortunately, we are working in chemical space and therefore do not need to sample all of space but instead just the feasible chemical structures for a given system.

Using this chemical insight, we can calculate the geometric features described in Section 2.3 for every point in the AIMD trajectory, subsequently select the minimum, maximum and mean value for each feature to add to the training set. This has the advantage of linear scaling: for example, for NMA,

we then start with only 90 points (3×30). The sample pool can subsequently be chosen from a random subsample of the rest of the CP2K trajectory, often using the majority of the points remaining, and thereby leaving roughly 500 points to use as a validation/test set.

3 Results and Discussion

3.1 Creating a Domain Space

To investigate the effectiveness of adaptive sampling on modelling potential energy surfaces (PES), we selected a range of molecules with increasing dimensionality: water, ammonia, methane, methanol and NMA (3D, 6D, 9D, 12D and 30D, respectively). Not only does the increase in system size increase the dimensionality of the input domain but also the conformational complexity.

Each system requires a CP2K AIMD simulation to generate a domain space. As water is a smaller system, the temperature for the simulation will be higher than that for the other systems. For this reason, the water was sampled at 3000 K and all other systems were sampled at 1000 K. Because each system is initialized with the min-max-mean method explained before, each system starts with a number of starting points in the training set amounting to three times the number of features (D) of the molecule being trained for. For the sample pool and validation set, each system will respectively take a random subsample of the rest of the trajectory of 9000 and 500 points.

Generating the training set, sample pool and validation set is a functionality provided by ICHOR. The calculation is limited by the calculation of the geometric features described in the Supplementary Material. These calculations are extremely fast such that the generation of the initial sets is very cheap.

The distortion of the domain space produced by the AIMD simulation can be visualized by means of a so-called mist plot. In this plot each tiny white dot represents a point in space where a nucleus has appeared in a molecular geometry corresponding to a timestep of the simulation trajectory. For clarity, a CPK representation of the initial geometry used in the simulation has been overlayed.
Figure 4 shows the mist plot for each system investigated.

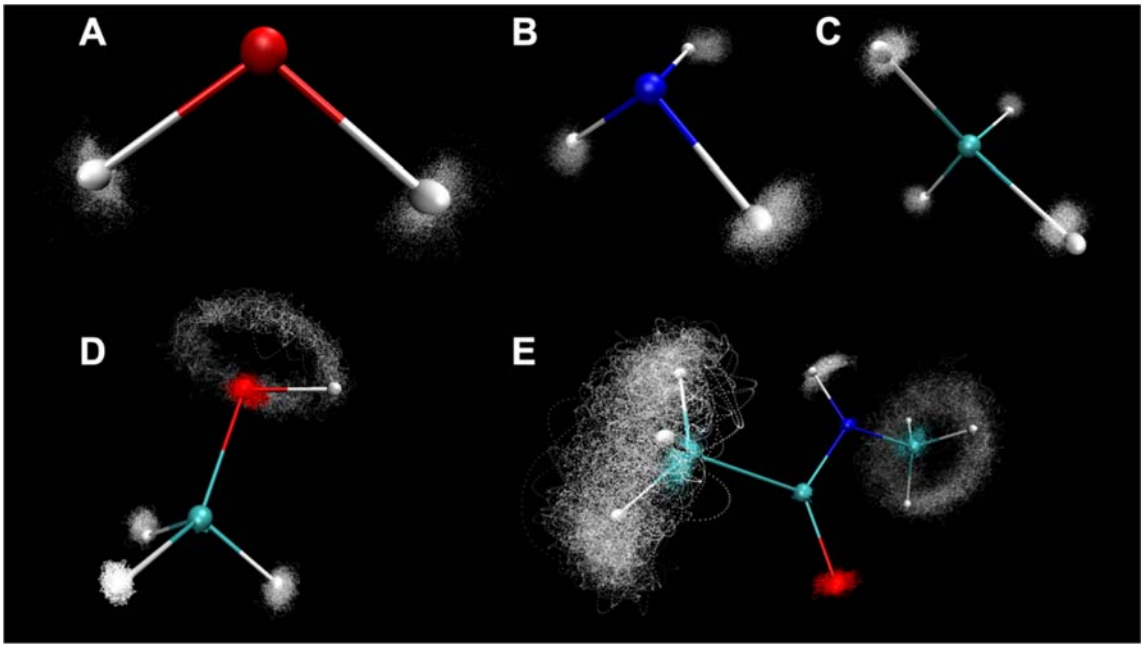

Figure 4. Mist plots showing the trajectory for each AIMD simulation from CP2K for (a) water 3000 K, (b) ammonia 1000 K, (c) methane 1000 K, (d) methanol 1000 K, and (e) NMA 1000 K.

The mist plots shown in Figure 4 were produced by first overlapping all of the points in the trajectory by means of minimizing the root mean square distance (RMSD) of the given timestep with the first point in the trajectory using Kabsch's algorithm. This utility is provided as part of ICHOR outputting a new trajectory file that can be inputted to VMD to render the visualization.

Every point in the mist plot can be added to the training set by the adaptive sampling. All plots display a large distortion in bond length and angles. Importantly, Figures 4d and 4e also show full rotations of the torsional angles with a low-energy barrier. Table 1 shows the distortions of water from a 3000 K AIMD simulation.

Table 1. Bond and angle distortions for water simulated at 3000 K by the program CP2K.

	r(O1-H2) / Å	r(O1-H3) / Å	a(H2-O1-H3) / ˚
Min.	0.84	0.81	70.7
Max.	1.31	1.35	137.4
Avg.	1.00	1.00	102.0

A full list of bond lengths, angles and dihedrals for all systems used in this study can be found in Tables S2 to S7 of the Supplementary Material.

3.2 Geometry Optimizations

In order to test the performance of the GPR models in their predicting the PES, the in-house program DL_FFLUX performs geometry optimizations. DL_FFLUX uses the GPR models produced by FEREBUS to predict the IQA energy for each atom and computes the derivatives of the predicted surface to calculate the forces required to perform a zero-kelvin geometry optimization. The predicted minimum geometry is then tested against the minimum geometry calculated obtained from GAUSSIAN09 by calculating the energy of the predicted minimum at the same level of theory (B3LYP/6-31+G(d,p)) and taking the difference in energy. The closer the predicted minimum to the true minimum geometry, the smaller the energy gap. This process is repeated for each iteration in the adaptive sampling run producing a plot for all molecules, shown in Figure 5. Such a plot tracks the progression of the accuracy of the adaptive sampling in predicting the minimum geometry of the given system.

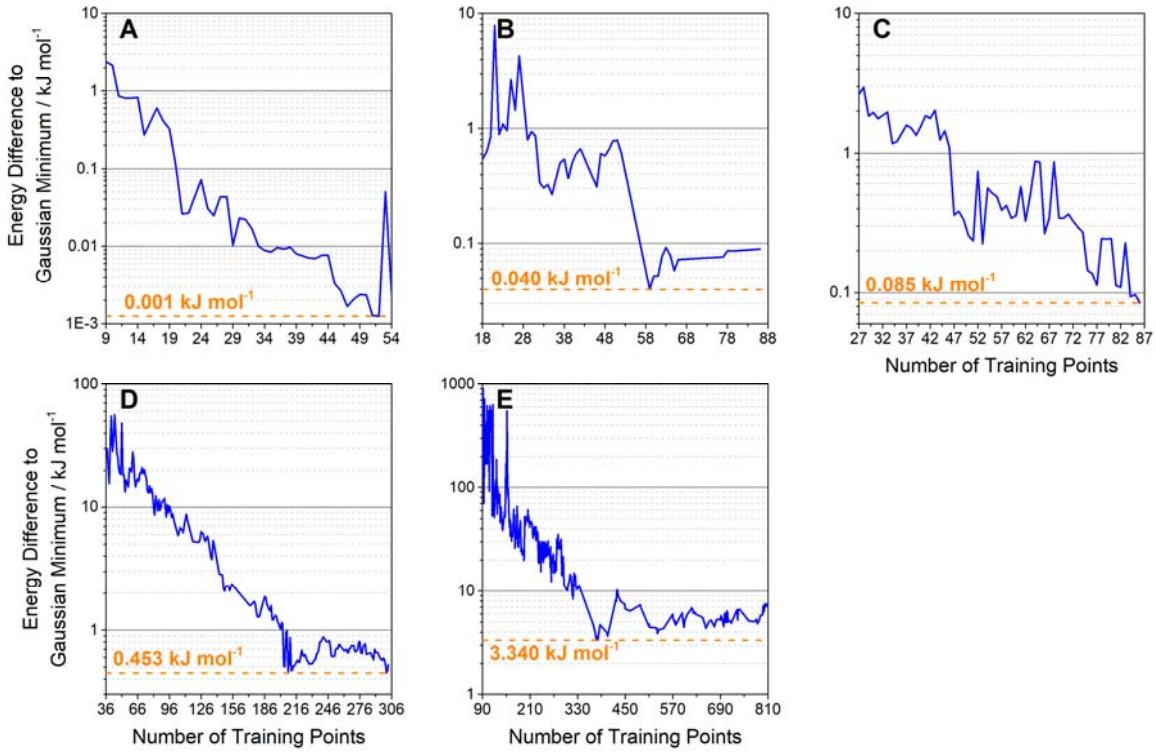

Figure 5. DLPOLY geometry optimization results showing the progression of the accuracy in prediction of the minimum for (a) water, (b) ammonia, (c) methane, (d) methanol, and (e) NMA.

As can be seen from Figure 5, sub kJ mol^{-1} accuracy can be achieved for all molecules except NMA. For example, water already hits energy errors of the order of 0.01 kJ mol^{-1} with only 29 training points. As the dimensionality and chemical complexity of the system increases, the number of points required to describe the system accurately also increases. Still, the 9D system methane, treated without any symmetry constraints, hits 0.1 kJ mol^{-1} with only 84 points. This sort of accuracy is reminiscent of the PES fitting in the so-called Permutationally Invariant Polynomial (PIP) literature[56, 57] where spectroscopic accuracy is aimed for. This approach dates from 2003, when a global PES for CH_5^+ was reported using a basis of polynomials that are invariant with respect to the 120 permutations of the five equivalent H atoms. Errors are typically reported in cm^{-1} (=12 J mol^{-1}) such that ~100 cm^{-1} is equivalent to the 1 kJ mol^{-1} threshold we ultimately aim for. Our 84-point methane model thus has an

error of the order of 10 cm^{-1}. Very recently[58], PIP has been integrated with GPR, from which work we can quote a root mean square error (RMSE) of 12 cm^{-1} for a 5000-point PIP model for H_3O^+, for an energy range of ~21,000 cm^{-1} (or ~250 kJ mol^{-1}).

In summary, adaptive sampling allows for a very accurate model to be produced with relatively few points. Even for the largest system investigated (NMA), sub kcal mol^{-1} accuracy was demonstrated in fewer than 400 points for a large domain space (1000 K).

3.3 Accuracy in Predictions

Performing geometry optimizations shows that the shape of the potential energy surface is accurate. To gain an understanding of how accurate the model is at predicting the true function values, one must look at the prediction errors. The following definition of prediction error will be used.

$$PE(\boldsymbol{x}) = \sqrt{\left(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})\right)^2} \tag{26}$$

A validation set was constructed for each system using a random 500-point subset of the AIMD trajectory, left after forming the initial training set and sample pool. A model from the output of the adaptive sampling is then used to make predictions. As there is a model for each atom, the prediction error for the system is defined as the sum of the atomic prediction errors.

$$PE_{total}(\boldsymbol{x}) = \sum_{i=1}^{N_{atoms}} \sqrt{\left(f_i(\boldsymbol{x}) - \hat{f}_i(\boldsymbol{x})\right)^2} \tag{27}$$

where $f_i(\boldsymbol{x})$ is the IQA energy for atom i and $\hat{f}_i(\boldsymbol{x})$ is the predicted IQA energy for atom i. The total prediction errors are sorted and plotted versus their percentile producing the S-curves shown below in Figure 6, so-called because of their typical sigmoidal shape. S-curves are cumulative error distributions and if such a curve, for example, intersects the 50% percentile at 1 kJ mol^{-1} (see water, Fig. 6a) then this means that half of the validation geometries are predicted with an energy less than 1 kJ mol^{-1}. The more an S-curve is shifted to the left, including the point at which its maximum tail hits the 100% ceiling, the better the performance of the model. Also, the mean prediction error is a good indicator of the overall performance of the model in predicting the validation set.

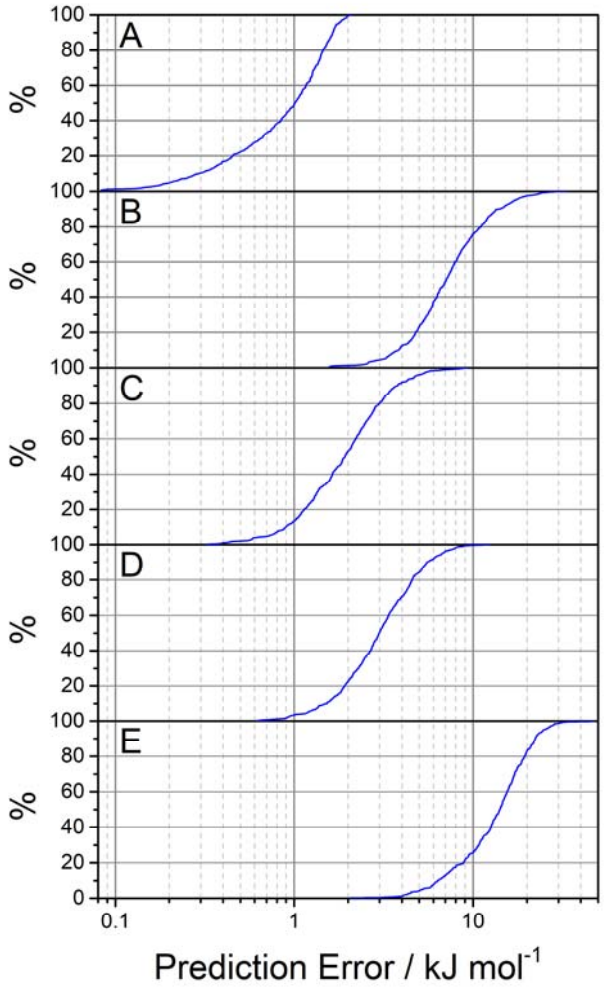

Figure 6. S-Curves plotting the total prediction error of a 500-point validation set vs the percentile of the point for (a) water 54-point model 3000 K validation set, (b) ammonia 86-point model 1000 K validation set, (c) methane 86-point model 1000 K validation set, (d) methanol 303-point model 1000 K validation set, and (e) NMA 805-point model 1000 K validation set.

In general, the average prediction error increases with the dimensionality. However, it seems that ammonia breaks this trend and has a higher RMSE than both methane (9D) and methanol (12D). A possible explanation for this anomaly is the observation that the hydrogens have a larger range of motion than the equivalent hydrogens in methane or methanol, as can be seen in Figure 4. The

increased distortion at the same temperature is due to the hydrogen or carbon atom in methane and methanol respectively being replaced by a lone pair in ammonia giving the ammonia higher conformational flexibility. The anomaly in the trend seen in Figure 6 suggests that the performance of the model is not only a function of the dimensionality of the system but also of the system's conformational flexibility, i.e. the more flexible the system, the harder it is to produce an accurate model.

Table 2. The RMSE for each system's model.

System	Number of Training Points	RMSE / kJ mol^{-1}
Water	54	0.98
Ammonia	86	8.18
Methane	86	2.19
Methanol	303	3.35
NMA	805	14.39

3.4 Assessment of NMA Prediction Errors

As the largest and most complex system, NMA is the most difficult system to model. NMA is also the closest system to modelling a peptide bond found in many biological systems, and therefore it is one of the most important systems to model accurately. One of the benefits of predicting atomic properties is that each atom can be analyzed separately. The prediction error for the IQA energy of each atom can be calculated separately using Eq.(26), and each model is plotted as a separate S-curve shown in Figure 7.

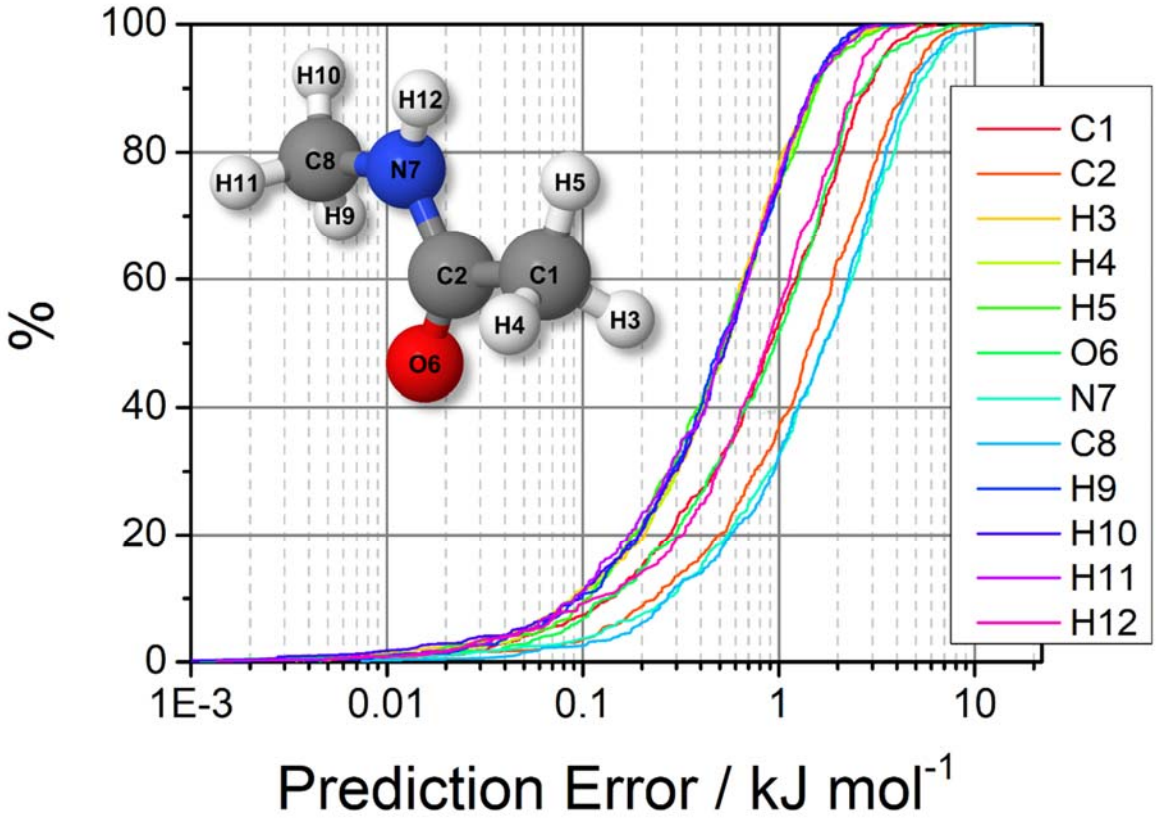

Figure 7. S-Curve for each individual atom in NMA showing the atom's prediction error *versus* the percentile for an 805-point model with a 1000 K validation set. The atom labelling used for NMA is overlayed.

As can be seen from Figure 7, it is the methyl carbons (C1 and C8) that are the worst performing models. From the mist plot in Figure 4e it is clear that the methyl groups are 'swinging' from side to side resulting in a large search space that must be sampled. However, these methyl groups are actually artificial in the context of NMA being used as a model of a peptide bond in an oligopeptide. In that environment, C1 and C8 are part of the protein backbone and thus do not spin like in a methyl.

The S-curve shown in Figure 6e was produced using a validation set taken from the same AIMD run that produced the search space (1000 K). However, real simulations are carried out at much lower

temperatures, closer to 300 K. The same procedure of taking a validation set can be used on a separate AIMD run taken at the lower temperature and resulting in a new validation set consisting of 500 points at 300 K. A new S-curve can be produced with the new validation set, shown in Figure 8 alongside the previous S-curve from the 1000 K simulation (see Figure 6e).

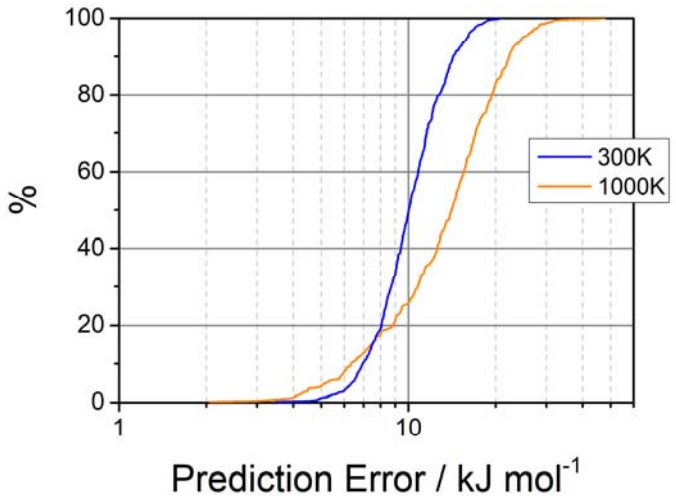

Figure 8. S-Curves of NMA predictions of an 805-point model with a validation set produced with a 300 K AIMD simulation and an equivalent simulation run at 1000 K.

Figure 8 shows that the model performs much better on the 300 K validation set than it does on the 1000 K validation set. This is good news because the 300 K validation points are the points the model will more likely encounter. Table 3 gives the RMSE values.

Table 3. Table showing the RMSE for an 805-point model on a validation set produced by either a 300 K or 1000 K validation set.

Validation Set Temperature / K	RMSE / kJ mol^{-1}
300	10.44
1000	14.39

Another analysis of prediction errors can be made by directly comparing the true energy of the system with the energy predicted by the GPR model. The true energy of the system is calculated by summing the IQA energies for the system and the predicted energy of the system is calculated by summing the predicted IQA energies of each atom. Figure 9 shows the true *versus* predicted energy plot.

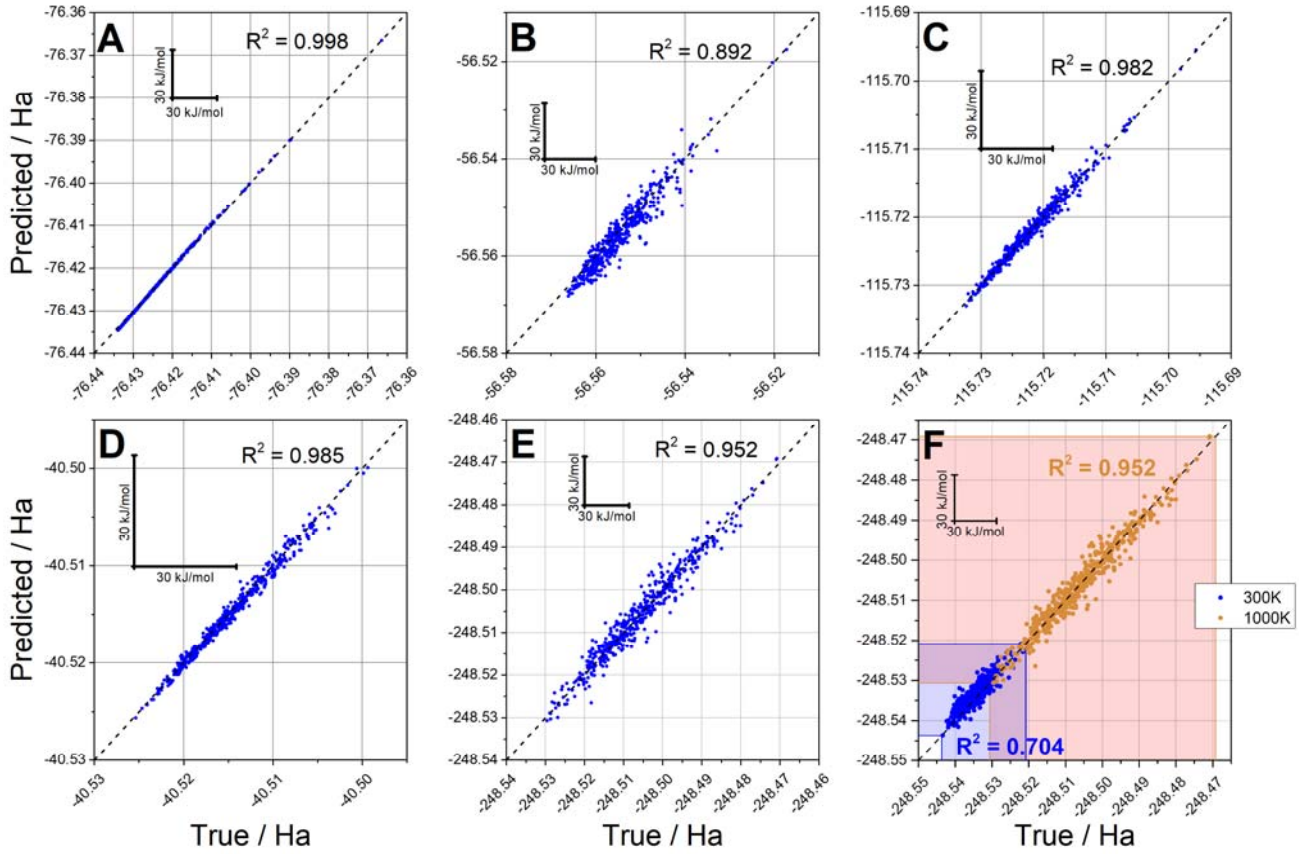

Figure 9. The true *versus* predicted energies for a (a) 54-point water model with 3000 K validation set, (b) 86-point ammonia model with 1000 K validation set, (c) 86-point methane model with 1000 K validation set, (d) 303-point methanol model with 1000 K validation set, (e) 805-point NMA model with 1000 K validation set, and (f) 805-point NMA model with 300 K and 1000 K validation sets.

Each model demonstrates excellent predictions across a wide range of input values. Each panel shows the R^2 value of the validation points measuring the deviation from the ideal $y = x$. All plots have R^2 values close to 1 indicating that the models predict the true values very well. Figure 9f shows the worst model (R^2= 0.70), which is the 805-point NMA model used on the 300 K validation set. Figure 9f also shows the 1000 K true *versus* predicted plot, which highlights the energy range covered by the validation set (≈154 kJ mol^{-1}) in orange, as well as the range of energies covered by the 300 K validation set shown in blue. A large portion of the points being predicted in the 300 K validation set lie outside of the range covered in the 1000 K AIMD simulation indicating why the 805-point 1000 K model is worse at predicting the 300 K validation set compared to the 1000 K validation set. However, we should keep in mind that the 1000 K model is essentially asked to make extrapolatory predictions in the purely blue part of the energy range. That the GPR does not break down more catastrophically (R^2= 0.70 is still reasonable) in its prediction bodes well.

The solution to the poor predictions of a different training set lies in the construction of the domain space. As outlined in Section 2.2.1, the domain space is created using an AIMD simulation in CP2K using a single temperature allowing for control over the molecular distortions. In order to sample as much of the domain space as possible, the temperature for this study is set to 1000 K, deliberately providing large distortions as a severe test. Unfortunately, fixing the temperature so high leads to the overall energy of all molecules to increase and thereby excluding the low energy geometries from the domain space. The GPR model copes well with this sampling deficiency and still predicts points outside of the domain space described by the training set with an R^2 of ≈0.7. However, a better model may be constructed by moving away from a single temperature sampling method, which is currently being investigated in our lab.

3.5 Assessment of Adaptive Sampling

The above report demonstrates the usefulness of adaptive sampling in creating small, accurate models for single molecule systems. It has been shown that such models are able to accurately predict the minimum geometry of the system as well as accurately predicting the total energy of the system.

All domain spaces for this paper were created by AIMD simulations at high temperatures allowing for extremely flexible models that can be used for real MD simulations of systems at room temperature. We briefly compare the current results with our previous work on producing GPR models for single molecule systems, which produced the initial input geometries by sampling from normal mode distortions generated by the in-house program TYCHE[59]. As expected, the current molecular distortions are much larger. As an example, the NMA model constructed from TYCHE had a domain space with an energy range[26] of ≈84 kJ mol^{-1} compared to the new model with a range of ≈154 kJ mol^{-1}. The great increase in flexibility of the model should provide models that are valid over a larger temperature range and improve the stability of the simulation (i.e. avoiding molecules being ripped apart or explode).

Not only does adaptive sampling allow for accurate models, but also with drastically fewer points than was previously available. Adaptive sampling provides a deterministic method for creating models, with the benefit of creating of accurate models to arbitrary precision and reproducible across all the systems shown. Previous model-creation pipelines[26, 32] involved selecting points randomly from a domain space with the hope of generating a sufficiently accurate model. This way of working involved many redundant calculations and no clear way to improve the model produced other than merely adding more random points and repeating the tests. Adaptive sampling ensures that expensive IQA calculations are required only for the points in the training set while the best points to add to the training set can be predicted using just the geometry (i.e. without IQA).

3.6 Recovery Errors

Alongside scrubbing mentioned above, recovery errors are a statistic to be aware of whilst producing GPR models. The recovery error of a system is the difference between the true wavefunction energy and the sum of the IQA energies. In an ideal world, these values would be identical but due to numerical

errors and integration errors, there can be a significant discrepancy affecting the final GPR model. The recovery error is defined as follows

$$E_{recovery}(\boldsymbol{x}) = \sqrt{\left(E_{wfn}(\boldsymbol{x}) - \sum_{i=1}^{N_{atom}} E_{IQA_i}(\boldsymbol{x})\right)^2} \tag{28}$$

where $E_{recovery}$ is the recovery error, E_{wfn} is the (original GAUSSIAN) wavefunction energy and E_{IQA_i} is the IQA energy for atom i. The recovery error can be calculated for every point in the training set and visualized as a histogram. In general, a recovery error of less than 1 kJ mol^{-1} is desirable but recovery error also increases with system size. Consequently, larger systems will result in higher recovery errors and a noisier GPR model, which can be seen in Figure 10.

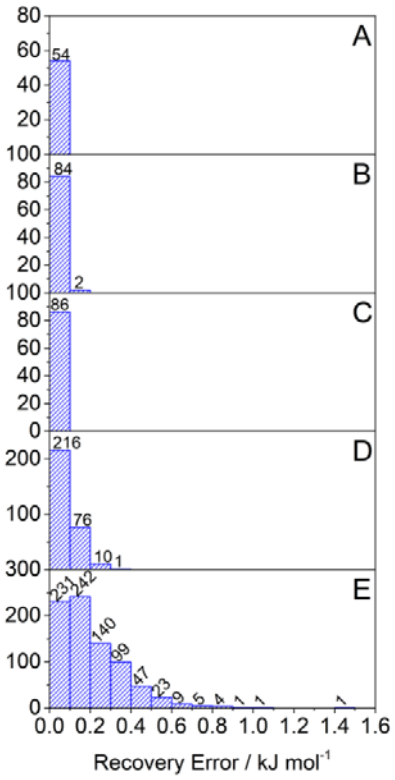

Figure 10. Histogram of the recovery errors for the training set of (a) water (54 points), (b) ammonia (86 points), (c) methane (86 points), (d) methanol (303 points), and (e) NMA (805 points).

As predicted, the larger the system the larger the recovery error with the exception of ammonia, which exhibits 2 points that fall in the error range of 0.1 - 0.2 kJ mol^{-1}. In total there are only 2 points, both part of the NMA model, that are greater than the desired 1 kJ mol^{-1} threshold. In this example, the NMA recovery errors are generally in an acceptable noise range but could become a concern with larger systems. The recovery errors can be improved by increasing the accuracy of the IQA calculations but with the consequence of increased computational cost.

4 Conclusions

FFLUX overhauls the architecture of classical force fields by replacing the typical energy types of bond (stretch), valence (bend), torsion (dihedral) and out-of-plane potentials by atom-based energy predictors. Such a predictor is a machine-learnt (kriging) model that focuses on the atom's internal energy and interatomic interaction energy. Every atom interacts with any other atom in the small but already high-dimensional systems (water, ammonia, methane, methanol and NMA) that we investigated. As such, FFLUX is free from the constraint of a Lewis diagram and allows for any possible coupling of classical energy types.

The training of FFLUX's kriging model has been majorly updated by implementing adaptive sampling into its brand new pipeline called ICHOR, which replaces the old pipeline GAIA. ICHOR is a press-one-button Python script that removes the guesswork out of model making. It is now so that the true function needs to be evaluated only for points that are included in the training set, which substantially reduces the computational cost associated with the topological energy partitioning method IQA. ICHOR also allows for the automation of the machine learning pipeline making the model creation as easy as providing an input molecule. All aspects of the pipeline have been upgraded or changed such as the program CP2K (AIMD single-molecule temperature-controlled molecular distortion) replacing the program TYCHE (normal mode distortion) in order to allow larger molecular distortions, including full torsion rotation.

All systems other than NMA achieved a minimum-energy-geometry energy of less than 1 kJ mol^{-1} from the minimum calculated using DFT, and all systems predict well the true energy of the system.

NMA demonstrated the competency of the model by even predicting points that lay outside of the initial search domain while keeping a relatively high accuracy.

Kriging models of water and methanol are currently being tested in molecular simulations of their neat liquids, as well as a mixture using the program DL_FFLUX, which is a FFLUX-adapted parallellised in-house version of the widely distributed simulation package DL_POLY. To move towards the goal of simulating polypeptides, models for single amino acids are required. Future work will focus on producing models that are effective enough to be used in real world applications. The next big step is for kriging models to be 'stitched' together so that oligopeptides can be predicted for, from monomeric amino acids only, by invoking the atomic transferability offered by QCT.

Supplementary Material

(1) Feature Calculation, (2) Gaussian Process Regression Training: 2.1 cyclic feature correction, 2.2 Noisy Gaussian Process Regression, 2.3 hyperparameter optimization, (3) CP2K distortions: 3.1 Water 300 K, 3.2 Ammonia 1000 K, 3.3 Methane 1000 K, 3.4 Methanol, 2000 K, 3.5 NMA 300 K and 1000 K.

ACKNOWLEDGEMENTS

M.J.B acknowledges the MRC DTP for the award of a PhD studentship and P.L.A.P the EPSRC for funding through the award of an Established Career Fellowship (grant EP/K005472).

AUTHOR INFORMATION

Corresponding Author

*Phone: +44 161 3064511. E-mail: pla@manchester.ac.uk

Present Address

†Present Address: Manchester Institute of Biotechnology, The University of Manchester, Manchester, M1 7DN, UK

DATA AVAILABILITY STATEMENT

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

REFERENCES

1. D.Verbaro, I. Ghosh, W. M. Nau and R. Schweitzer-Stenner, J.Phys.Chem.B, **114**, 17201–17208 (2010).

2. M. Dean Smith, S. Rao, E. Segelken and L. Cruz, J.Chem.Inf.Model. **55**, 2587−2595 (2015).

3. S. Rauscher, V. Gapsys, M. J. Gajda, M. Zweckstetter, B. L. de Groot and H. Grubmüller, J.Chem.Theor.Comput. **11**, 5513-5524 (2015).

4. Y. Mu, D. S. Kosov and G. Stock, J.Phys.Chem.B **107**, 5064-5073 (2003).

5. A. T. Hagler, J.Comp.-Aid. Mol.Design **33**, 205–264 (2019).

6. S. Cardamone, T. J. Hughes and P. L. A. Popelier, Phys.Chem.Chem.Phys. **16**, 10367-10387 (2014).

7. A. Albaugh, H. A. Boateng, R. T. Bradshaw, O. N. Demerdash, J. Dziedzic, Y. Mao, D. T. Margul, J. Swails, Q. Zeng, D. A. Case, P. Eastman, L.-P. Wang, J. W. Essex, M. Head-Gordon, V. S. Pande, J. W. Ponder, Y. Shao, C.-K. Skylaris, I. T. Todorov, M. E. Tuckerman and T. Head-Gordon, J.Phys.Chem.B **120** (37), 9811–9832 (2016).

8. P. Y. Ren, C. J. Wu and J. W. Ponder, J.Chem.Theory Comput. **7** (10), 3143-3161 (2011).

9. A. Holt, J. Boström, G. Karlström and R. Lindh, J.Comput.Chem. **31** (8), 1583-1591 (2010).

10. R. Chaudret, N. Gresh, O. Parisel and J. P. Piquemal, J.Comput.Chem. **32** (14), 2949-2957 (2011).

11. J. G. Vinter, J.Comput.Aided Mol. Des. **8**, 653-668 (1994).

12. D. Ghosh, D. Kosenkov, V. Vanovschi, C. F. Williams, J. M. Herbert, M. S. Gordon, M. S. Schmidt and L. V. Slipchenko, J.Phys.Chem.A. **114**, 12739-12754 (2010).

13. D. A. Bardwell, C. S. Adjiman, Y. A. Arnautova, E. Bartashevich, S. X. M. Boerrigter, D. E. Braun, A. J. Cruz-Cabeza, G. M. Day, R. G. Della Valle, G. R. Desiraju, B. P. van Eijck, J. C. Facelli, M. B. Ferraro, D. Grillo, M. Habgood, D. W. M. Hofmann, F. Hofmann, K. V. J. Jose, P. G. Karamertzanis, A. V. Kazantsev, J. Kendrick, L. N. Kuleshova, F. J. J. Leusen, A. V. Maleev, A. J. Misquitta, S. Mohamed, R. J. Needs, M. A. Neumann, D. Nikylov, A. M. Orendt, R. Pal, C. C. Pantelides, C. J. Pickard, L. S. Price, S. L. Price, H. A. Scheraga, J. van de Streek, T. S. Thakur, S. Tiwari, E. Venuti and I. K. Zhitkov, Acta Cryst.B **67** (6), 535-551 (2011).

14. R. J. Wheatley and J. B. O. Mitchell, J.Comput.Chem. **15**, 1187-1198 (1994).

15. A. Volkov, H. F. King, P. Coppens and L. Farrugia, Acta Cryst. **A62**, 400−408 (2006).

16. P. Ren and J. W. Ponder, J.Phys.Chem.B **107**, 5933-5947 (2003).

17. M. P. Hodges, A. J. Stone and S. S. Xantheas, J.Phys.Chem.A **101** (48), 9163-9168 (1997).

18. R. Kumar, F.-F. Wang, G. Jenness and K. A. Jordan, J.Chem.Phys. **132**, 014309 (2010).

19. P. L. A. Popelier, Int.J.Quant.Chem. **115**, 1005–1011 (2015).

20. C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning.* (The MIT Press, Cambridge, USA, 2006).

21. R. F. W. Bader, *Atoms in Molecules. A Quantum Theory.* (Oxford Univ. Press, Oxford, Great Britain, 1990).

22. P. L. A. Popelier, in *The Nature of the Chemical Bond Revisited*, edited by G. Frenking and S. Shaik (Wiley-VCH, Chapter 8, 2014), pp. 271-308.

23. M. A. Blanco, A. Martín Pendás and E. Francisco, J.Chem.Theor.Comput. **1**, 1096-1109 (2005).

24. Z. E. Hughes, J. C. R. Thacker, A. L. Wilson and P. L. A. Popelier, J.Chem.Theor.Comp. **15**, 116-126 (2019).

25. Z. E. Hughes, E. Ren, J. C. R. Thacker, B. C. B. Symons, A. F. Silva and P. L. A. Popelier, J.Comput.Chem. **41**, 619–628 (2019).

26. P. Maxwell, N. di Pasquale, S. Cardamone and P. L. A. Popelier, Theor.Chem.Acc. **135**, 195 (2016).

27. J. C. R. Thacker, A. L. Wilson, Z. E. Hughes, M. J. Burn, P. I. Maxwell and P. L. A. Popelier, Mol.Simul. **44**, 881-890 (2018).

28. M. J. L. Mills and P. L. A. Popelier, Theor.Chem.Acc. **131**, 1137-1153 (2012).

29. T. L. Fletcher and P. L. A. Popelier, J.Chem.Theor.Comput. **12** (6), 2742-2751 (2016).

30. F. Zielinski, P. I. Maxwell, T. L. Fletcher, S. J. Davie, N. Di Pasquale, S. Cardamone, M. J. L. Mills and P. L. A. Popelier, Scientific Reports **7** (1), 12817 (2017).

31. T. L. Fletcher and P. L. A. Popelier, J.Comput.Chem. **38**, 1005-1014 (2017).

32. S. M. Kandathil, T. L. Fletcher, Y. Yuan, J. Knowles and P. L. A. Popelier, J.Comput.Chem. **34**, 1850-1861 (2013).

33. H. Liu, J. Cai and Y.-S. Ong, Comp.Chem.Eng. **106**, 171-182 (2017).

34. U. Borštnik, J. Vandevondele, V. Weber and J. Hutter, **40** (5-6), 47-58 (2014).

35. M. Frigo and S. G. Johnson, Proceedings of the IEEE **93** (2), 216-231 (2005).

36. S. Goedecker, M. Teter and J. Hutter, Physical Review B **54** (3), 1703-1710 (1996).

37. S. Grimme, J. Antony, S. Ehrlich and H. Krieg, The Journal of Chemical Physics **132** (15), 154104 (2010).

38. S. Grimme, S. Ehrlich and L. Goerigk, Journal of Computational Chemistry **32** (7), 1456-1465 (2011).

39. C. Hartwigsen, S. Goedecker and J. Hutter, Physical Review B **58** (7), 3641-3662 (1998).

40. J. Hutter, M. Iannuzzi, F. Schiffmann and J. Vandevondele, Wiley Interdisciplinary Reviews: Computational Molecular Science **4** (1), 15-25 (2014).

41. J. Kolafa, Journal of Computational Chemistry **25** (3), 335-342 (2004).

42. M. Krack, Theor Chem Acc, **114** (1-3), 145-152 (2005).

43. S. Nosé, The Journal of Chemical Physics **81** (1), 511-519 (1984).

44. S. Nosé, Molecular Physics **52** (2), 255-268 (1984).

45. J. P. Perdew, K. Burke and M. Ernzerhof, Physical Review Letters **77** (18), 3865-3868 (1996).

46. O. Schütt, P. Messmer, J. Hutter and J. Vandevondele, (John Wiley & Sons, Ltd, 2016), pp. 173-190.

47. J. Vandevondele and J. Hutter, The Journal of Chemical Physics **118** (10), 4365-4369 (2003).

48. J. Vandevondele, M. Krack, F. Mohamed, M. Parrinello, T. Chassaing and J. Hutter, Computer Physics Communications **167** (2), 103-128 (2005).

49. M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, J. A. Montgomery Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, O. Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski and D. J. Fox, (Wallingford, CT, 2009).

50. T. A. Keith, AIMALL program, Overland Park KS, USA, 2017.

51. N. Di Pasquale, M. Bane, S. J. Davie and P. L. A. Popelier, J.Comput.Chem. **37**, 2606-2616 (2016).

52. I. T. Todorov, W. Smith, K. Trachenko and M. T. Dove, J.Mater.Chem. **16**, 1911-1918 (2006).

53. P. Maxwell, A. Martín Pendás and P. L. A. Popelier, PhysChemChemPhys **18**, 20986-21000 (2016).

54. M. J. L. Mills and P. L. A. Popelier, J.Chem.Theory Comput. **10**, 3840−3856 (2014).

55. N. Di Pasquale, S. J. Davie and P. L. A. Popelier, J.Chem.Theor.Comp. **12**, 1499-1513 (2016).

56. B. J. Braams and J. M. Bowman, Int.Rev.Phys.Chem. **28**, 577-606 (2009).

57. C. Qu, Q. Yu and J. M. Bowman, Annu.Rev.Phys.Chem. **69**, 6.1-6.25 (2018).

58. C. Qu, Q. Yu, B. L. Van Hoozen, J. M. Bowman and R. A. Vargas-Hernández, J.Chem.Theor.Comp. **14** (7), 3381-3396 (2018).

59. T. J. Hughes, S. Cardamone and P. L. A. Popelier, J.Comput.Chem. **36**, 1844–1857 (2015).